CSE 4020 - Database Systems Project

Lingjing Huang, Shreyas Ugemuge

December 1, 2017

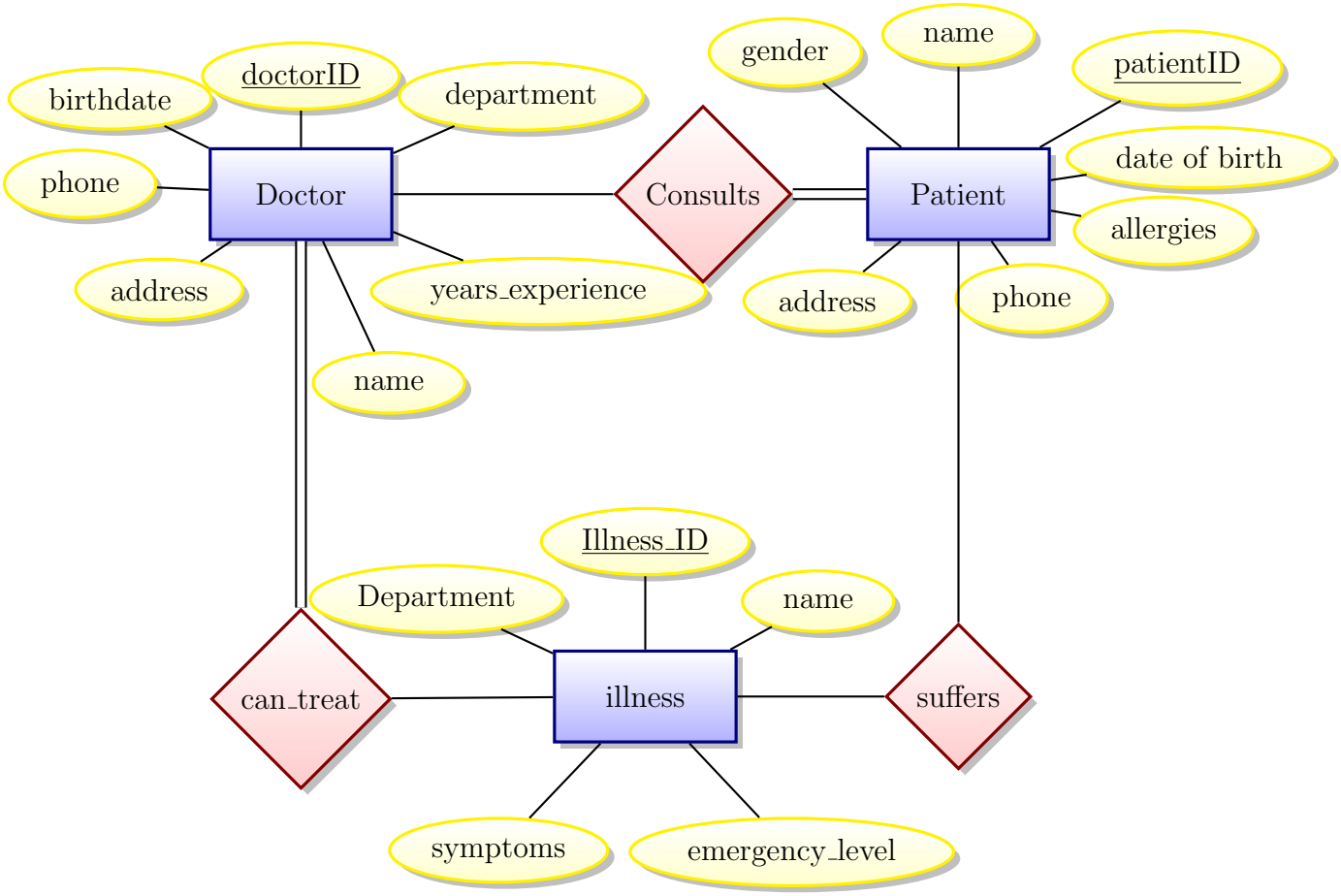
Contents

Problem Statement

Design a database for storing information about medical doctors, their patients, and ilnesses (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 columns 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

