TUTORIAL for Lecture Week 10

INFORMAL DESIGN GUIDELINES AND FUNCTIONAL DEPENDENCIES

(Prof. Sham Navathe- Lecture on Oct 24, 2023)

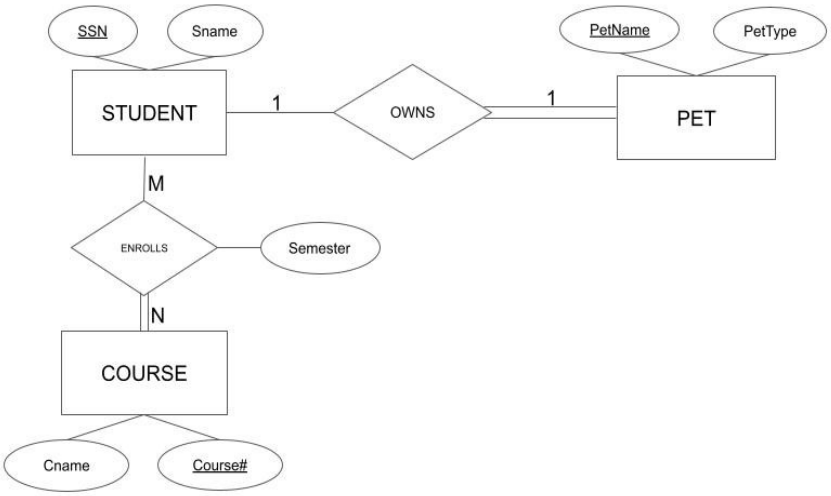
TUTORIAL+SOLUTIONS

This tutorial includes questions on three topics:

1) Informal Guidelines for Relational Design
2) Functional Dependencies, Closure and Equivalence
3) Minimum cover of FDs.

Q1. ABOUT INFORMAL DESIGN GUIDELINES:

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



A. Suppose that only 30% of the students own a pet. The relational design for this schema was done by mapping into the following table:

STUDENT_PET (Student_Ssn, Sname, Petname, Pettype)

i) What informal guideline(s) is/are violated by this design?
ii) What type of problem would be caused in this table?
iii) Propose a correct design that will eliminate null values.
iv) List the functional dependencies present in the relation STUDENT_PET .
v) List the functional dependencies in your correct design. Can you say that functional dependencies are preserved in the correct design but the nulls are eliminated?
vi) If the OWNS relationship type was modified to be One-to-many , allowing a student to own multiple pets, would your design above change? Why or why not?

ANSWER:

1A.

(i) It violates the guideline that attributes from entities should not be mixed inappropriately and that null values should be avoided as much as possible

(ii) In this table 70% of the tuples will have null values under the columns Petname and Pettype

(iii) The correct design that eliminates the problem has two tables:
STUDENT (SSN, Sname)
PET (Petname, Pettype, Student_SSN)
In PET, Student_SSN is a foreign key.

(iv) The given relation has FDs:
{ *Student_SSN* $\rightarrow$ *Sname, Petname, Pettype, and*
Petname $\rightarrow$ *Pettype, Petname* $\rightarrow$ *Student_SSN* }

(v) Relation STUDENT has FD : *Student_SSN* $\rightarrow$ *Sname.*
Relation PET has FD: *Petname* $\rightarrow$ *Pettype, Student_SSN.*
Yes, the correct design maintains the FDs but eliminates the nulls.

(vi) Making the OWNS relationship type one-to-many does NOT change the above correct design. The foreign key Student_SSN in the PET relation takes care of the one-to-many relationship type.

++++++++++++++++++++++++++++

B. Suppose the ENROLLS relationship type was mapped into the following table:
ENROLLS (SSN, Course#, Student_name, Cname, Semester)

i) What informal guideline(s) are being violated here?

ii) Due to the violation of these guidelines, Insertion, deletion and update anomalies will be present. Give one example of each anomaly.

iii) Propose a correct design that will eliminate these anomalies.

iv) List the functional dependencies present in the relation ENROLLS.

v) List the functional dependencies in your correct design. Can you say that functional dependencies are preserved in the correct design but the anomalies are eliminated?

vi) If each student was allowed to take only one course per semester, would you still maintain your design? Give a proper explanation.

