

CSE 4020/5231 - Database Systems Project

Lingjing Huang, Shreyas Ugemuge

December 1, 2017

Contents

Problem Statement	3
1 ER-Diagram	4
2 Relational Representation and Functional Dependencies	4
2.1 Relations	4
2.2 Functional Dependencies	4
2.2.1 Doctor	4
2.2.2 Patient	5
2.2.3 Illness	5
3 Decompositions	5
3.1 Decomposition 1	5
3.1.1 Decomposing Doctor	5
3.1.2 Proof: Lossless Join and Functional Preserving	5
3.2 Decomposition 2	6
3.2.1 Decomposing Patient	6
3.2.2 Proof: Lossless Join and Functional Preserving	6
4 SQL DDLs	6
4.1 Version 1	6
4.2 Version 2	7
4.3 Version 3	8
5 Filling The database with data	9
5.1 Data Generation	9
5.2 Data Insertion	9
6 Copy the data from the first version to the decompositions	10
6.1 Copy data to Version 2	10
6.2 Copy data to Version 3	10

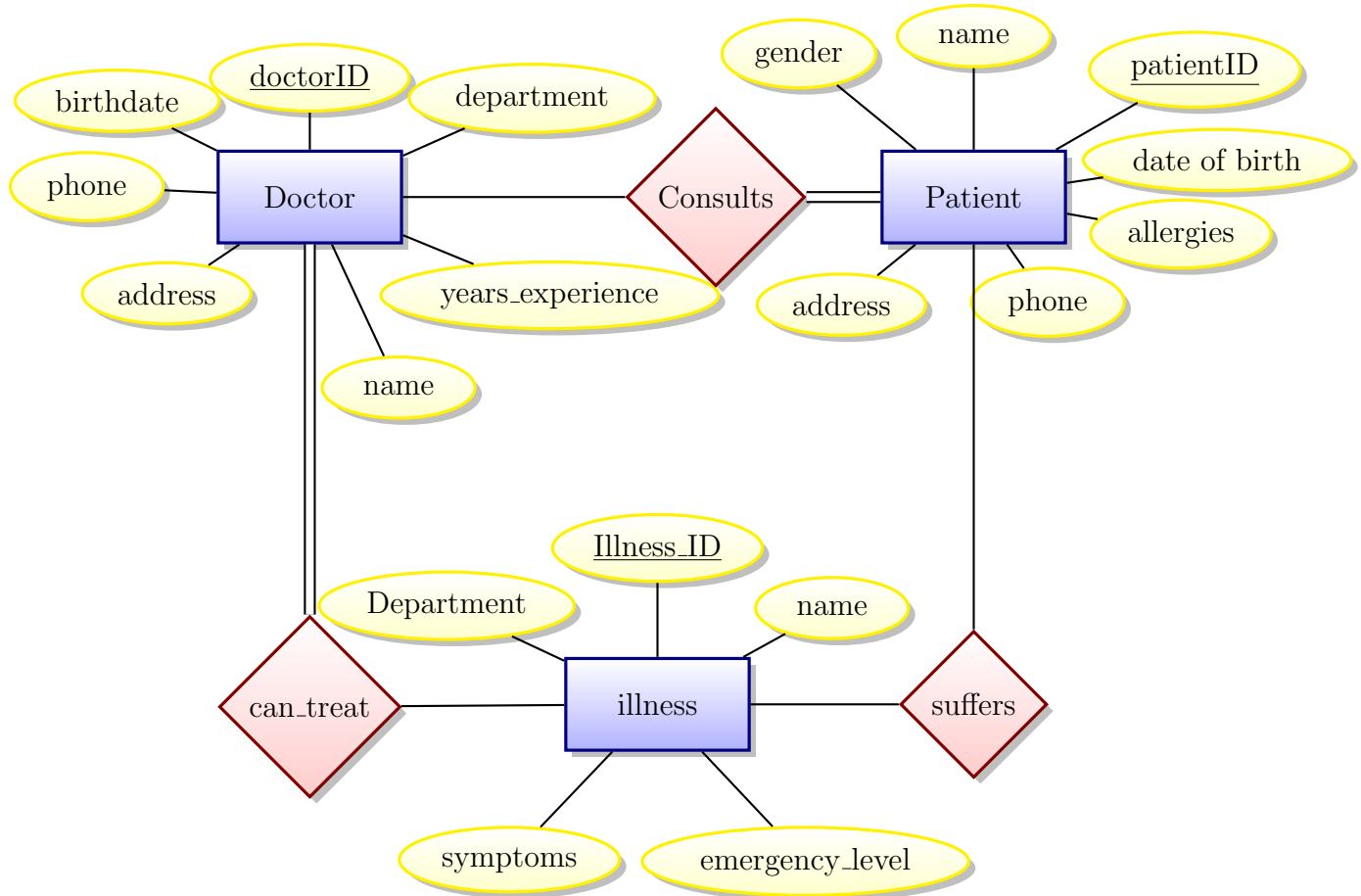
7	English Queries	10
7.1	First English Query	10
7.2	Second English Query	11
7.3	Third English Query	11
8	SQL implementations of the queries	11
8.1	First English Query	11
8.1.1	Version 1	11
8.1.2	Version 2	11
8.1.3	Version 3	11
8.2	Second English Query	11
8.2.1	Version 1	12
8.2.2	Version 2	12
8.2.3	Version 3	12
8.3	Third English Query	12
8.3.1	Version 1	12
8.3.2	Version 2	12
8.3.3	Version 3	13
9	Test the time	13
10	References	13

