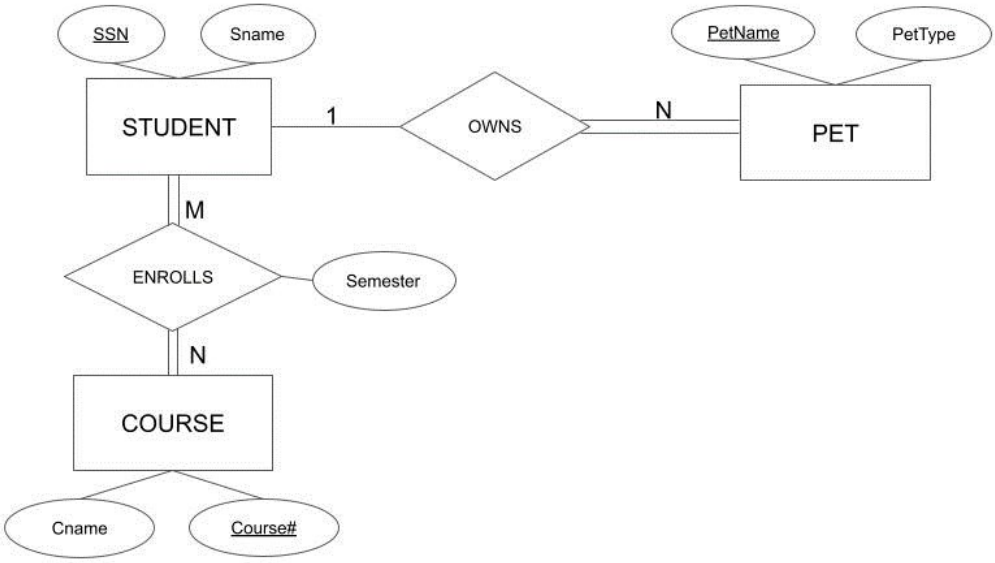
TUTORIAL for Lecture Week 11 (QUESTIONS AND ANSWERS)

NOMALIZATION in Relational Database Design

(Prof. Sham Navathe- Lecture on Oct 31, 2023)

Q1. CONSIDER THE FOLOWING ER SCHEMA about students, courses they enroll in and the pets they have.



A. Consider a table created from this schema :
STUDENT_PET (Student_Ssn, Sname, Petname, Pettype)
Suppose 70% students own a pet and we are tolerating 30% nulls in the last two columns for students that do not own a pet.

i) What are the functional dependencies in this table?
ii) Is this a good design? What is/are the keys of this table?.
iii) Can Petname be considered a candidate key ?
iv) What normal form is this table in? Do you approve of this design?
v) Consider decomposing it into
STUDENT1 (Student_Ssn, Sname)
PET (Petname, Pettype, Student_SSN)
What FDs are present? Does this design meet BCNF? Is this a better design?

SOLUTION:

Q1 A.

(i) FDs: Student_ssn → Sname; Petname → Pettype

(ii) The relation is poorly designed and has no keys. Student_ssn repeats if a student has multiple Pets. For some students, the Petname is Null. Hence (Student_ssn, Petname) cannot be considered a key.

(iii) Petname cannot be considered a candidate key either because it has a null value for 30% of students. Relational model disallows a key to have null value.
NOTE: Postgres will allow the statement UNIQUE (Petname) because whatever non-null values of Petname exist, they will be unique. However, that still does NOT make it a candidate key.

(iv) The table is in a bad state. It is not even in 1NF.

(v) The new design with STUDENT1 and PET has FDs: Student_ssn → Sname and Petname → Pettype, Student_ssn. It meets the requirements of BCNF and is a better design.

B. Suppose the ENROLLS relationship type was mapped into the following table with primary key (SSN, Course#)

ENROLLS (SSN, Course#, Student_name, Cname, Semester)

i) What are the FDs in this table?

ii) What normal form is it in? Why?

iii) Do successive decomposition of this table.

iv) Test and justify that the result is in BCNF

v) Now suppose we introduce an FD Semester → Course#. Does the design remain in BCNF? If not, convert it into BCNF.

SOLUTION

Q1B.

