AML140830

14.19. Suppose that we have the following requirements for a university database that is used to keep track of students' transcripts:

a. The university keeps track of each student's name (Sname), student number (Snum), Social Security number (Ssn), current address (Sc_addr) and birth date (Bdate), sex (Sex), class (Class) ('freshman', 'sophomore', ... ,'graduate'), major department (Major_code), minor department (Minor_code) (if any), and degree program (Prog) ('b.a.', 'b.s.', ... , 'ph.d.'). Both Ssn and student number have unique values for each student.

b. Each department is described by a name (Dname), department code (Dcode), office number (Doffice), office phone (Dphone), and college (Dcollege). Both name and code have unique values for each department.

c. Each course has a course name (Cname), description (Cdesc), course number (Cnum), number of semester hours (Credit), level (Level), and offering department (Cdept). The course number is unique for each course.

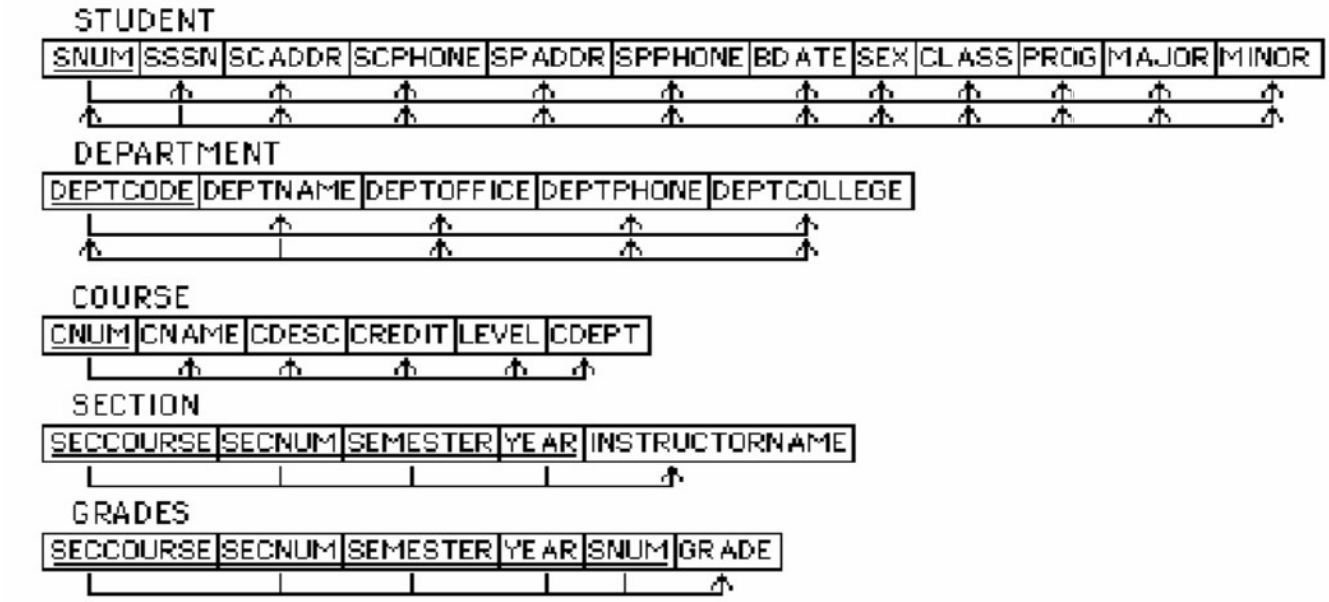
d. Each section has an instructor (Iname), semester (Semester), year (Year), course (Sec_course), and section number (Sec_num). The section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the total number of sections taught during each semester.

e. A grade record refers to a student (Ssn), a particular section, and a grade (Grade).

Design a relational database schema for this database application. First show all the functional dependencies that should hold among the attributes. Then design relation schemas for the database that are each in 3NF or BCNF. Specify the key attributes of each relation. Note any unspecified requirements and make appropriate assumptions to render the specification complete.

From the above description, we can presume that the following functional dependencies hold on the attributes:
FD1: {SSSN} -> {SNAME, SNUM, SCADDR, SCPHONE, SPADDR, SPPHONE, BDATE, SEX, CLASS, MAJOR, MINOR, PROG}
FD2: {SNUM} -> {SNAME, SSSN, SCADDR, SCPHONE, SPADDR, SPPHONE, BDATE, SEX, CLASS, MAJOR, MINOR, PROG}
FD3: {DEPTNAME} -> {DEPTCODE, DEPTOFFICE, DEPTPHONE, DEPTCOLLEGE}
FD4: {DEPTCODE} -> {DEPTNAME, DEPTOFFICE, DEPTPHONE, DEPTCOLLEGE}
FD5: {CNUM} -> {CNAME, CDESC, CREDIT, LEVEL, CDEPT}
FD6: {SECCOURSE, SEMESTER, YEAR, SECNUM} -> {INSTRUCTORNAME}
FD7: {SECCOURSE, SEMESTER, YEAR, SECNUM, SSSN} -> {GRADE}
These are the basic FDs that we can define from the given requirements; using inference rules IR1 to IR3, we can deduce many others. FD1 and FD2 refer to student attributes; we can define a relation STUDENT and choose either SSSN or SNUM as its primary key.