ANSWER:

1B.

(i) It violates the informal guideline that attributes from relationship type and entity types should not be mixed inappropriately. The table should be easy to explain as one that stands for an entity type or a relationship type.

(ii) Insertion anomaly: A student cannot be inserted unless he/she is enrolled in some course
Deletion anomaly: Deletion of a course may result in accidentally deleting student(s) that are enrolled only in that course.
Update anomaly: If a Course_name is changed it will cause updating each row for every student enrolled in that course.

(iii) The correct design to eliminate these anomalies:
STUDENT (Student_SSN, Sname)
COURSE (Course#, Cname)
ENROLLS_CORRECT (Student_SSN, Course#, Semester)

(iv) FDs present in ENROLLS:

{ *Student_SSN, Course#* $\rightarrow$ *Sname, Cname , Semester*

Student_SSN $\rightarrow$ *Sname,* *Course#* $\rightarrow$ *Cname}*

(v) *In the correct design:*

STUDENT has FD: *Student_SSN* $\rightarrow$ *Sname*

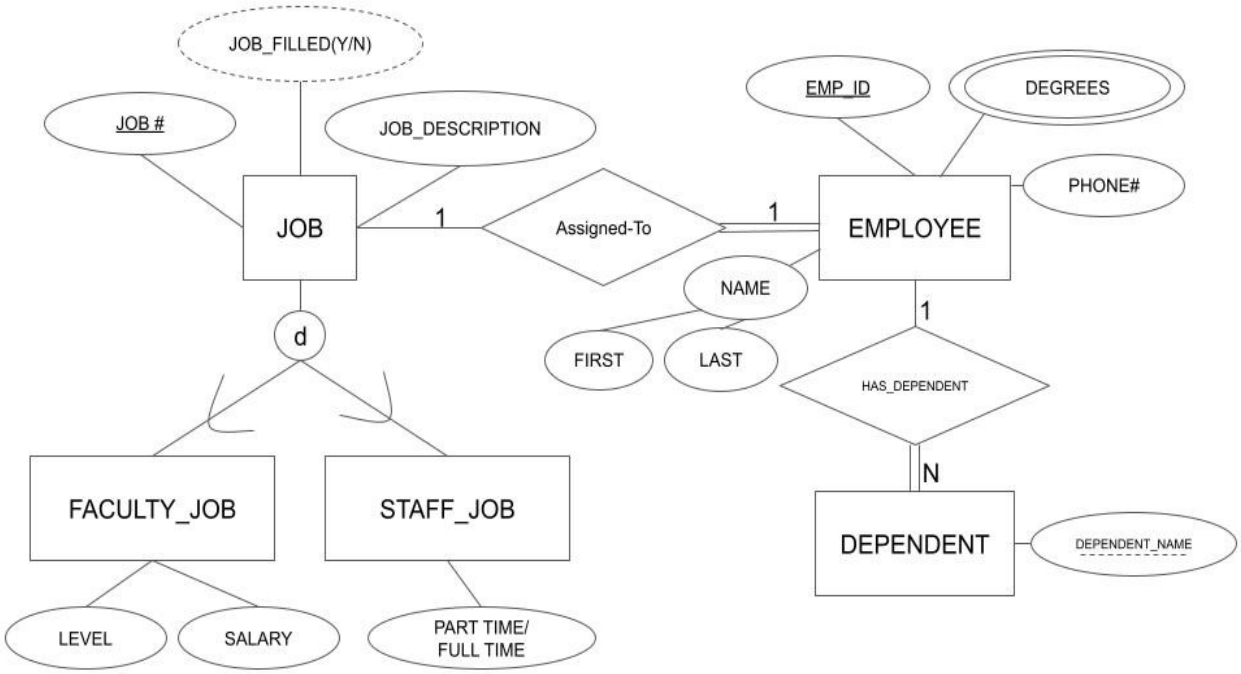
COURSE has FD: *Course#* $\rightarrow$ *Cname*

ENROLLS_CORRECT has FD: *Student_SSN, Course#* $\rightarrow$ *Semester*

By applying the Armstrong Axiom of augmentation, we see that the set of FDs in correct design are equivalent to the FDs in the poorly designed ENROLLS table; yet it avoids all the anomalies.