Problem Statement

Design a database for storing information about medical doctors, their patients, and illnesses (using sqlite3):

1. Draw its E-R diagram.
2. Give a relational representation of the E-R diagram that allows for 2 functional preserving and lossless join decompositions. Describe its functional dependencies.
3. Give the two decompositions and prove that they are lossless join and functional preserving.
4. Give the SQL DDLs for creating the three versions of the database.
5. Fill the first database with data (at least 100 rows in each table).
6. Give the SQL queries to copy the data from the first version into its 2 decompositions.
7. Propose in english 3 queries that require at least 2 table joins each and such that all tables are involved in at least 2 queries.
8. Propose SQL implementations of the 3 queries on all three versions of the database.
9. Test the time in ns for exeuting the 3 queries on each database, by running each of them 1000 times.

1 ER-Diagram



2 Relational Representation and Functional Dependencies

2.1 Relations

Doctor(Doctor_ID, Doctor_Name, Phone, Address, Birthday, ExperiencesYears, Department)

Patient(Patient_ID, Patient_Name, Phone, Address, Birthday, Gender, AllergiesHistory, Doctor_ID, Illness_ID)

Illness(Illness_ID, Illness_Name, Department, Symptoms, EmergencyLevel)

2.2 Functional Dependencies

2.2.1 Doctor

$Doctor_Name, Phone \rightarrow Doctor_ID, Department, ExperienceYears$

$Doctor_ID \rightarrow Doctor_Name, Address, Phone, Birthday$

2.2.2 Patient

$Patient_Name, Phone \rightarrow Patient_ID, Address, Birthday, AllergiesHistory$
 $Patient_ID \rightarrow Patient_Name, Doctor_ID, Illness_ID, Phone, Gender$

2.2.3 Illness

$Illness_Name \rightarrow Illness_ID, Department, Symptoms$
 $Illness_ID \rightarrow Illness_Name, EmergencyLevel$

3 Decompositions

3.1 Decomposition 1

3.1.1 Decomposing Doctor

$R1$: Doctor Contact Info ($Doctor_Name, Phone, Address, Department$)
 $R2$: Doctor Personal Info ($Doctor_ID, Doctor_Name, Phone, Department, Birthday$)

