CSC 453 Database Technologies 701/710

Assignment 3 (9/21)

Due 11:59:00pm, Monday 10/2.

Problems:

1. Consider the following relational database schema:

> TYPE(<u>TypeName</u>, Weight, NumberOfGuns)
> SHIP(<u>ShipName</u>, Country, ShipType, DateLaunched)
> BATTLE(<u>BattleName</u>, BattleDate)
> OUTCOME(<u>BattleName</u>, <u>ShipName</u>, Result)

ShipType in SHIP is a foreign key referencing the TypeName attribute of TYPE. BattleName and ShipName in OUTCOME are foreign keys referencing the attributes with the same names in BATTLE and SHIP.

The TYPE relation records information about different types of ships: the name describing the type (e.g., Carrier), the weight of that type of ship, and the number of guns on that type of ship. The SHIP relation records information about particular ships: the ship name, its country and type, and the date it was launched. The BATTLE relation records information on battles fought at sea: a unique name for each battle, and its date. The OUTCOME relation records information on the effects of battles on the ships that participated in them: the name of a battle, the name of a ship that fought in it, and the result of the battle for that ship (e.g., Undamaged, Damaged, Disabled, Sunk).

Write SQL queries for this schema to do the following:

a. Give the names of the countries that have a ship with the smallest weight among all ships.

SELECT Country from (Type T INNER JOIN SHIP S ON T.TypeName = S.ShipType) Where Weight = (SELECT Min(Weight) FROM TYPE);

b. Give the types of ships for which no ship of that type has ever been sunk in a battle.

SELECT ShipType from (SHIP S INNER JOIN OUTCOME O ON S.ShipName = O.ShipName) Where Result != 'Sunk';

c. Give the names of all ships (not their types, but the names of the individual ships) that have at least six guns.

SELECT ShipName FROM (TYPE T INNER JOIN SHIP S ON T.TypeName = S.ShipType) WHERE NumberOfGuns >= 6;

d. Give the names of all battles in which a ship of type Destroyer participated.

SELECT BattleName FROM (SHIP S INNER JOIN OUTCOME O ON S.ShipName = O.ShipName) WHERE ShipType = 'Destroyer';

2. Consider a relation *R* with schema *R*(A, B, C, D), and the following three functional dependencies: B $\rightarrow$ A ; C $\rightarrow$ B ; A,D $\rightarrow$ C .

a. For every non-empty subset X of the set {A, B, C, D} of attributes, find the closure of X under the set of three functional dependencies given above. (Yes, there are 15 such subsets to consider, but most of the closures do not take long to compute. If you are unsure why there are 15 possible non-empty subsets of the set {A, B, C, D}, google "power set"…)

$\{A\}^+ \rightarrow \{A\}$ only derives itself
$\{B\}^+ \rightarrow \{B, A\}$
$\{C\}^+ \rightarrow \{C, B, A\}$
$\{D\}^+ \rightarrow$ cannot derive anything on it's own
$\{A,B\}^+ \rightarrow \{A, B, C\}$
$\{A,C\}^+ \rightarrow \{A, C, B\}$
$\{A,D\}^+ \rightarrow \{A, D, C, B\}$
$\{B,C\}^+ \rightarrow \{B, C, A\}$
$\{B,D\}^+ \rightarrow \{B, D, A, C\}$
$\{C,D\}^+ \rightarrow \{C, D, B, A\}$
$\{A,B,D\}^+ \rightarrow \{A, B, D, C\}$
$\{A,B,C\}^+ \rightarrow \{A, B, C\}$
$\{A,C,D\}^+ \rightarrow \{A, B, D, C\}$
$\{B,C,D\}^+ \rightarrow \{A, B, D, C\}$
$\{A,B,C,D\}^+ \rightarrow \{A, B, D, C\}$

b. List all superkeys of the relation *R*. (can derive a full, unique set)

AD, BD, CD, BCD, ACD, ABD, ABCD

$\{A,D\}^+ \rightarrow \{A, D, C, B\}$
$\{B,D\}^+ \rightarrow \{B, D, A, C\}$
$\{C,D\}^+ \rightarrow \{C, D, B, A\}$
$\{A,B,D\}^+ \rightarrow \{A, B, D, C\}$