(i) ENROLLS has the following FDs: SSN → Student_name; Course# → Cname; (SSN, Course#) → Semester

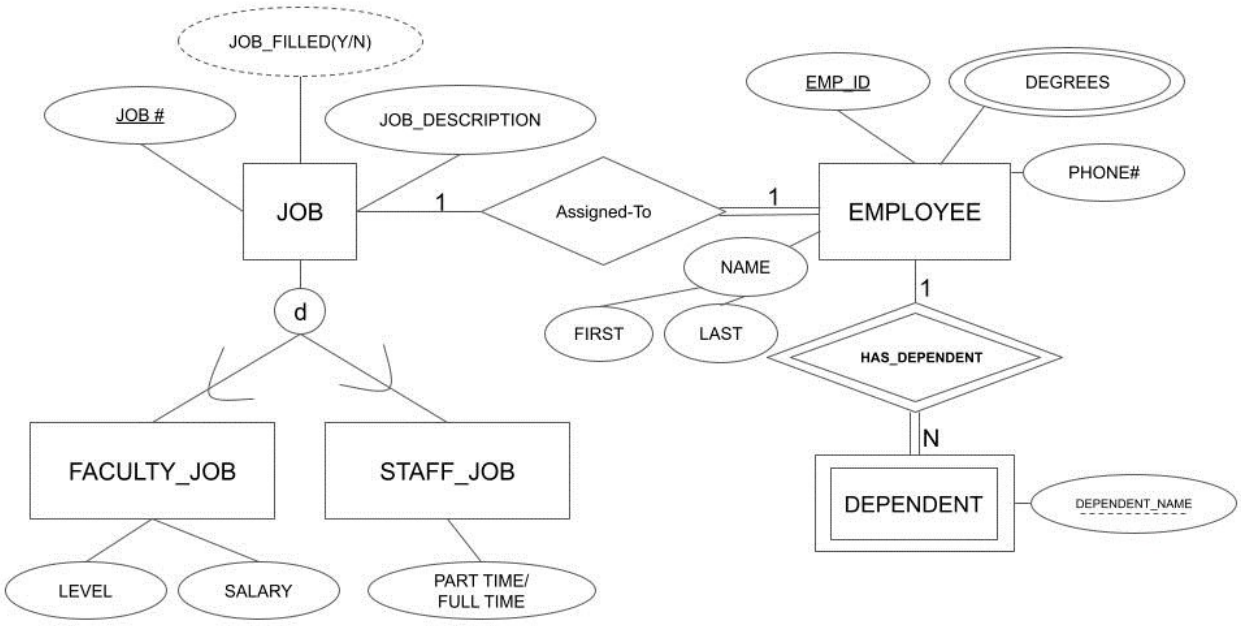
(ii) It is in 1NF. Attributes Student_name and Cname are NOT fully functionally dependent on the entire key: SSN, Course#. Hence, 2NF is violated.

(iii) Second Normalization:
ENROLLS1 (SSN, Course#, Semester)
STUDENT (SSN , Student_name)
COURSE (Course# , Cname)

(iv) Yes, the result is in BCNF because in each relation the LHS of every FD is a superkey.

(v) With the new FD Semester → Course#, ENROLLS1 fails BCNF because the LHS Semester is not a superkey. It needs to be decomposed.
Using the result from Algorithm 15.5, if X → Y violates BCNF in a relation R, we replace it by two relations: R1 (R-Y) and R2 (XUY). By applying this, we convert ENROLLS into:
ENROLLS1 (SSN, Semester)
ENROLLS2 (Semester, Course#). This decomposition meets the NJB property and is the correct decomposition.

Q2. Consider the EER schema below about Employees and Dependents and answer the following questions:



2A. Suppose the designer designed the following table for Employees:

EMPLOYEE (Empid, Phone#, Degrees)

i) What are the FDs in this table?

ii) What normal form is it in?

iii) Show two possible first normalizations of this table – one that may have redundancy and one that will avoid any redundancy. Show the FDs in each case.

SOLUTION:

Q2A.

(i) FDs in the table EMPLOYEE: Empid → Phone#.

(ii) EMPLOYEE is not even in 1 NF

(iii) First Normalizations of table EMPLOYEE:

ONE: EMP_REPEAT (Empid, Degree, Phone#). FDs present are:

Empid, Degree → Phone#; Empid → Phone#.

This design redundantly stores Empid when there are multiple degrees for that employee.

TWO: EMP_PHONE (Empid, Phone#), **EMP_DEGREE** (Empid, Degree)

The table EMP_PHONE has the FD: Empid → Phone#.

The table EMP_DEGREE is an all key table and has NO FDs.

This is the preferred design with no redundancy.

2B. Suppose the designer proposed the design of the Employee and Dependent information as a nested relation:

EMP_NESTED (Empid, Phone#, (Degree), (Dependent_name))

i) What normal form is this design in?

ii) Suppose the following table EMP_FLAT was presented to you as a relational design:

Empid	Phone#	Degree	Dependent_name
101	6146	BS	Peter
101	6146	MS	Mary
102	6234	Mtech	Jill
102	6234	PhD	Janet
102	6234	<null>	John

What are the problems with this design?

iii) How will you convert this to a proper design?

iv) Justify that your final design meets BCNF.

SOLUTION:

Q2B.

(i) EMP_NESTED is not even in 1NF. The key Emp# does NOT functionally determine the nested multi-valued attributes.