All other relations remain the same as the original schema

3.1.2 Proof: Lossless Join and Functional Preserving

First condition holds true as

$$\begin{aligned} & Att(R1) \cup Att(R2) \\ &= (Doctor_Name, Phone, Address, Department) \cup (Doctor_ID, Doctor_Name, Phone, Department, Birthday) \\ &= (Doctor_ID, Doctor_Name, Phone, Address, Birthday, Experiences Years, Department) \\ &= Att(R). \end{aligned}$$

Second condition holds true as

$$\begin{aligned} & Att(R1) \cap Att(R2) \\ &= (Doctor_Name, Phone, Address, Department) \cap (Doctor_ID, Doctor_Name, Phone, Department, Birthday) \\ &= (Doctor_Name, Phone) \\ &\neq \phi \end{aligned}$$

Third condition holds true as

$Att(R1) \cap Att(R2) = (Doctor_Name, Phone, Department)$ is a key of $R1$ ($Doctor_Name, Phone, Address, Department$) because of the FD's given

Furthermore, all dependencies of R either can be a part of $R1$ or $R2$ or must be derivable from combination of FDs of $R1$ and $R2$.

3.2 Decomposition 2

3.2.1 Decomposing Patient

R1: Patient Contact Info (*Patient_Name, Phone, Address, Gender*)

R2: Patient Personal Info (*Patient_Name, Phone, Patient_ID, Birthday, AllergiesHistory, Doctor_ID, Illness_ID*)

All other relations remain the same as the original schema

3.2.2 Proof: Lossless Join and Functional Preserving

First condition holds true as

$\text{Att}(R1) \cup \text{Att}(R2)$
= (Patient_Name, Phone, Address, Gender) \cup (Patient_Name, Phone, Patient_ID, Birthday, Allergies History, Doctor_ID, Illness_ID)
= (Patient_ID, Patient_Name, Phone, Address, Birthday, Gender, Allergies History, Doctor_ID, Illness_ID)
= $\text{Att}(R)$.

Second condition holds true as

$\text{Att}(R1) \cap \text{Att}(R2)$
= (Patient_Name, Phone, Address, Gender) \cap (Patient_Name, Phone, Patient_ID, Birthday, Allergies History, Doctor_ID, Illness_ID)
 $\neq \phi$.

Third condition holds true as

$\text{Att}(R1) \cap \text{Att}(R2)$
= (Patient_Name, Phone) is a key of
R1(Patient_Name, Phone, Address, Patient_ID) because of the given FD's

Furthermore, all dependencies of R either can be a part of R1 or R2 or must be derivable from combination of FDs of R1 and R2.

4 SQL DDLs

4.1 Version 1

This version is the original database as defined in section 2.1.

Filename: create_d.ddl

```
1 CREATE TABLE 'Patient' (  
2   'Patient_ID' NUMERIC NOT NULL,  
3   'Patient_Name' TEXT NOT NULL,  
4   'Phone' NUMERIC,  
5   'Address' TEXT,  
6   'Birthday' NUMERIC,
```

```

7  'Gender' ' TEXT,
8  'Allergies_History' ' TEXT,
9  'Doctor_ID' ' NUMERIC NOT NULL,
10 'Illness_ID' ' NUMERIC,
11 PRIMARY KEY('Patient_ID ' )
12 );
13 CREATE TABLE IF NOT EXISTS "Doctor" (
14 'Doctor_ID' ' NUMERIC NOT NULL UNIQUE,
15 'Doctor_Name' ' TEXT NOT NULL,
16 'Phone' ' NUMERIC,
17 'Address' ' TEXT,
18 'Birthday' ' NUMERIC,
19 'Experiences_Year' ' INTEGER,
20 'Department' ' TEXT,
21 PRIMARY KEY('Doctor_ID ' )
22 );
23 CREATE TABLE 'Illness' (
24 'Illness_ID' ' NUMERIC NOT NULL UNIQUE,
25 'Illness_Name' ' TEXT NOT NULL UNIQUE,
26 'Department' ' TEXT,
27 'Symptom' ' TEXT,
28 'Emergency_Level' ' INTEGER,
29 PRIMARY KEY('Illness_ID ' )
30 );