(vi) If the student is allowed to take only one course per semester, the relationship ENROLLS still remains many-to-many and the correct design still remains as shown above. The constraint acts as a semantic constraint on the table ENROLLS_CORRECT to make sure that for a given semester there are no two distinct Course#s for a given Student_SSN. That will need to be enforced externally by an application.
An alternate solution is to change the design to make it an all-key relation so that the new key includes Semester:
ENROLLS_CORRECT (Student_SSN, Semester, Course#)

C. Consider the EER schema below and answer the following questions:



C1. About relational design based on the JOB Superentity type

i) Show a possible design where the super entity type JOB and the sub-entity types FACULTY_JOB and STAFF_JOB are all accommodated in one table called JOBS.

ii) If only 10% of jobs are faculty jobs and 90% jobs are staff jobs, what potential problem exists?

iii) Propose a design where null values will not occur and jobs will have a discriminating attribute that defines it as a faculty or staff job.

ANSWER:

C1.

(i) Table design using one table:
JOBS (Job#, Job_Descript, Faculty_level, Faculty_salary, Staff_full_parttime, Job_type)

(ii) The attributes < Faculty_level, Faculty_salary> will have null values for 90% rows and the attribute Staff_full_parttime will have null values for 10% rows.

(iii) The correct design will have 3 tables:
JOB (Job#, Job_Descript, Job_type)
FACULTY_JOB (Job#, Fac_level, Fac_salary)
STAFF_JOB (Job#, Staff_full_parttime)

C2. About the ASSIGNED_To relationship type:

Suppose there are 1000 Jobs in the populated database and a table was stored as:
JOB_ASSIGNMENT (Job#, Job_Description, Emp_id, Degree, Phone#)
80% of jobs have been filled and 50% of employees have two degrees.

i) Assume that this table has been populated with the 1000 jobs announced including those that have been filled. What is the nature of redundancy in the above design?

ii) How many tuples will get stored in the JOB_ASSIGNMENT table?

iii) How many null values will get stored?

iv) How many duplicate values will get stored?

v) Assuming that the JOB table is stored as follows:
JOBS (Job#, Job_description, Job_type).
Propose a proper design that takes care of the Assigned-to relationship type and the multivalued attribute Degrees correctly and point out the estimated number of tuples in each table.

vi) List the functional dependencies in your correct design. Can you say that functional dependencies among the attributes in the original EER schema are preserved in the correct design?

ANSWER:

C2.

(i) The redundancy in this table arises from the fact that the multivalued attribute Degrees has been accommodated as a column in this table causing Employee and Job information to be repeated for the Employees with multiple degrees. A large number of null values will be the result.

(ii) JOB_ASSIGNMENT table will have 1000 tuples for the 1000 jobs and 400 extra tuples for 50% of the employees who have 2 degrees. Thus 1400 total tuples.

(iii) 200 unfilled jobs result in 400 null values under the two columns Emp_id and Degree.

(iv) 400 tuples for employees with 2 degrees will have duplicate values stored under first 3 columns of the JOB_ASSIGNMENT table giving 1200 duplicate values.

(v) A proper design of relations to account for EMPLOYEE entity type and ASSIGNED_TO relationship type is:
EMPLOYEE (Emp_id, Phone#, Job#) – 800 tuples; Job# foreign key takes care of the Assigned-to relationship type.
EMP_DEGREES (Emp_id, Degree) – 1200 tuples; 800 employees have 800 tuples; 400 with 2 degrees have 400 additional tuples.

