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 phone (Sc_phone), permanent address (Sp_addr) and phone (Sp_phone), 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.

Step 1 of 4

Functional Dependency:

Functional dependency exists when one attribute in a relation uniquely determines another attribute. Functional dependency is represented as XY. X and Y can be composite.

The functional dependencies from the given information are as follows:

$$FD\,1: Ssn \rightarrow \left\{\begin{matrix} Sname, Snum, Sc\_addr, Sc\_phone, Sp\_addr, Sp\_phone, \\ Bdate, Sex, Class, Major\_code, Minor\_code, Prog \end{matrix}\right\}$$

$$FD\,2: Snum \rightarrow \left\{\begin{matrix} Sname, Ssn, Sc\_addr, Sc\_phone, Sp\_addr, Sp\_phone, \\ Bdate, Sex, Class, Major\_code, Minor\_code, Prog \end{matrix}\right\}$$

$$FD\,3: Dname \rightarrow \{Dcode, Doffice, Dphone, Dcollege\}$$

$$FD\,4: Dcode \rightarrow \{Dname, Doffice, Dphone, Dcollege\}$$

$$FD\,5: Cnum \rightarrow \{Cname, Cdesc, Credit, Level, Cdept\}$$

$$FD\,6: \{Sec\_num, Sec\_course, Semester, Year\} \rightarrow Iname$$

$$FD\,7: \{Ssn, Sec\_course, Semester, Year\} \rightarrow Grade$$

$$FD\,8: \{Ssn, Sec\_num, Semester, Year\} \rightarrow Grade$$

Step 2 of 4

From the functional dependencies FD 1 and FD 2, the relation STUDENT can be defined. Either Ssn or Snum can be primary key.

From the functional dependencies FD 3 and FD 4, the relation DEPARTMENT can be defined. Either Dname or Dcode can be primary key.

From the functional dependencies FD 5, the relation COURSE can be defined. Cnum is the primary key.

From the functional dependencies FD 6, the relation SECTION can be defined. Sec_num, Sec_course, Semester, Year will be the composite primary key.

From the functional dependencies FD 7 and FD 8, the relation GRADE can be defined. {Ssn, Sec_course, Semester, Year} will be the composite primary key.

Step 3 of 4

The relations that are in third normal form are as follows:

$$\text{STUDENT}\left(\begin{array}{l}\text{Ssn, Sname, Snum, Sc\_addr, Sc\_phone, Sp\_addr, Sp\_phone,}\\ \text{Bdate, Sex, Class, Major\_code, Minor\_code, Prog}\end{array}\right)$$

$$\text{DEPARTMENT}\left(\text{Dname, Dcode, Doffice, Dphone, Dcollege}\right)$$

$$\text{COURSE}\left(\text{Cnum, Cname, Cdesc, Credit, Level, Cdept}\right)$$

$$\text{SECTION}\left(\text{Sec\_num, Sec\_course, Semester, Year, Iname}\right)$$

$$\text{GRADE}\left(\text{Ssn, Sec\_course, Semester, Year, Grade}\right)$$

Explanation:

• In STUDENT relation, either Ssn or Snum can be primary key. Either keys can be used to retrieve the data from the STUDENT table.

• In DEPARTMENT relation, either Dname or Dcode can be primary key. Either keys can be used to retrieve the data from the DEPARTMENT table.

• In COURSE table, Cnum is the primary key.

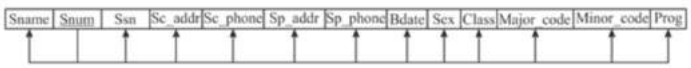
• The primary key for the SECTION table is {Sec_num, Sec_course, Semester, Year} which is a composite primary key.

• The primary key for the GRADE table is {Ssn, Sec_course, Semester, Year} which is a composite primary key.

Step 4 of 4

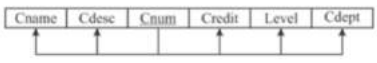
The relational schema is as follows:

STUDENT

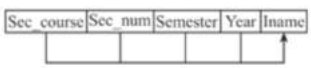


DEPARTMENT

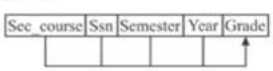
COURSE



SECTION



GRADE



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

Step 1 of 6

Insertion anomaly refers to the situation where it is not possible to enter data of certain attributes into the database without entering data of other attributes.

Deletion anomaly refers to the situation where data of certain attributes are lost as the result of deletion of some of the attributes.

Modification anomaly refers to the situation where partial update of redundant data leads to inconsistency of data.