```

4.2 Version 2

This version was created using Decomposition 1 (Section 3.1)

Filename: create_d2.ddl

```

1 CREATE TABLE IF NOT EXISTS "Doctor Personal Info" (
2 'Doctor_ID' ' NUMERIC NOT NULL UNIQUE,
3 'Doctor_Name' ' TEXT NOT NULL,
4 'Phone' ' NUMERIC,
5 'Department' ' TEXT,
6 'Birthday' ' INTEGER,
7 PRIMARY KEY('Doctor_ID ' )
8 );
9 CREATE TABLE IF NOT EXISTS "Doctor Contact Info" (
10 'Doctor_Name' ' TEXT NOT NULL,
11 'Phone' ' NUMERIC,
12 'Address' ' TEXT,
13 'Department' ' TEXT
14 );
15 CREATE TABLE IF NOT EXISTS "Illness" (
16 'Illness_ID' ' NUMERIC NOT NULL UNIQUE,
17 'Illness_Name' ' TEXT NOT NULL,
18 'Department' ' TEXT,
19 'Symptom' ' TEXT,
20 'Emergency_Level' ' INTEGER,
21 PRIMARY KEY('Illness_ID ' )
22 );
23 CREATE TABLE IF NOT EXISTS "Patient" (

```

```

24 'Patient_ID ' NUMERIC NOT NULL UNIQUE,
25 'Patient_Name ' TEXT NOT NULL,
26 'Phone ' NUMERIC,
27 'Address ' TEXT,
28 'Birthday ' NUMERIC,
29 'Gender ' TEXT,
30 'Allergies_History ' TEXT,
31 'Doctor_ID ' NUMERIC,
32 'Illness_ID ' NUMERIC,
33 PRIMARY KEY('Patient_ID ')
34 );

```

4.3 Version 3

This version was created using Decomposition 2 (Section 3.2)

Filename: create_d3.ddl

```

1 CREATE TABLE 'Doctor ' (
2   'Doctor_ID ' NUMERIC NOT NULL UNIQUE,
3   'Doctor_Name ' TEXT NOT NULL,
4   'Phone ' NUMERIC,
5   'Address ' TEXT,
6   'Birthday ' NUMERIC,
7   'Experiences_Years ' INTEGER,
8   'Department ' TEXT,
9   PRIMARY KEY('Doctor_ID ')
10 );
11 CREATE TABLE 'Illness ' (
12   'Illness_ID ' NUMERIC NOT NULL UNIQUE,
13   'Illness_Name ' TEXT NOT NULL,
14   'Department ' TEXT,
15   'Symptom ' TEXT,
16   'Emergency_Level ' INTEGER,
17   PRIMARY KEY('Illness_ID ')
18 );
19 CREATE TABLE IF NOT EXISTS "Patient Personal Info" (
20   'Patient_Name ' TEXT NOT NULL,
21   'Phone ' NUMERIC,
22   'Gender ' TEXT,
23   'Birthday ' NUMERIC,
24   'Allergies_History ' TEXT,
25   'Doctor_ID ' NUMERIC,
26   'Illness_ID ' NUMERIC
27 );
28 CREATE TABLE IF NOT EXISTS "Patient Contact Info" (
29   'Patient_Name ' TEXT NOT NULL,
30   'Phone ' NUMERIC,
31   'Address ' TEXT,
32   'Patient_ID ' INTEGER NOT NULL UNIQUE,
33   PRIMARY KEY('Patient_ID ')
34 );

```


5 Filling The database with data

Java code was used to generate and insert data into d.db. Filename: DataGen.Java
Below are some relevant snippets:

