CHAPTER 14: Database Design Theory: Introduction to Normalization Using Functional and Multivalued Dependencies

Answers to Selected Exercises

14.19 - Suppose 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 (SSSN), current address (SCADDR) and phone (SCPHONE), permanent address (SPADDR) and phone (SPPHONE), birthdate (BDATE), sex (SEX), class (CLASS) (freshman, sophomore, ..., graduate), major department (MAJORDEPTCODE), minor department (MINORDEPTCODE) (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 (DEPTNAME), department code (DEPTCODE), office number (DEPTOFFICE), office phone (DEPTPHONE), and college (DEPTCOLLEGE). Both name and code have unique values for each department.

(c) Each course has a course name (CNAME), description (CDESC), code number (CNUM), number of semester hours (CREDIT), level (LEVEL), and offering department (CDEPT). The value of code number is unique for each course.

(d) Each section has an instructor (INSTUCTORNAME), semester (SEMESTER), year (YEAR), course (SECCOURSE), and section number (SECNUM). Section numbers distinguish different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ...; up to the number of sections taught during each semester.

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

Design an 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 make the specification complete.

Answer:
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 18.4 and 18.5, but not
according to the definitions of Section 18.3 since both relations have secondary keys.
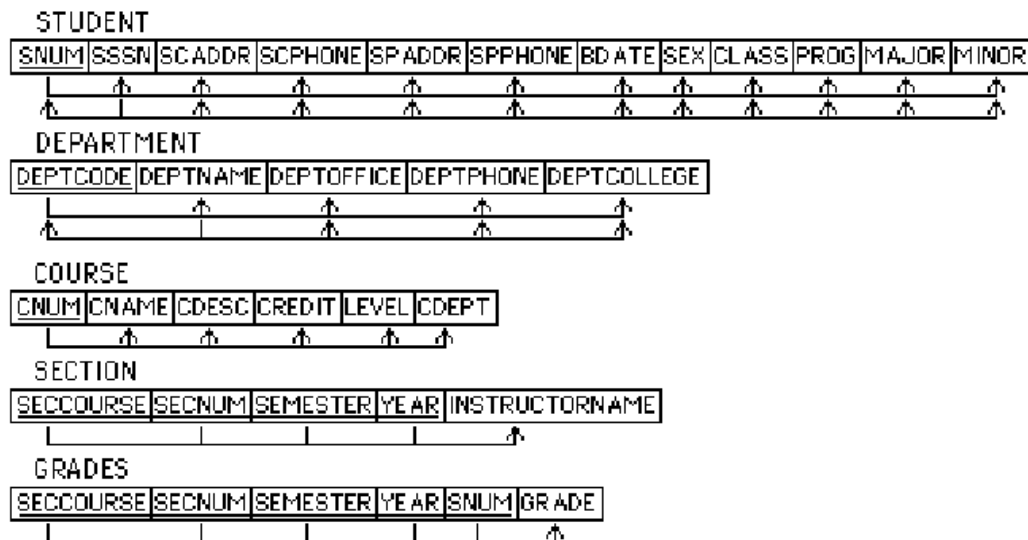
The foreign keys will be as follows:
STUDENT.MAJOR -> DEPARTMENT.DEPTCODE
STUDENT.MINOR -> DEPARTMENT.DEPTCODE
COURSE.CDEPT -> DEPARTMENT.DEPTCODE
SECTION.SECCOURSE -> COURSE.CNUM
GRADES.(SECCOURSE, SEMESTER, YEAR, SECNUM) ->
SECTION.(SECCOURSE, SEMESTER, YEAR, SECNUM)
GRADES.SNUM -> STUDENT.SNUM
Note: 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
Figure 15.3 and 15.4?