(ii) The table EMP_FLAT has the problem that it has mixed two multi-valued attributes in the same table creating a wrong association between the attributes Degree and Dependent. The pairing <BS, Peter> and <MS, Mary> creates a non-existent relationship between degree BS and dependent Peter and degree MS and dependent Mary for Employee 101. It creates FDs
Empid, Degree → Dependent_name and
Empid, Dependent_name → Degree
Both of which are meaningless.

(iii) The correct design is to decompose into 3 tables:
EMP1 (Empid, Phone)
EMP2 (Empid, Degree) and
EMP3 (Empid, Dependent_name)
The above design has only one FD: Empid → Phone#. EMP2 and EMP3 have NO FD.
This design meets BCNF.

Q3. SUCCESSIVE NORMALIZATION

Consider the following relation:

REFRIG_MODELS (Model#, Option_type, Option_Listtprice, Model_color, Discount%, Quantity, Sticker_price)

It refers to an inventory of Refrigerators at an appliance store. It includes options available on refrigerators (e.g. – ice maker, soda-water maker) that are sold and the list-prices and discounted prices etc.

The known f.d.s:

FD1:Model# → Model_color;

FD2:Option_type→ Option_Listprice;

FD3:Model#, Option_type → Quantity, Sticker_price;

FD4: Model_color → Discount%

Argue **using the generalized definition of the BCNF** that this relation is not in BCNF; further argue why it is not even in 3NF by the generalized definition of 3NF. Finally point out which dependency (or dependencies) violate the 2NF. Carry out successive normalization until you reach a BCNF design.

SOLUTION:

Q3. Successive Normalization

The key for this relation is (Model#,Option_type).
The Fds Fd1, FD2, FD4 all have LHS that is NOT a superkey.
Hence this relation is not in BCNF.
Applying the generalized definition of 3NF also the above 3 FDs violate the 3NF definition; hence it is not in 3NF.
The Fds FD1 and FD2 violate the 2NF because Model_color and Option_Listprice are not fully functionally dependent on the key (Model#,Option_type).
Hence the successive normalization proceeds as follows:
2nd Normalization:
MODEL (Model#, Model_color, Discount%)
OPTION (Option_type, Option_listprice)
REFRIG1(Model#, Option_type, Quantity, Sticker_price)
Now, the MODEL relation has a transitive dependency of Discount% via Model_color).
Hence it undergoes third normalization:
MODEL1 (Model#, Model_color)
MODEL2 (Model_color, Discount%)
The final design includes the tables: REFRIG1, OPTION, MODEL1, MODEL2.
All of them meet the BCNF property and hence the design is in BCNF.

Q4. NON-ADDITIVE BINARY DECOMPOSITION

Consider the following set of functional dependencies F in the STUDENT relation (it has obvious meaning of data related to students who enrolled in courses and received a grade):

STUDENT (**Stud_SSN, Course_no, Grade, Course_name, Stud_name, Course_Instr)**

F: { FD1: (Stud_SSN, Course_no) → Grade, Course_name, Stud_name, Course_Instr;

FD2: Stud_SSN → Stud_name;

FD3: Course_no → Course_name, Course_Instr }

The designer proposed to decompose this relation into

STUDENT1 (Stud_SSN, Course_no, Stud_name, Grade) and

COURSE (Course_no, Course_name, Course_Instr)

i) Apply the Non-additive join (NJB) test to determine if this decomposition is lossless
ii) What are the highest normal forms the above relations are in?
iii) Normalize the design further if possible.

SOLUTION:

Q4. Non-additive Join Binary Decomposition

- **PROPERTY NJB (non-additive join test for binary decompositions):**
  A decomposition D = {R1, R2} of R has the lossless join property with respect to a set of functional dependencies F on R *if and only if* either
  - The f.d. ((R1 ∩ R2) → (R1- R2)) is in F^+, or
  - The f.d. ((R1 ∩ R2) → (R2 - R1)) is in F

(i) Applying the NJB property to the given two relations:
STUDENT1 ∩ COURSE = { Course#}
COURSE – STUDENT1 = { Course_name, Course_Instr}.
Since { Course#} → { Course_name, Course_Instr}.
The given decomposition is lossless (non-additive).
(ii) STUDENT1 is in 1NF because Student_ssn → Student_name is a 2NF violation.
and COURSE is in BCNF.
(iii) STUDENT1 should be decomposed as : STUDENT2 (Student_ssn ,Student_name)
and ENROLL ((Stud_SSN, Course_no, Grade)
Hence the final decomposition of the given relation STUDENT as a BCNF design is the following set of relations:

COURSE (Course_no, Course_name, Course_Instr)

STUDENT2 (Student_ssn ,Student_name)

ENROLL ((Stud_SSN, Course_no, Grade)

++++++++++++++++++ +++++++++++++++ **END** ++++++++++++++++++++++++++++++++++