Comment

Step 2 of 6

Insertion, deletion and modification anomalies are considered bad due to the following reasons:

- It will be difficult to maintain consistency of data in the database.
- It leads to redundant data.
- It causes unnecessary updates of data.
- Memory space will be wasted at the storage level.

Step 3 of 6

Consider the following relation named Emp_Proj:

Empno	Ename	Job	Salary	ProjectID	Pname	Location
E01	JamesWilson	Manager	20000	P1	ProjectX	Atlanta
E07	Robbinson	Programmer	10000	P5	ProjectZ	London
E08	Jasper Smith	Programmer	12000	P10	ProjectY	Seattle
E10	Allen Jones	Analyst	12000	P1	ProjectX	Atlanta
E02	Emily Park	Analyst	12000	P1	ProjectX	Atlanta

Insertion Anomalies:

- Assume that there is an employee E11 who is not yet working in a project. Then it is not possible to enter details of employee E11 into the relation Emp_Proj.
- Similarly assume there is a project P7 with no employees assigned to it. Then it is not possible to enter details of project P7 into the relation Emp_Proj.
- Therefore, it is possible to enter an employee details into relation Emp_Proj only if he is assigned to a project.
- Similarly, it is possible to enter details of a project into relation Emp_Proj only if an employee is assigned to a project.

Step 4 of 6

Deletion Anomalies:

• Assume that an employee E07 has left the company. So, it is necessary to delete employee E07 details from the relation Emp_Pro.

• If employee E07 details are deleted from the relation Emp_Pro, then the details of project P5 will also be lost.

Update anomalies:

• Assume that the location of project P1 is changed from Atlanta to New Jersey. Then the update should be done at three places.

• If the update is reflected for two tuples and is not done for the third tuple, then inconsistency of data occurs.

Comment

Step 5 of 6

In order to remove insertion, deletion and modification anomalies, decompose the relation Emp_Proj into three relations as shown below:

Employee:

Empno	Ename	Job	Salary
E01	JamesWilson	Manager	20000
E07	Robbinson	Programmer	10000
E08	Jasper Smith	Programmer	12000
E10	Allen Jones	Analyst	12000
E02	Emily Park	Analyst	12000

Project:

ProjectID	Pname	Location
P1	ProjectX	Atlanta
P5	ProjectZ	London
P10	ProjectY	Seattle

Emp_Proj:

Empno	ProjectID
E01	P1
E07	P5
E08	P10
E10	P1
E02	P1

Comment

Step 6 of 6

Insertion Anomalies:

• It is possible to enter the details of employee E11 into relation Employee even though he is not yet working in a project.

• It is possible to enter the details of project P7 into relation Project even though there are no employees assigned to it.

Deletion Anomalies:

• If employee E07 details are deleted from the relation Employee, still the details of project P5 will not be lost.

Update anomalies:

• If the location of project P1 is changed from Atlanta to New Jersey, then the update should be done in relation Project at only one place.

14.24 Consider the universal relation R = {A, B, C, D, E, F, G, H, I, J} 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 and then 3NF relations.

575-10-26E

Let R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies

F = { {A, B}-> {C}, {A}->{D, E}, {B}->{F}, {F}->{G, H}, {D}->{I, J} }

A minimal set of attributes whose closure includes all the attributes in R is a key. Since the closure of {A, B}, {A, B} + = R,

So, one key of R is {A, B}

Decompose R into 2NF and then 3NF

For this normalize R intuitively into 2NF then 3NF, we may follow below steps

Step 1:

Identify partial dependencies and that may violate 2NF. These are attributes that are

functionally dependent on either parts of the key, {A} or {B}, alone.

Now 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

For normalizing into 2NF, we may 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 14.24 for the following different set of functional dependencies G = {{A, B}→{C}, {B, D}→{E, F}, {A, D}→{G, H}, {A}→{I}, {H}→{J}}.

Step 1 of 6

The relation R={ A, B, C, D, E, F, G, H, I, J}

The set of functional dependencies are as follows:

{A, B}{C}

{B, D}{E, F}

{A, D}{G, H}

{A}{I}

{H}{J}

Step 1: Find the closure of single attributes:

$\{A\}^+\{A, I\}$

$\{B\}^+\{B\}$

$\{C\}^+\{C\}$

$\{D\}^+\{D\}$

$\{E\}^+\{E\}$

$\{F\}^+\{F\}$

$\{G\}^+\{G\}$

$\{H\}^+\{H, J\}$

$\{I\}^+\{ I\}$

$\{J\}^+\{J\}$

From the above closures of single attributes, it is clear that the closure of any single attribute does not represent relation R. So, no single attribute forms the key for the relation R.