Answer:
In EMP_PROJ, the partial dependencies {SSN}->{ENAME} and {PNUMBER}->{PNAME,
PLOCATION} can cause anomalies. For example, if a PROJECT temporarily has no

EMPLOYEEs working on it, its information (PNAME, PNUMBER, PLOCATION) will not be represented in the database when the last EMPLOYEE working on it is removed (deletion anomaly). A new PROJECT cannot be added unless at least one EMPLOYEE is assigned to work on it (insertion anomaly). Inserting a new tuple relating an existing EMPLOYEE to an existing PROJECT requires checking both partial dependencies; for example, if a different value is entered for PLOCATION than those values in other tuples with the same value for PNUMBER, we get an update anomaly. Similar comments apply to EMPLOYEE information. The reason is that EMP_PROJ represents the relationship between EMPLOYEEs and PROJECTs, and at the same time represents information concerning EMPLOYEE and PROJECT entities.

In EMP_DEPT, the transitive dependency {SSN}->{DNUMBER}->{DNAME, DMGRSSN} can cause anomalies. For example, if a DEPARTMENT temporarily has no EMPLOYEEs working for it, its information (DNAME, DNUMBER, DMGRSSN) will not be represented in the database when the last EMPLOYEE working on it is removed (deletion anomaly). A new DEPARTMENT cannot be added unless at least one EMPLOYEE is assigned to work on it
(insertion anomaly). Inserting a new tuple relating a new EMPLOYEE to an existing DEPARTMENT requires checking the transitive dependencies; for example, if a different value is entered for DMGRSSN than those values in other tuples with the same value for DNUMBER, we get an update anomaly. The reason is that EMP_DEPT represents the relationship between EMPLOYEEs and DEPARTMENTs, and at the same time represents information concerning EMPLOYEE and DEPARTMENT entities.

14.21 - In what normal form is the LOTS relation schema in Figure 15.12(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?

Answer:
If we only take the primary key into account, the LOTS relation schema in Figure 14.11 (a) 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, and
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.

14.22 - Prove that any relation schema with two attributes is in BCNF.

Answer:
Consider a relation schema R={A, B} with two attributes. The only possible (non-trivial) FDs are {A} ->{B} and {B} ->{A}. There are four possible cases:
(i) No FD holds in R. In this case, the key is {A, B} and the relation satisfies BCNF.
(ii) Only {A} ->{B} holds. In this case, the key is {A} and the relation satisfies BCNF.
(iii) Only {B} ->{A} holds. In this case, the key is {B} and the relation satisfies BCNF.
(iv) Both {A} ->{B} and {B} ->{A} hold. In this case, there are two keys {A} and {B} and the relation satisfies BCNF.
Hence, any relation with two attributes is in BCNF.

14.23 - Why do spurious tuples occur in the result of joining the EMP_PROJ1 and EMPLOCS relations of Figure 15.5 (result shown in Figure 15.6)?

Answer:
In EMP_LOCS, a tuple (e, l) signifies that employee with name e works on some project located in location l. In EMP_PROJ1, a tuple (s, p, h, pn, l) signifies that employee with social security number s works on project p that is located at location l. When we join EMP_LOCS with EMP_PROJ1, a tuple (e, l) in EMP_LOCS can be joined with a tuple (s, p, h, pn, l) in EMP_PROJ1 where e is the name of some employee and s is the social security number of a different employee, resulting in spurious tuples. The lossless join property (see Chapter 13) can determine whether or not spurious tuples may result based on the FDs in the two relations being joined.

14.24 - Consider the universal relation R = {A, B, C, D, E, F, G, H, I} and the set of functional dependencies F = { {A, B} -> {C}, {A} -> {D, E}, {B} -> {F}, {F} ->
{G, H}, {D} -> {I, J} }. What is the key for R? Decompose R into 2NF, then 3NF relations.