$\{A,C,D\}^+ \rightarrow \{A, B, D, C\}$
$\{B,C,D\}^+ \rightarrow \{A, B, D, C\}$
$\{A,B,C,D\}^+ \rightarrow \{A, B, D, C\}$

c. List all candidate keys of the relation *R*. (smallest super key that can derive a full, unique set)

AD, BD, CD

$\{A,D\}^+ \rightarrow \{A, D, C, B\}$
$\{B,D\}^+ \rightarrow \{B, D, A, C\}$
$\{C,D\}^+ \rightarrow \{C, D, B, A\}$

(As an aside, it turns out that no matter which candidate key we choose to be the primary key, the simple normalization process that we discussed this week will not apply, since there are multiple candidates. In this case we must consider more general definition of the normal forms that we will discuss next week.)

3. Consider the following relational schema:

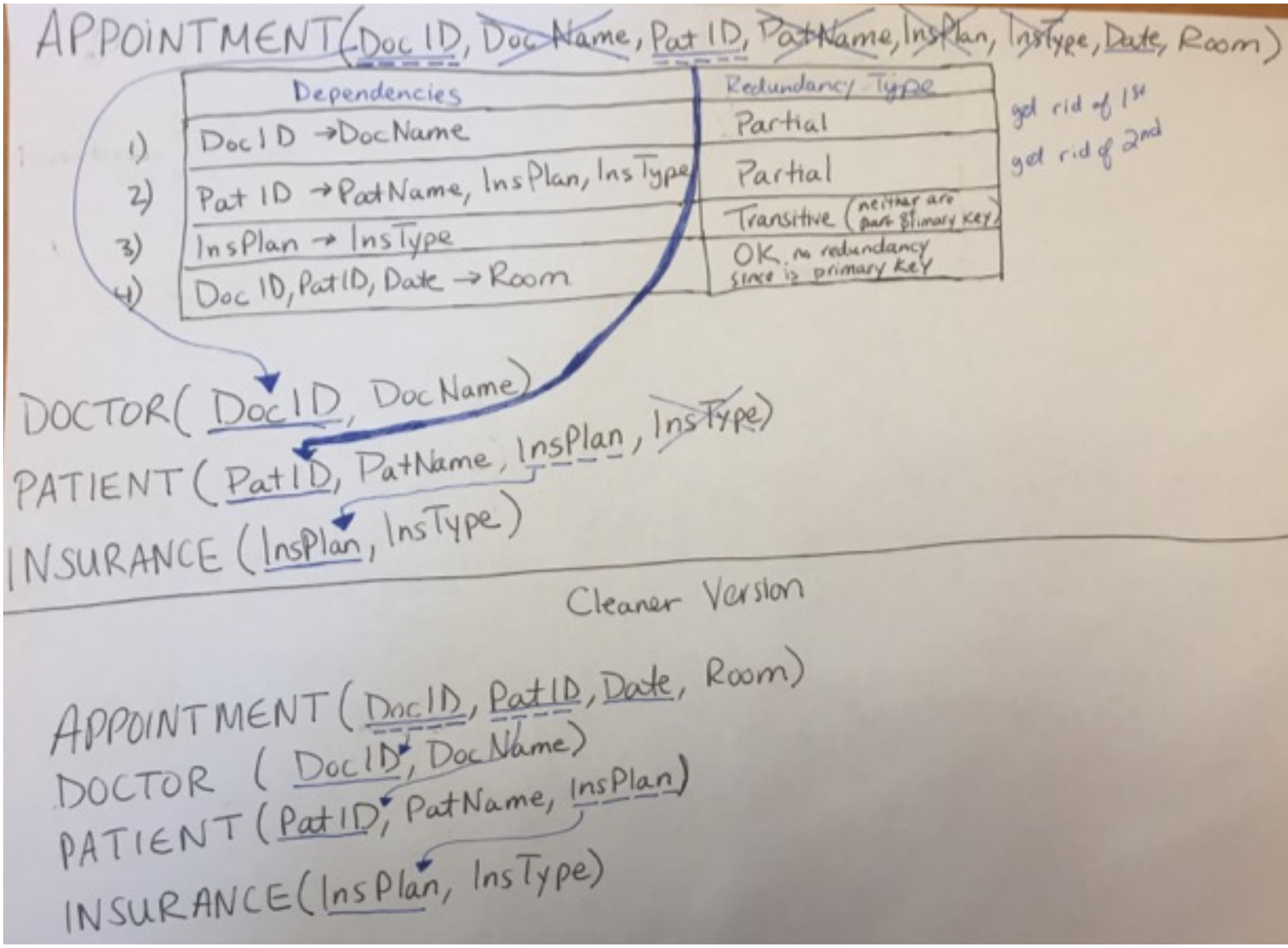
APPOINTMENT(DocID, DocName, PatID, PatName, InsPlan, PlanType, Date, Room)