Step 2 of 6

Step 2: Find the closure of pairs of attributes that are in the set of functional dependencies.

The closure of {A, B} is as shown below:

From the functional dependency {A, B}{C} and {A}{I},

$\{A, B\}^+\{A, B, C, I\}$

The closure of { B, D} is as shown below:

From the functional dependency {B, D}{E, F},

$\{B, D\}^+\{B, D, E, F\}$

The closure of { A, D} is as shown below:

From the functional dependency {A, D}{G, H}, {A}{I} and {H}{J},

$\{A, D\}^+\{A, D, G, H, I, J\}$

From the above closures of pairs of attributes, it is clear that the closure of any pairs of attributes does not represent relation R. So, no single attribute forms the key for the relation

Step 3 of 6

Step 3: Find the closure of union of the three pairs of attributes that are in the set of functional dependencies.

The closure of {A, B, D} is as shown below:

From the functional dependency {A, B}{C}, {B, D}{E, F} and {A, D}{G, H}

{A, B, D}$^+${A, B, C, D, E, F, G, H}

From the functional dependency{A}{I}, the attribute I is added to {A, B, D}$^+$.

Hence, {A, B, D}$^+${A, B, C, D, E, F, G, H, I}

From the functional dependency{H}{J}, the attribute J is added to {A, B, D}$^+$.

Hence, {A, B, D}$^+${A, B, C, D, E, F, G, H, I, J}

The closure of {A, B, D} represents relation R.

Hence, the key for relation R is {A, B, D}.

Step 4 of 6

Decomposing the relation R into second normal form (2NF):

According to the second normal form, each non-key attribute must depend only on primary key.

- The key for relation R is {A, B, D}.
- {A} is a partial key that functionally determines the attribute I.
- {A, B} is a partial key that functionally determines the attribute C.
- {B, D } is a partial key that functionally determines the attribute E and F.
- {A, D} is a partial key that functionally determines the attribute G and H.

So, decompose the relation R into the following relations.

R1{A, I}

The key for R1 is {A}.

R2{A, B, C}

The key for R2 is {A, B}.

R3{B, D, E, F}

The key for R3 is { B, D}.

R4{A, B, D)

The key for R4 is { A, B, D}.

R5{A, D, G, H, J}

The key for R5 is { A, D}.

The relations R1, R2, R3, R4, R5 are in second normal form.

Step 5 of 6

Decomposing the relation R into third normal form (3NF):

According to the third normal form, the relation must be in second normal form and any non-key attribute should not describe any non-key attribute.

• H is a non-key attribute that functionally determines the attribute J.

So, decompose the relation R5 into the following relations.

R6{A, D, G, H,}

The key for R3 is { A, D}.

R7{H, J}

The key for R7 is {H}.

Comment

Step 6 of 6

The final set of relations that re in third normal form are as follows:

R1{A, I}

R2{A, B, C}

R3{B, D, E, F}

R4{A, B, D)

R6{A, D, G, H,}

R7{H, J}

14.26. Consider the following relation:

A	B	C	TUPLE#
10	b1	c1	1
10	b2	c2	2
11	b4	c1	3
12	b3	c4	4
13	b1	c1	5
14	b3	c4	6

a. Given the previous extension (state), which of the following dependencies may hold in the above relation? If the dependency cannot hold, explain why by specifying the tuples that cause the violation.

i. $A \rightarrow B$, ii. $B \rightarrow C$, iii. $C \rightarrow B$, iv. $B \rightarrow A$, v. $C \rightarrow A$

b. Does the above relation have a potential candidate key? If it does, what is it? If it does not, why not?

Step 1 of 2

a)

1.) A->B does not hold good in current state of relation as attribute B has two values corresponding to value 10 of attribute A.

2.) B->C: this relation can hold good in current relation state.

3.) C->B does not hold good in current state of relation as attribute B has two values corresponding to value c1 of attribute C.

4.) B->A does not hold good in current state of relation as attribute A has two values corresponding to value b1 and b3 of attribute B.

5.) C->A does not hold good in current state of relation as attribute A has two values corresponding to value c1, c4 of attribute C.

Comment

Step 2 of 2

b) If value of attribute - TUPLE# remains different for all tuples in relation it can act as candidate key.

Comment

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.

Step 1 of 3

The candidate key is the minimal field or the combination of fields in a relation that can be used to uniquely identify all the other fields of the given relation.

The candidate key is checked using the closure property of the set and the functional dependencies of the given relation.