Answer:
A minimal set of attributes whose closure includes all the attributes in R is a key. (One can also apply algorithm 15.4a (see chapter 15 in the textbook)). Since the closure of {A, B}, {A, B}+ = R, one key of R is {A, B} (in this case, it is the only key).
To normalize R intuitively into 2NF then 3NF, we take the following steps (alternatively, we can apply the algorithms discussed in Chapter 15):
First, identify partial dependencies that violate 2NF. These are attributes that are functionally dependent on either parts of the key, {A} or {B}, alone. We can calculate the closures {A}+ and {B}+ to determine partially dependent attributes:
{A}+ = {A, D, E, I, J}. Hence {A} -> {D, E, I, J} ({A} -> {A} is a trivial dependency)
{B}+ = {B, F, G, H}, hence {A} -> {F, G, H} ({B} -> {B} is a trivial dependency)
To normalize into 2NF, we remove the attributes that are functionally dependent on part of the key (A or B) from R and place them in separate relations R1 and R2, along with the part of the key they depend on (A or B), which are copied into each of these relations but also remains in the original relation, which we call R3 below:
R1 = {A, D, E, I, J}, R2 = {B, F, G, H}, R3 = {A, B, C}
The new keys for R1, R2, R3 are underlined. Next, we look for transitive dependencies in R1, R2, R3. The relation R1 has the transitive dependency {A} -> {D} -> {I, J}, so we remove the transitively dependent attributes {I, J} from R1 into a relation R11 and copy the attribute D they are dependent on into R11. The remaining attributes are kept in a relation R12. Hence, R1 is decomposed into R11 and R12 as follows:
R11 = {D, I, J}, R12 = {A, D, E}
The relation R2 is similarly decomposed into R21 and R22 based on the transitive dependency {B} -> {F} -> {G, H}:
R2 = {F, G, H}, R2 = {B, F}
The final set of relations in 3NF are {R11, R12, R21, R22, R3}

14.25 - Repeat exercise 15.24 for the following different set of functional dependencies G = { {A, B} -> {C}, {B, D} -> {E, F}, {A, D} -> {G, H}, {A} -> {I}, {H} -> {J} }.

Answer:
To help in solving this problem systematically, we can first find the closures of all single attributes to see if any is a key on its own as follows:
{A}+ -> {A, I}, {B}+ -> {B}, {C}+ -> {C}, {D}+ -> {D}, {E}+ -> {E}, {F}+ -> {F}, {G}+ -> {G}, {H}+ -> {H, J}, {I}+ -> {I}, {J}+ -> {J}
Since none of the single attributes is a key, we next calculate the closures of pairs of attributes that are possible keys:
{A, B}+ -> {A, B, C, I}, {B, D}+ -> {B, D, E, F}, {A, D}+ -> {A, D, G, H, I, J}
None of these pairs are keys either since none of the closures includes all attributes. But the union of the three closures includes all the attributes:
{A, B, D}+ -> {A, B, C, D, E, F, G, H, I}
Hence, {A, B, D} is a key. (Note: Algorithm 15.4a (see chapter 15 in the textbook) can be used to determine a key).
Based on the above analysis, we decompose as follows, in a similar manner to problem 14.26, starting with the following relation R:
R = {A, B, D, C, E, F, G, H, I}
The first-level partial dependencies on the key (which violate 2NF) are:
{A, B} -> {C, I}, {B, D} -> {E, F}, {A, D}+ -> {G, H, I, J}
Hence, R is decomposed into R1, R2, R3, R4 (keys are underlined):
R1 = {A, B, C, I}, R2 = {B, D, E, F}, R3 = {A, D, G, H, I, J}, R4 = {A, B, D}
Additional partial dependencies exist in R1 and R3 because {A} -> {I}. Hence, we remove {I} into R5, so the following relations are the result of 2NF decomposition:
R1 = {A, B, C}, R2 = {B, D, E, F}, R3 = {A, D, G, H, J}, R4 = {A, B, D}, R5 = {A, I}
Next, we check for transitive dependencies in each of the relations (which violate 3NF). Only R3 has a transitive dependency {A, D} -> {H} -> {J}, so it is decomposed into R31 and R32 as follows:
R31 = {H, J}, R32 = {A, D, G, H}
The final set of 3NF relations is {R1, R2, R31, R32, R4, R5}

14.26 – No solution provided.