(vi) The EMPLOYEE table has FD: $Emp\_id \rightarrow Phone\#, Job\#$
EMP_DEGREES table has NO FD.
JOBS table has FDs: $Job\# \rightarrow Phone\#, Job\_Description, Job\_type$
Yes, the FDs among the attributes of the EER schema are preserved.
An additional FD : $Job\# \rightarrow Job\_type$ has been added in this design to take care of the classification of Jobs.

++

Q2. ABOUT CLOSURES of attributes and equivalence of sets of FDs :

A. 1. Consider the relation: $R(A, B, C, D)$
with $F = \{A \rightarrow BCD, C \rightarrow D\}$

What are the closures of attributes A, B, C, D?

2. Consider the set of FDs :

$G = \{ A \rightarrow BC, C \rightarrow D\}$.

Compute the closures of attributes A, B, C, D in the set G.

Are F and G equivalent?

3. Consider the set of FDs:

$H = \{ A \rightarrow BD , A \rightarrow C\}$

Compute the closures of attributes A, B, C, D in the set H.

Are F and H equivalent?

ANSWER:

1. Closures of attributes in F:

- $\{A\}^+ = \{ABCD\}$
- $\{B\}^+ = \{B\}$
- $\{C\}^+ = \{CD\}$
- $\{D\}^+ = \{D\}$

2. Closures of attributes in G:

- Stepwise: $\{A\}^+ = \{ABC\}$
- But given that $C \rightarrow D$, $\{A\}^+ = \{ABCD\}$
- $\{B\}^+ = \{B\}$
- $\{C\}^+ = \{CD\}$
- $\{D\}^+ = \{D\}$

Thus the closures of attributes A, B, C, D in F and G are identical.

F can be derived from G and G can be derived from F.

Thus F and G are equivalent.

3. Closures of attributes in H:

- The Union of $A \rightarrow BD$ and $A \rightarrow C$ gives $\{A\}^+ = \{ABCD\}$
- But $\{B\}^+ = \{B\}$
- $\{C\}^+ = \{C\}$
- $\{D\}^+ = \{D\}$

The closure of C is different in F and H.

The FD , $C \rightarrow D$ in F cannot be derived from H.

Hence F and H are not equivalent.

++

Q3. About Minimum Cover of FDs

Find the minimum cover for the set G of f.d.s:

G: **{ A → C, AC → D, AD→ B, CD → EF, E→F }**

Show your work step by step and explain how you arrive at the final min cover.

ANSWER:

Step 1: Decomposing so that every FD has RHS with only one attribute:

CD → EF can be expressed as { CD → E, CD → F } .

Step2: Extraneous attribute removal

(i) From A→ C, we get AA→ AC by augmentation, which means A→AC ;

Now, A→AC and AC → D, Hence, by transitivity, we conclude A→ D.

Thus the **extraneous attribute C in AC → D is not needed.**

Hence LHS of AC →D can be reduced to just A. Hence A→ D.

So the set becomes:

G' : **{ A → C, A → D, AD→ B, CD → E, CD →F, E→F }**

(ii) From A→ D and AD →B, we find A→ B; hence LHS of AD →B can be reduced to just A with D being extraneous. Hence A→ B.

Thus, the **extraneous attribute D in AD → B is not needed.**

Hence we now get:

G'' : **{ A → C, A → D, A→ B, CD → E, CD →F, E→F }**

Step3: Removal of redundant FDs.
Because CD → E and E→F, by transitivity, CD →F.
Hence, **CD →F can be eliminated**. Then we get:

G''' : **{ A → C, A → D, A→ B, CD → E, E→F }**

No further reduction is possible of any FDs from the above set. Hence, it is the minimum cover.

G'''' = G_{min}

REGROUPING STEP:
Taking the union of first three FDs and next two FDs, we can write:

G_{min} : **{ A → BCD, CD → E, E→F }**

This is the minimum cover of the given dependencies.

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