Step 2 of 3

Consider the given relation R (A, B, C, D, E) and the following function dependencies:

AB C, CD E, DE B

To check whether the key AB is the candidate key of the given relation R, find the closure of AB as shown below:

Closure set of $\{AB\}^+$	Functional Dependency used
{A, B}	Trivial
{A, B, C}	AB → C

Since, all the attributes of the relation R cannot be identified using the key AB, the AB is not the candidate key for the given relation R.

Step 3 of 3

To check whether the key ABD is the candidate key of the given relation R, find the closure of ABD as shown below:

Closure set of $\{ABD\}^+$	Functional Dependency used
{A, B, D}	Trivial
{A, B, C, D}	AB → C
{A, B, C, D, E}	CD → E

Since, all the attributes of the relation R can be identified using the key ABD, the ABD is a candidate key for the given relation R.

Hence, proved.

14.28. Consider the relation R, which has attributes that hold schedules of courses and sections at a university; R = {Course_no, Sec_no, Offering_dept, Credit_hours, Course_level, Instructor_ssn, Semester, Year, Days_hours, Room_no, No_of_students}. Suppose that the following functional dependencies hold on R:

{Course_no} → {Offering_dept, Credit_hours, Course_level}

{Course_no, Sec_no, Semester, Year} → {Days_hours, Room_no, No_of_students, Instructor_ssn}

{Room_no, Days_hours, Semester, Year} → {Instructor_ssn, Course_no,Sec_no}

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

Step 1 of 5

Consider the following relation and functional dependencies:

Relation $R = \begin{Bmatrix} Course\_no, Sec\_no, Offering\_dept, Credit\_hours, \\ Course\_level, Instructor\_ssn, Semester, Year, \\ Days\_hours, Room\_no, No\_of\_students \end{Bmatrix}$

Functional dependencies:

$$\{Course\_no\} \rightarrow \{Offering\_dept, Credit\_hours, Course\_level\}$$

$$\begin{Bmatrix} Course\_no, Sec\_no, \\ Semester, Year \end{Bmatrix} \rightarrow \begin{Bmatrix} Days\_hours, Room\_no, \\ No\_of\_students, Instructor\_ssn \end{Bmatrix}$$

$$\begin{Bmatrix} Room\_no, Days\_hours, \\ Semester, Year \end{Bmatrix} \rightarrow \begin{Bmatrix} Instructor\_ssn, Course\_no, \\ Sec\_no \end{Bmatrix}$$

Comment

Step 2 of 5

The closure of Course_no is as shown below:

From the functional dependency

$$\{Course\_no\} \rightarrow \{Offering\_dept, Credit\_hours, Course\_level\}$$

$$\{Course\_no\}^+ = \{Offering\_dept, Credit\_hours, Course\_level\}$$

The attributes Offering_dept, Credit_hours, Course_level are added to the closure of Course_no as Course_no functionally determines Offering_dept, Credit_hours, Course_level.

Step 3 of 5

The closure of Course_no, Sec_no, Semester, Year is as shown below:

From the functional dependency

$$\begin{Bmatrix} Course\_no, Sec\_no, \\ Semester, Year \end{Bmatrix} \rightarrow \begin{Bmatrix} Days\_hours, Room\_no, \\ No\_of\_students, Instructor\_ssn \end{Bmatrix},$$

the attributes $Days\_hours, Room\_no, No\_of\_students, Instructor\_ssn$ are added to the closure of $Course\_no, Sec\_no, Semester, Year$.

$$\text{Hence, } \begin{Bmatrix} Course\_no, Sec\_no, \\ Semester, Year \end{Bmatrix}^{+} = \begin{Bmatrix} Course\_no, Sec\_no, Semester, Year, \\ Days\_hours, Room\_no, \\ No\_of\_students, Instructor\_ssn \end{Bmatrix}$$

From the functional dependency

$\{Course\_no\} \rightarrow \{Offering\_dept, Credit\_hours, Course\_level\}$,

the attributes $Offering\_dept, Credit\_hours, Course\_level$ are added to the closure of $Course\_no, Sec\_no, Semester, Year$.

$$\text{Hence, } \begin{Bmatrix} Course\_no, Sec\_no, \\ Semester, Year \end{Bmatrix}^{+} = \begin{Bmatrix} Course\_no, Sec\_no, Semester, Year, \\ Days\_hours, Room\_no, \\ No\_of\_students, Instructor\_ssn, \\ Offering\_dept, Credit\_hours, \\ Course\_level \end{Bmatrix}$$

The closure of $\{Course\_no, Sec\_no, Semester, Year\}$ represents the relation R.