14.27 – Consider a relation *R(A,B,C,D,E)* with the following dependencies:
AB -> C
CD -> E
DE -> B

Is *AB* a candidate key of this relation? If not, is *ABD*? Explain your answer.

Answers:
No, AB+ = {A,B,C},a proper subset of {A,B,C,D,E}
Yes, ABD+ = {A,B,C,D,E}

14.28 - Consider the relation R, which has attributes that hold schedules of courses and sections at a university; R = {CourseNo, SecNo, OfferingDept, CreditHours, CourseLevel, InstructorSSN, Semester, Year, Days_Hours, RoomNo, NoOfStudents}. Suppose that the following functional dependencies hold on R:
{CourseNo} -> {OfferingDept, CreditHours, CourseLevel}
{CourseNo, SecNo, Semester, Year} ->
{Days_Hours, RoomNo, NoOfStudents, InstructorSSN}
{RoomNo, Days_Hours, Semester, Year} -> {InstructorSSN, CourseNo, SecNo}

Try to determine which sets of attributes form keys of R. How would you normalize this relation?

Answer:
Let us use the following shorthand notation:
C = CourseNo, SN = SecNo, OD = OfferingDept, CH = CreditHours, CL = CourseLevel,
I = InstructorSSN, S = Semester, Y = Year, D = Days_Hours, RM = RoomNo,
NS = NoOfStudents
Hence, R = {C, SN, OD, CH, CL, I, S, Y, D, RM, NS}, and the following functional dependencies hold:
{C} -> {OD, CH, CL}
{C, SN, S, Y} -> {D, RM, NS, I}
{RM, D, S, Y} -> {I, C, SN}
First, we can calculate the closures for each left hand side of a functional dependency, since these sets of attributes are the candidates to be keys:
(1) {C}+ = {C, OD, CH, CL}
(2) Since {C, SN, S, Y} -> {D, RM, NS, I}, and {C}+ = {C, OD, CH, CL}, we get:
{C, SN, S, Y}+ = {C, SN, S, Y, D, RM, NS, I, OD, CH, CL} = R
(3) Since {RM, D, S, Y} -> {I, C, SN}, we know that {RM, D, S, Y}+ contains {RM, D, S, Y, I, C, SN}. But {C}+ contains {OD, CH, CL} so these are also contained in {RM, D, S, Y}+ since C is already there. Finally, since {C, SN, S, Y} are now all in {RM, D, S, Y}+ and {C, SN, S, Y}+ contains {NS} (from (2) above), we get:
{RM, D, S, Y}+ = {RM, D, S, Y, I, C, SN, OD, CH, CL, NS} = R
Hence, both K1 = {C, SN, S, Y} and K2 = {RM, D, S, Y} are (candidate) keys of R. By applying the general definition of 2NF, we find that the functional dependency {C} -> {OD, CH, CL} is a partial dependency for K1 (since C is included in K1). Hence, R is normalized into R1 and R2 as follows:
R1 = {C, OD, CH, CL}
R2 = {RM, D, S, Y, I, C, SN, NS} with candidate keys K1 and K2
Since neither R1 nor R2 have transitive dependencies on either of the candidate keys, R1 and R2 are in 3NF also. They also both satisfy the definition of BCNF.

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 any assumptions you make.)

Answer:
Given relations
Order(O#, Odate, Cust#, Total_amt)
Order_Item(O#, I#, Qty_ordered, Total_price, Discount%),
the schema of Order * Order_Item looks like

T₁(O#,I#,Odate, Cust#, Total_amount, Qty_ordered, Total_price, Discount%)
and its key is O#,I#.
It has functional dependencies
O#I# . Qty_ordered
O#I# .Total_price
O#I# .Discount%
O# . Odate
O# .Cust#
O# .Total_amount
It is not in 2NF, as attributes Odate, Cut#, and Total_amount are only partially
dependent on the primary key, O#I#
Nor is it in 3NF, as a 2NF is a requirement for 3NF.

