CSE 4020 - Database Systems Project

Lingjing Huang, Shreyas Ugemuge

November 30, 2017

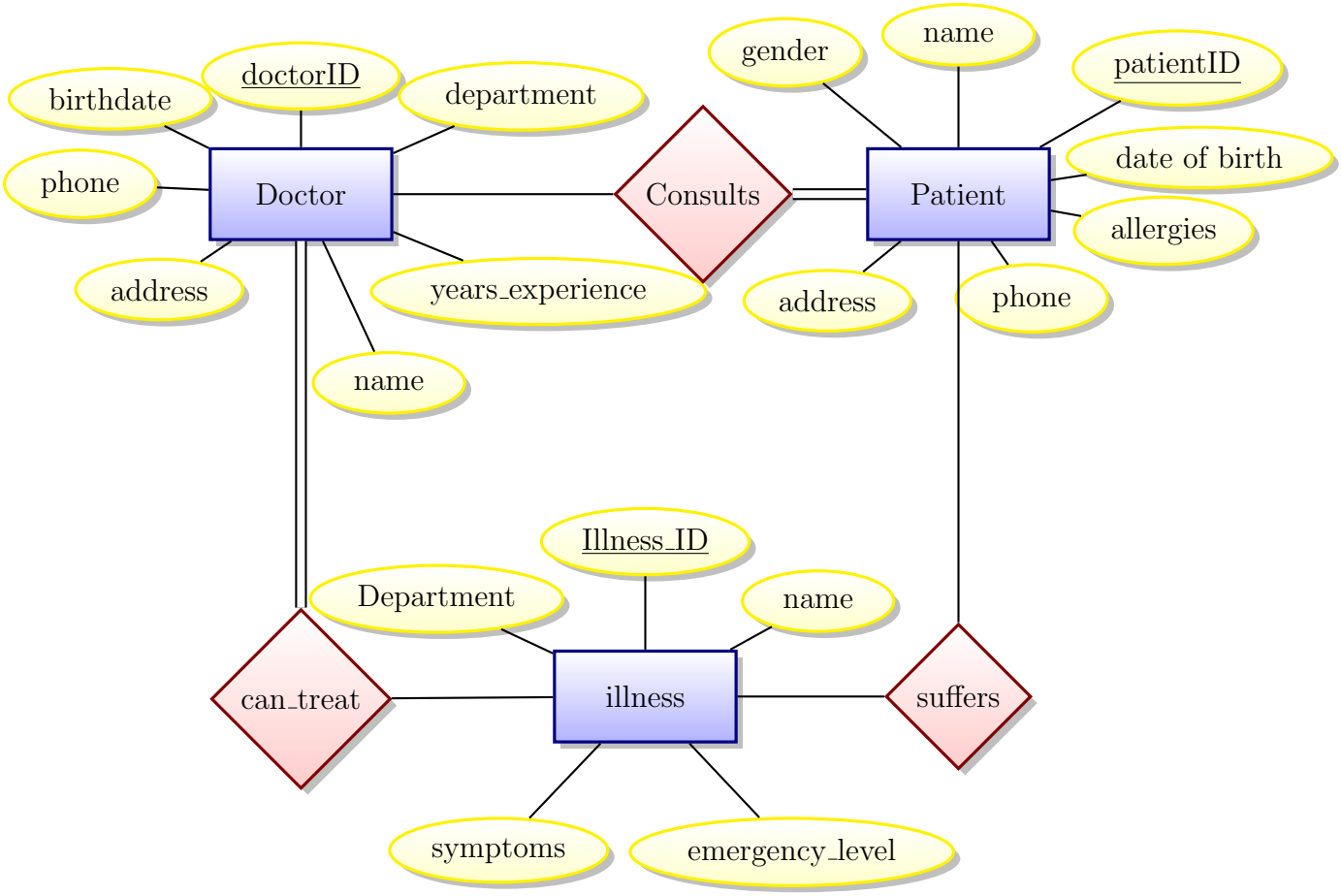
Contents

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

$Doctor\_Name, Phone \rightarrow Address, Department,$
$Doctor\_Name, Birthday \rightarrow Phone, Department$

2.2.2 Patient

$Patient\_Name, Phone \rightarrow Address, Birthday,$
$Birthday, Phone \rightarrow Gender$

2.2.3 Illness

$Illness\_Name \rightarrow Department, Symptoms$

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) is a key of R1 (Doctor_Name, Phone, Address, Department) because
Doctor_Name, Phone → Address, Department is 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
Patient_Name, Phone → Address, Birthday and
Birthday, Phone → Gender are 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.

4 SQL DDLs

4.1 Version 1

This version is the original database as defined in section 2.1.
Filename: `create_d.ddl`

```
  CREATE TABLE `Patient` (
   `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`)
);
```