5.1 Data Generation

Below is an example for java code used to generate symptoms for the illness table. It used a text file with a list of generic symptoms and randomly picks out a number of symptoms represented by i. Similar code was used to generate all other fields.

```
1  private static ArrayList<String> GenerateSymptoms (int i) throws IOException
2  {
3      ArrayList<String> Diseases = new ArrayList<>();
4      Random r = new Random(System.currentTimeMillis());
5      FileReader f = new FileReader(new File("etc/sym.txt"));
6      BufferedReader br = new BufferedReader(f);
7      String s;
8      for (int j = 0; j < i; j++) {
9          for (int k = 0; k < r.nextInt(20); k++)
10             br.readLine();
11             s = br.readLine();
12             Diseases.add(s.split("^[^\\s]*\\s")[1]);
13         }
14         br.close();
15     }
16     return Diseases;
```

5.2 Data Insertion

sqlite JDBC for Java was used to insert the generated data, here is the code for inserting data into the Patient table

```
1  public static void insertDoctor (int id, String name, long phone, String add
2  , String birthday, int exp, String dept) {
3      String sql = "INSERT INTO Doctor(Doctor_ID,Doctor_Name,Phone,Address,
4      Birthday,Experiences_Year,Department) VALUES(?,?,?,?,?,?,?)";
5      try (Connection conn = connect(); PreparedStatement pstmt = conn.
6      preparedStatement(sql)) {
7          pstmt.setInt(1, id);
8          pstmt.setString(2, name);
9          pstmt.setLong(3, phone);
10         pstmt.setString(4, add);
11         pstmt.setString(5, birthday);
12         pstmt.setInt(6, exp);
13         pstmt.setString(7, dept);
14         pstmt.executeUpdate();
15     } catch (SQLException e) {
16         System.out.println(e.getMessage());
17     }
18 }
```

6 Copy the data from the first version to the decompositions

6.1 Copy data to Version 2

This queries was using to copy data from version 1. Filename: copy_d_to_d2.sql

```
1 .open d.db
2 ATTACH 'd2.db' as d2;
3
4 INSERT INTO d2.'Doctor Personal Info'
5 SELECT Doctor_ID, Doctor_Name, Phone, Department, Birthday FROM Doctor;
6
7 INSERT INTO d2.'Doctor Contact Info'
8 SELECT Doctor_Name, Phone, Address, Department FROM Doctor;
9
10 INSERT INTO d2.Illness
11 SELECT * FROM Illness;
12
13 INSERT INTO d2.Patient
14 SELECT * FROM Patient;
```

6.2 Copy data to Version 3

This queries was using to copy data from version 1. Filename: copy_d_to_d3.sql

```
1 .open d.db
2 ATTACH 'd3.db' as d2;
3
4 INSERT INTO d2.'Patient Personal Info'
5 SELECT Patient_Name, Phone, Gender, Birthday, Allergies_History, Doctor_ID,
6     Illness_ID FROM Patient;
7
8 INSERT INTO d2.'Patient Contact Info'
9 SELECT Patient_Name, Phone, Address, Patient_ID FROM Patient;
10
11 INSERT INTO d2.Illness
12 SELECT * FROM Illness;
13
14 INSERT INTO d2.Doctor
15 SELECT * FROM Doctor;
```

7 English Queries

7.1 First English Query

Give Doctor_Name = 'Taren Batarse', find all his patients Patient_ID and Patient_Name.
(Relate to Doctor, Patient table)

7.2 Second English Query

Give Patient_Name is 'Erika Heuberger' and Phone is '2449717107', find Doctor_Name, and her symptom. (Relate to Doctor, Illness, Patient table)

7.3 Third English Query

Give Illness_ID = '1', find all patient has this kind of illness Patient_Name and their Doctor_Name. (Relate to Doctor, Illness, Patient table)

8 SQL implementations of the queries

8.1 First English Query

Give Doctor_Name = 'Taren Batarse', find all his patients Patient_ID and Patient_Name. (Relate to Doctor, Patient table)