14.30 - Consider the following relation:
CAR_SALE(Car#, Date_sold, Salesman#, Commision%, Discount_amt
Assume that a car may be sold by multiple salesmen and hence {CAR#, SALESMAN#} is the
primary key. Additional dependencies are:
Date_sold ->Discount_amt
and
Salesman# ->commission%

Based on the given primary key, is this relation in 1NF, 2NF, or 3NF? Why or why not? How
would you successively normalize it completely?

Answer:
Given the relation schema
Car_Sale(Car#, Salesman#, Date_sold, Commission%, Discount_amt)
with the functional dependencies
Date_sold ☐☐Discount_amt
Salesman# ☐☐Commission%
Car# ☐☐Date_sold
This relation satisfies 1NF but not 2NF (Car# ☐☐Date_sold and Car# ☐
Discount_amt
so these two attributes are not FFD on the primary key) and not 3NF.
To normalize,
2NF:
Car_Sale1(Car#, Date_sold, Discount_amt)
Car_Sale2(Car#, Salesman#)
Car_Sale3(Salesman#,Commission%)
3NF:
Car_Sale1-1(Car#, Date_sold)
Car_Sale1-2(Date_sold, Discount_amt)
Car_Sale2(Car#, Salesman#)
Car_Sale3(Salesman#,Commission%)

14.31 - Consider the following relation for published books:
BOOK (Book_title, Authorname, Book_type, Listprice, Author_affil, Publisher)
Author_affil referes to the affiliation of the author. Suppose the following dependencies exist:
Book_title -> Publisher, Book_type
Book_type -> Listprice
Author_name -> Author-affil

(a) What normal form is the relation in? Explain your answer.
(b) Apply normalization until you cannot decompose the relations further. State the reasons
    behind each decomposition.

Answer:
Given the relation
Book(Book_title, Authorname, Book_type, Listprice, Author_affil, Publisher)
and the FDs
Book_title □□Publisher, Book_type
Book_type □□Listprice
Authorname □Author_affil

(a)The key for this relation is Book_title,Authorname. This relation is in 1NF and not in
2NF as no attributes are FFD on the key. It is also not in 3NF.

(b) 2NF decomposition:
Book0(Book_title, Authorname)
Book1(Book_title, Publisher, Book_type, Listprice)
Book2(Authorname, Author_affil)
This decomposition eliminates the partial dependencies.
3NF decomposition:
Book0(Book_title, Authorname)
Book1-1(Book_title, Publisher, Book_type)
Book1-2(Book_type, Listprice)
Book2(Authorname, Author_affil)

This decomposition eliminates the transitive dependency of Listprice

14.32 - This exercise asks you to converting business statements into dependencies.
Consider the following relation DiskDrive(serialNumber, manufacturer, model, batch,
capacity, retailer). Each tuple in the relation DiskDrive contains information about a disk drive
with a unique serialNumber, made by a manufacturer, with a particular model, released in a
certain batch, which has a certain storage capacity, and is sold by a certain retailer. For
example, the tuple DiskDrive(1978619, WesternDigital, A2235X, 765234, 500, CompUSA)
specifies that WesternDigital made a disk drive with serial number 1978619, model number
A2235X in batch 765235 with 500GB that is sold by CompUSA.

Write each of the following dependencies as an FD:
a. The manufacturer and serial number uniquely identifies the drive
b. A model number is registered by a manufacturer and hence can't be used by another
    manufacturer.
c. All disk drives in a particular batch are the same model.
d. All disk drives of a particular model of a particular manufacturer have exactly the same
    capacity.

Answer:

a. manufacturer, serialNumber → model, batch, capacity, retailer

b. model → manufacturer

c. manufacturer, batch → model

d. model → capacity (Comment: 10.35.d can be "model, manufacturer → capacity." There is nothing in the question that suggests we should "optimize" our FDs by assuming the other requirements)

14.33 - Consider the following relation:

R (Doctor#, Patient#, Date, Diagnosis, Treat_code, Charge)

In this relation, a tuple describes a visit of a patient to a doctor along with a treatment code and daily charge. Assume that diagnosis is determined (uniquely) for each patient by a doctor. Assume that each treatment code has a fixed charge (regardless of patient). Is this relation in 2NF? Justify your answer and decompose if necessary. Then argue whether further normalization to 3NF is necessary, and if so, perform it.