Similarly, FD3 and FD4 refer to department attributes, with either DEPTNAME or DEPTCODE as primary key. FD5 defines COURSE attributes, and FD6 SECTION attributes.
Finally, FD7 defines GRADES attributes. We can create one relation for each of STUDENT, DEPARTMENT, COURSE, SECTION, and GRADES as shown below, where the primary keys are underlined. The COURSE, SECTION, and GRADES relations are in 3NF and BCNF if no other dependencies exist. The STUDENT and DEPARTMENT relations are in 3NF and BCNF according to the general definition given in Sections 14.4 and 14.5, but not according to the definitions of Section 14.3 since both relations have secondary keys.



The foreign keys will be as follows:
STUDENT.MAJOR -> DEPARTMENT.DEPTCODE
STUDENT.MINOR -> DEPARTMENT.DEPTCODE
COURSE.CDEPT -> DEPARTMENT.DEPTCODE
SECTION.SECCOURSE -> COURSE.CNUM
GRADES.(SECCOURSE, SEMESTER, YEAR, SECNUM) ->
SECTION.(SECCOURSE, SEMESTER, YEAR, SECNUM)
GRADES.SNUM -> STUDENT.SNUM
Note: We arbitrarily chose SNUM over SSSN for primary key of STUDENT, and DEPTCODE over DEPTNAME for primary key of DEPARTMENT.

14.20. What update anomalies occur in the EMP_PROJ and EMP_DEPT relations of Figures 14.3 and 14.4?

EMP_PROJ

Can't INSERT a new employee without a project or a new project without an employee.

DELETE employee "Smith, John B." Must delete both tuples. If he had been the last employee on project 1 or 2, would also have to delete that project, losing PNAME, etc..

DELETE project 1. Must delete both tuples. If an associated employee was not working on another project, ENAME, etc. for that employee would be lost.

MODIFY employee "Smith, John B." name Must modify both tuples.
MODIFY project 1's name or location. Must modify both tuples.

EMP_DEPT

Can't INSERT new employee without a department.

DELETE department 5. Must delete 4 tuples.

MODIFY name of department 5. Must modify 4 tuples.

14.21. In what normal form is the LOTS relation schema in Figure 14.12(a) with respect to the restrictive interpretations of normal form that take *only the primary key* into account? Would it be in the same normal form if the general definitions of normal form were used?

Will be in 2NF since there are no partial dependencies on the primary key.

However, it is not in 3NF, since there are the following two transitive dependencies on the primary key:
PROPERTY_ID# ->COUNTY_NAME ->TAX_RATE
PROPERTY_ID# ->AREA ->PRICE

Now, if we take all keys into account and use the general definition of 2NF and 3NF, the LOTS relation schema will only be in 1NF because there is a partial dependency COUNTY_NAME ->TAX_RATE on the secondary key {COUNTY_NAME, LOT#}, which violates 2NF.

14.29. Consider the following relations for an order-processing application database at ABC, Inc.
ORDER (O#, Odate, Cust#, Total_amount)
ORDER_ITEM(O#, I#, Qty_ordered, Total_price, Discount%)

Assume that each item has a different discount. The Total_price refers to one item, Odate is the date on which the order was placed, and the Total_amount is the amount of the order. If we apply a natural join on the relations ORDER_ITEM and ORDER in this database, what does the resulting relation schema look like? What will be its key? Show the FDs in this resulting relation. Is it in 2NF? Is it in 3NF? Why or why not? (State assumptions, if you make any.)

The schema of order *
Schema: ORDER_ITEM Looks like T_1(O#, I#, Odate, Cust#, Total_amount, Qty_ordered, Total_price, Discount%) **key:** O#I#

FDD
Qty_ordered,
O#I#. Discount%,
O#, Odate,
O#, Cust#,
O#. Total_amount.

It is not in 2NF, because Odate, Cust# and Total_amount are only partially dependent on primary key O#I#.

Nor is it in 3NF, as a 2NF is a requirement for 3NF.

14.35. Consider the relation:

BOOK (Book_Name, Author, Edition, Year)
with the data:

Book_Name	Author	Edition	Year
DB_fundamentals	Navathe	4	2004
DB_fundamentals	Elmasri	4	2004
DB_fundamentals	Elmasri	5	2007
DB_fundamentals	Navathe	5	2007

a. Based on a common-sense understanding of the above data, what are the possible candidate keys of this relation?

(Author, Edition), (Author, Copyright_Year), (Book_Name, Author, Edition), (Book_Name, Author, Copyright_Year), (Author, Edition, Copyright_Year), (Book_Name, Author, Edition, Copyright_Year)

All above sets are candidate keys. Any one candidate key can be implemented. (Author, Edition), (Author, Copyright_Year) will be a better choice to implement.

b. Justify that this relation has the MVD { Book }□□{ Author } | { Edition, Year }.

c. What would be the decomposition of this relation based on the above MVD? Evaluate each resulting relation for the highest normal form it possesses.

14.36. Consider the following relation:
TRIP (Trip_id, Start_date, Cities_visited, Cards_used)
This relation refers to business trips made by company salespeople. Suppose the TRIP has a single Start_date, but involves many Cities and salespeople may use multiple credit cards on the trip.Make up a mock-up population of the table.

a. Discuss what FDs and/or MVDs exist in this relation.
FD's

Cities_visited is functionally dependent on the Trip_ID

Cards_used is also functionally dependent on the Trip_ID

MVD's

Cities_visited can have many values

Cards_used can have many values

b. Show how you will go about normalizing it.

I would split the table into two

Trip ID	Start Date	**Cities_Visited**

Trip ID	**Cards used**