$Doctor(\underline{Doctor\_ID}, Doctor\_Name, Phone, Address, Birthday, ExperiencesYears, Department)$
$Patient(\underline{Patient\_ID}, Patient\_Name, Phone, Address, Birthday,$
$Gender, AllergiesHistory, Doctor\_ID, Illness\_ID)$
$Illness(\underline{Illness\_ID}, Illness\_Name, Department, Symptoms, EmergencyLevel)$

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
Att(R1) ∪ Att(R2)
= (Doctor_Name, Phone, Address, Department) ∪ (Doctor_ID, Doctor_Name, Phone, Department, Birthday)
= (Doctor_ID, Doctor_Name, Phone, Address, Birthday, Experiences Years, Department)
= Att(R).

Second condition holds true as
Att(R1) ∩ Att(R2)
= (Doctor_Name, Phone, Address, Department) ∩ (Doctor_ID, Doctor_Name, Phone, Department, Birthday)
= (Doctor_Name, Phone)
$\neq \phi$

Third condition holds true as
Att(R1) ∩ 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
Att(R1) ∪ Att(R2)
= (Patient_Name, Phone, Address, Gender) ∪ (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)
= Att(R).

Second condition holds true as
Att(R1) ∩Att(R2)
= (Patient_Name, Phone, Address,Gender) ∩ (Patient_Name, Phone, Patient_ID, Birthday, Allergies History, Doctor_ID, Illness_ID)
$\neq \phi$.

Third condition holds true as
Att(R1) ∩ 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,
```

```
    `Patient_Name`  TEXT NOT NULL,
    `Phone` NUMERIC,
    `Address` TEXT,
    `Birthday`  NUMERIC,
    `Gender`  TEXT,
    `Allergies_History` TEXT,
    `Doctor_ID` NUMERIC NOT NULL,
    `Illness_ID`  NUMERIC,
    PRIMARY KEY(`Patient_ID`)
);
CREATE TABLE IF NOT EXISTS "Doctor" (
    `Doctor_ID` NUMERIC NOT NULL UNIQUE,
    `Doctor_Name` TEXT NOT NULL,
    `Phone` NUMERIC,
    `Address` TEXT,
    `Birthday`  NUMERIC,
    `Experiences_Year`  INTEGER,
    `Department`  TEXT,
    PRIMARY KEY(`Doctor_ID`)
);
CREATE TABLE `Illness` (
    `Illness_ID`  NUMERIC NOT NULL UNIQUE,
    `Illness_Name`  TEXT NOT NULL UNIQUE,
    `Department`  TEXT,
    `Symptoms`  TEXT,
    `Emergency_Level` INTEGER,
    PRIMARY KEY(`Illness_ID`)
);
```

4.2 Version 2

This version was created using Decomposition 1 (Section 3.1)
Filename: `create_d2.ddl`

```
CREATE TABLE IF NOT EXISTS "Doctor Personal Info" (
    `Doctor_ID` NUMERIC NOT NULL UNIQUE,
    `Doctor_Name` TEXT NOT NULL,
    `Phone` NUMERIC,
    `Department`  TEXT,
    `Birthday`  INTEGER,
    PRIMARY KEY(`Doctor_ID`)
);
CREATE TABLE IF NOT EXISTS "Doctor Contact Info" (
    `Doctor_Name` TEXT NOT NULL,
    `Phone` NUMERIC,
    `Address` TEXT,
    `Department`  TEXT
);
CREATE TABLE IF NOT EXISTS "Illness" (
    `Illness_ID`  NUMERIC NOT NULL UNIQUE,
    `Illness_Name`  TEXT NOT NULL,
    `Department`  TEXT,
    `Symptom` TEXT,
```