Answer:
From the question's text, we can infer the following functional dependencies:

{Doctor#, Patient#, Date}→{Diagnosis, Treat_code, Charge}
{Treat_code}→{Charge}

Because there are no partial dependencies, the given relation is in 2NF already. This however is not 3NF because the Charge is a nonkey attribute that is determined by another nonkey attribute, Treat_code. We must decompose further:

R (Doctor#, Patient#, Date, Diagnosis, Treat_code)
R1 (Treat_code, Charge)

We could further infer that the treatment for a given diagnosis is functionally dependant, but we should be sure to allow the doctor to have some flexibility when prescribing cures.

14.34 - Consider the following relation:
CAR_SALE (CarID, Option_type, Option_Listprice, Sale_date, Discounted_price)
This relation refers to options installed on cars (e.g.- cruise control) that were sold at a dealership and the list and discounted prices for the options.
If CarID → Sale_date and Option_type→ Option_Listprice, and
CarID, Option_type → Discounted_price, argue using the generalized definition of the 3NF that this relation is not in 3NF. Then argue from your knowledge of 2NF, why it is not in 2NF.

Answer:
For this relation to be in 3NF, all of the nontrivial functional dependencies must both be fully functional and nontransitive on every key of the relation. However, in this relation we have two dependencies (CarID → Sale_date and Option_type → Option_Listprice) that violate

these requirements. Both of these dependencies are partial and transitive.

For this relation to be in 2NF, all of the nontrivial functional dependencies must be fully functional on every key of the relation. Again, as was discussed about 3NF, this relation has two partial dependencies that violate this requirement.

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?

b. Does the above have one or more functional dependency (do not list FDs by applying derivation rules)? If so, what is it? Show how you will remove it by decomposition.

c. Does the resulting relation have an MVD? If so, what is it?

d. What will the final decomposition look like?

Answer:
a. The only candidate key is {Book_Name, Author, Edition}. From the example, it would appear that {Book_Name, Author, Year} would also be a candidate key but we should consider that some books may have a release cycle which causes multiple editions to appear in a given year.

b. Yes, we have the following FD: Book_Name, Edition $\rightarrow$ Year. We can decompose to remove this FD in the following way:

     BOOK (<u>Book_Name</u>, <u>Author</u>, <u>Edition</u>)
     BOOK_YEAR (<u>Book_Name</u>, <u>Edition</u>, Year)

c. Yes, BOOK contains Book_Name $\rightarrow\rightarrow$ Author and Book_Name $\rightarrow\rightarrow$ Edition.

d. The final decomposition would look like:

     BOOK (<u>Book_Name</u>, <u>Edition</u>)
BOOK_AUTHOR (<u>Book_Name</u>, <u>Edition</u>, Author)
     BOOK_YEAR (<u>Book_Name</u>, <u>Edition</u>, Year)

14.36 - Consider the following relation:
     TRIP (trip_id, start_date, cities_visited, cards_used)

This relation refers to business trips made by salesmen in a company. Suppose the trip has a single start_date but involves many cities and one may use multiple credit cards for that trip. Make up a mock-up population of the table.

     a. Discuss what FDs and / or MVDs exist in this relation.
     b. Show how you will go about normalizing it.

Answer:
a. The TRIP relation has the following FDs and MVDs:

     trip_id $\rightarrow$ start_date
     trip_id $\rightarrow\rightarrow$ cities_visited
     trip_id $\rightarrow\rightarrow$ cards_used

b. Because there are no interdependencies, this relation can be trivially decomposed to conform to 4NF:

     TRIP_DATE (trip_id, start_date)
     TRIP_CITIES (trip_id, cities_visited)
     TRIP_CARDS (trip_id, cards_used)
.