Each tuple in a relation state of APPOINTMENT represents a patient's appointment with a doctor on a particular date. Each doctor has a unique ID, as does each patient, but doctors' names and patients' names might not be unique. Each patient has just one insurance plan, but many patients may have the same insurance plan; every insurance plan has a single type. When a patient makes an appointment with a doctor for a particular date, that appointment is assigned to some examination room. A patient can make appointments with different doctors for the same date, and can make appointments for different dates with the same doctor, but if a patient has made an appointment with a doctor for a particular date, the patient cannot make another appointment with the same doctor for that same date.

Thus the functional dependencies in APPOINTMENT are

DocID $\rightarrow$ DocName
PatID $\rightarrow$ PatName, InsPlan, InsType
InsPlan $\rightarrow$ InsType
DocID, PatID, Date $\rightarrow$ Room

<u>Scratch work:</u>
$\{$DocID$\}^+ \rightarrow \{$DocID, DocName$\}$
$\{$DocName$\}^+ \rightarrow$ derives nothing on its own
$\{$PatID$\}^+ \rightarrow \{$PatName, InsPlan, InsType$\}$
$\{$PatName$\}^+ \rightarrow$ derives nothing on its own



APPOINTMENT(DocID, ~~DocName~~, PatID, ~~PatName~~, ~~InsPlan~~, ~~InsType~~, Date, Room)

	Dependencies	Redundancy Type	
1)	DocID → DocName	Partial	get rid of 1st
2)	PatID → PatName, InsPlan, InsType	Partial	get rid of 2nd
3)	InsPlan → InsType	Transitive (neither are part of primary key)	
4)	DocID, PatID, Date → Room	OK, no redundancy since is primary key	

DOCTOR(DocID, DocName)
PATIENT(PatID, PatName, InsPlan, ~~InsType~~)
INSURANCE(InsPlan, InsType)

Cleaner Version

APPOINTMENT(DocID, PatID, Date, Room)
DOCTOR (DocID, DocName)
PATIENT(PatID, PatName, InsPlan)
INSURANCE(InsPlan, InsType)

{ InsPlan}$^+$ → {InsType}
{ InsType}$^+$ → derives nothing on its own
{Date} $^+$ → derives nothing on its own
{Room}$^+$ → derives nothing on its own
{DocID, PatID, Date}$^+$ → {Room, DocName, PatName, InsPlan, InsType} (only way to derive all values)

a. APPOINTMENT has only one candidate key. What is it?

{DocID, PatID, Date}

b. The APPOINTMENT relation is in first normal form. Transform APPOINTMENT into a set of linked relational schemas in third normal form by removing all partial and transitive dependencies on the primary key of any relation. (You can describe the relational schemas as I did in Problem 1 – **underline primary keys and explain foreign keys verbally** – if you don't want to draw a picture.)

See next page:

primary keys	**foreign keys**
DocID, PatID, Date in APPOINTMENT	DocID (referencing DocID in DOCTOR) and PatID (referencing PatID in PATIENT) in APPOINTMENT
DocID in DOCTOR	
PatID in PATIENT	InsPlan (referencing InsPlan in INSURANCE) in PATIENT
InsPlan in INSURANCE	