8.1.1 Version 1

Filename: query_1_db1.sql

```
1 select p.Patient_ID , p.Patient_Name
2 from 'Patient' as p
3 join 'Doctor' as d on d.Doctor_ID = p.Doctor_ID
4 where d.Doctor_Name = 'Taren Batarse';
```

8.1.2 Version 2

Filename: query_1_db2.sql

```
1 select p.Patient_ID , p.Patient_Name
2 from Patient as p
3 join 'Doctor Personal Info' as d on d.Doctor_ID = p.Doctor_ID
4 where d.Doctor_Name = 'Taren Batarse';
```

8.1.3 Version 3

Filename: query_1_db3.sql

```
1 select p.Patient_ID , p.Patient_Name
2 from 'Patient Personal Info' as pp
3 join Doctor as d on d.Doctor_ID = pp.Doctor_ID
4 join 'Patient Contact info' as p on pp.Patient_Name = p.Patient_Name
5 where d.Doctor_Name='Taren Batarse';
```

8.2 Second English Query

Give Patient_Name is 'Erika Heuberger' and Phone is '2449717107', find Doctor_Name, and her symptom. (Relate to Doctor, Illness, Patient table)

8.2.1 Version 1

Filename: query_2_db1.sql

```
1 select d.Doctor_Name, i.symptom
2 from Patient as p
3 join Doctor as d on d.Doctor_ID = p.Doctor_ID
4 join Illness as i on i.Illness_ID = p.Illness_ID
5 where p.Patient_Name = 'Erika Heuberger' and p.Phone = '2449717107'
```

8.2.2 Version 2

Filename: query_2_db2.sql

```
1 select d.Doctor_Name , i.symptom
2 from Patient as p
3 join 'Doctor Personal Info' as d on d.Doctor_ID = p.Doctor_ID
4 join Illness as i on i.Illness_ID = p.Illness_ID
5 where p.Patient_Name = 'Erika Heuberger' and p.Phone = '2449717107'
```

8.2.3 Version 3

Filename: query_2_db3.sql

```
1 select d.Doctor_Name , i.Illness_ID
2 from 'Patient Contact Info' as p
3 join 'Patient Personal Info' as pp on p.Phone = pp.Phone
4 join Doctor as d on pp.Doctor_ID = d.Doctor_ID
5 join Illness as i on pp.Illness_ID = i.Illness_ID
6 where p.Patient_Name = 'Erika Heuberger' and p.Phone = '2449717107';
```

8.3 Third English Query

Give Illness_ID = '1', find all patient has this kind of illness Patient_Name and their Doctor_Name. (Relate to Doctor, Illness, Patient table)

8.3.1 Version 1

Filename: query_3_db1.sql

```
1 select p.Patient_Name, d.Doctor_Name
2 from Patient as p
3 join Doctor as d on d.Doctor_ID = p.Doctor_ID
4 join Illness as i on i.Illness_ID = p.Illness_ID
5 where p.Illness_ID = '1';
```

8.3.2 Version 2

Filename: query_3_db2.sql

```

1 select p.Patient_Name , d.Doctor_Name
2 from Patient as p
3 join 'Doctor Personal Info' as d on d.Doctor_ID = p.Doctor_ID
4 join Illness as i on i.Illness_ID = p.Illness_ID
5 where i.Illness_ID = '1';

```

8.3.3 Version 3

Filename: query_3_db3.sql

```

1 select p.Patient_Name , d.Doctor_Name
2 from 'Patient Personal Info' as pp
3 join 'Patient Contact Info' as p on p.Phone = pp.Phone
4 join Doctor as d on pp.Doctor_ID = d.Doctor_ID
5 join Illness as i on i.Illness_ID = pp.Illness_ID
6 where i.Illness_ID = '1';

```

9 Test the time

Filename: results.csv

10 References

1. <http://www.texample.net/tikz/examples/entity-relationship-diagram/>
2. <http://www.mockaroo.com> - used this to generate random data
3. <https://www.cdc.gov/datastatistics/index.html>