```
  `Emergency_Level` INTEGER,
  PRIMARY KEY(`Illness_ID`)
);
CREATE TABLE IF NOT EXISTS "Patient" (
  `Patient_ID`  NUMERIC NOT NULL UNIQUE,
  `Patient_Name`  TEXT NOT NULL,
  `Phone` NUMERIC,
  `Address` TEXT,
  `Birthday`  NUMERIC,
  `Gender`  TEXT,
  `Allergies_History` TEXT,
  `Doctor_ID` NUMERIC,
  `Illness_ID`  NUMERIC,
  PRIMARY KEY(`Patient_ID`)
);
```

4.3 Version 3

This version was created using Decomposition 2 (Section 3.2)
Filename: `create_d3.ddl`

```
CREATE TABLE `Doctor` (
  `Doctor_ID` NUMERIC NOT NULL UNIQUE,
  `Doctor_Name` TEXT NOT NULL,
  `Phone` NUMERIC,
  `Address` TEXT,
  `Birthday`  NUMERIC,
  `Experiences_Years` INTEGER,
  `Department`  TEXT,
  PRIMARY KEY(`Doctor_ID`)
);
CREATE TABLE `Illness` (
  `Illness_ID`  NUMERIC NOT NULL UNIQUE,
  `Illness_Name`  TEXT NOT NULL,
  `Department`  TEXT,
  `Symptom` TEXT,
  `Emergency_Level` INTEGER,
  PRIMARY KEY(`Illness_ID`)
);
CREATE TABLE IF NOT EXISTS "Patient Personal Info" (
  `Patient_Name`  TEXT NOT NULL,
  `Phone` NUMERIC,
  `Gender`  TEXT,
  `Birthday`  NUMERIC,
  `Allergies_History` TEXT,
  `Doctor_ID` NUMERIC,
  `Illness_ID`  NUMERIC
);
CREATE TABLE IF NOT EXISTS "Patient Contact Info" (
  `Patient_Name`  TEXT NOT NULL,
  `Phone` NUMERIC,
  `Address` TEXT,
  `Patient_ID`  INTEGER NOT NULL UNIQUE,
```

```
  PRIMARY KEY(`Patient_ID`)
);
```

5 The database with data

6 Copy the data from the first version

6.1 Copy data to Version 2

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

```
.open d.db
ATTACH 'd2.db' as d2;

INSERT INTO d2.'Doctor Personal Info'
SELECT Doctor_ID, Doctor_Name, Phone, Department, Birthday FROM Doctor;

INSERT INTO d2.'Doctor Contact Info'
SELECT Doctor_Name, Phone, Address, Department FROM Doctor;

INSERT INTO d2.Illness
SELECT * FROM Illness;

INSERT INTO d2.Patient
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`

```
.open d.db
ATTACH 'd3.db' as d2;

INSERT INTO d2.'Patient Personal Info'
SELECT Patient_Name, Phone, Gender, Birthday,Allergies_History,Doctor_ID,
    Illness_ID FROM Patient;

INSERT INTO d2.'Patient Contact Info'
SELECT Patient_Name, Phone, Address, Patient_ID FROM Patient;

INSERT INTO d2.Illness
SELECT * FROM Illness;

INSERT INTO d2.Doctor
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`

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

8.1.2 Version 2

Filename: `query_1_db2.sql`

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

8.1.3 Version 3

Filename: `query_1_db3.sql`

```sql
select p.Patient_ID, p.Patient_Name
from 'Patient Personal Info' as pp
join Doctor as d on d.Doctor_ID = pp.Doctor_ID
join 'Patient Contact info' as p on pp.Patient_Name = p.Patient_Name
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`

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

8.2.2 Version 2

Filename: `query_2_db2.sql`

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

8.2.3 Version 3

Filename: `query_2_db3.sql`

```sql
select d.Doctor_Name , i.Illness_ID
from `Patient Contact Info` as p
join `Patient Personal Info` as pp on p.Phone = pp.Phone
join Doctor as d on pp.Doctor_ID = d.Doctor_ID
join Illness as i on pp.Illness_ID = i.Illness_ID
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`

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

8.3.2 Version 2

Filename: `query_3_db2.sql`

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

8.3.3 Version 3

Filename: `query_3_db3.sql`

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