Step 4 of 5

The closure of Room_no, Days_hours, Semester, Year is as shown below:

From the functional dependency

$$\begin{Bmatrix} Room\_no, Days\_hours, \\ Semester, Year \end{Bmatrix} \rightarrow \begin{Bmatrix} Instructor\_ssn, Course\_no, \\ Sec\_no \end{Bmatrix},$$

the attributes $Instructor\_ssn, Course\_no, Sec\_no$ are added to the closure of $Room\_no, Days\_hours, Semester, Year$.

$$\text{Hence, } \begin{Bmatrix} Room\_no, Days\_hours, \\ Semester, Year \end{Bmatrix}^{+} \rightarrow \begin{Bmatrix} Room\_no, Days\_hours, Semester, \\ Year, Instructor\_ssn, \\ Course\_no, Sec\_no \end{Bmatrix}$$

From the functional dependency

$\{Course\_no\} \rightarrow \{Offering\_dept, Credit\_hours, Course\_level\}$,

the attributes $Offering\_dept, Credit\_hours, Course\_level$ are added to the closure of $Room\_no, Days\_hours, Semester, Year$.

$$\text{Hence, } \begin{Bmatrix} Room\_no, Days\_hours, \\ Semester, Year \end{Bmatrix}^{+} = \begin{Bmatrix} Room\_no, Days\_hours, Semester, \\ Year, Instructor\_ssn, Course\_no, \\ Sec\_no, Offering\_dept, \\ Credit\_hours, Course\_level \end{Bmatrix}$$

From the functional dependency

$$\begin{Bmatrix} Course\_no, Sec\_no, \\ Semester, Year \end{Bmatrix} \rightarrow \begin{Bmatrix} Days\_hours, Room\_no, \\ No\_of\_students, Instructor\_ssn \end{Bmatrix},$$

the attribute $No\_of\_students$ is added to the closure of $Room\_no, Days\_hours, Semester, Year$.

Hence, $\{Room\_no, Days\_hours, Semester, Year\}^+ = \{Room\_no, Days\_hours, Semester, Year, Instructor\_ssn, Course\_no, Sec\_no, Offering\_dept, Credit\_hours, Course\_level, No\_of\_students\}$

The closure of $\{Room\_no, Days\_hours, Semester, Year\}$ represents relation R.

Hence the keys of relation R are $\{Course\_no, Sec\_no, Semester, Year\}$ and $\{Room\_no, Days\_hours, Semester, Year\}$.

Comment

Step 5 of 5

The relation R is normalized as shown below:

The keys of relation R are $\{Course\_no, Sec\_no, Semester, Year\}$ and $\{Room\_no, Days\_hours, Semester, Year\}$.

According to second normal form, partial dependency should not exist in a relation.

$\{Course\_no\}$ is a partial key that functionally determines $Offering\_dept$, $Credit\_hours, Course\_level$.

Hence, the relation R is normalized into two relations R1 and R2 as shown below:

R1=$\{Course\_no, Offering\_dept, Credit\_hours, Course\_level\}$
The key of R1 is Course_no.

R2=$\{Room\_no, Days\_hours, Semester, Year, Instructor\_ssn, Course\_no, Sec\_no, No\_of\_students\}$
The keys of R2 are $\{Room\_no, Days\_hours, Semester, Year\}$ and $\{Course\_no, Sec\_no, Semester, Year\}$.

The relation R1 and R2 also satisfy third normal form as there exists no transitive dependency in relations R1 and R2.

Hence, the relation R1 and R2 are in third normal form.

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 oneitem, 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 RES look like? What will be its key? Show the FDs in this resulting relation. Is RES in 2NF? Is it in 3NF? Why or why not? (State assumptions, if you make any.)

Step 1 of 4

The natural join of two relations can be performed only when the relations have a common attribute with the same name.

The relations ORDER and ORDER_ITEM have O# as a common attribute. So, based on the attribute O#, the natural join of two relations ORDER and ORDER_ITEM can be performed.

The resulting relation RES when natural join is applied on relations ORDER and ORDER_ITEM is as follows:

O#	Odate	Cust#	Total_amount	I#	Qty_ordered	Total_price	Discount%

The key of the relation RES will be {O#,I#}.

Comment

Step 2 of 4

The functional dependencies in the relation RES are as given below:

$\{O\#\} \rightarrow \{Odate,\ Cust\#,\ Total\_amt\}$
$\{O\#,\ I\#\} \rightarrow \{Qty\_ordered,\ Total\_price,\ Discount\%\}$

Comment

Step 3 of 4

The relation RES is not in second normal form as partial dependencies exist in the relation.

- The key of the relation RES is {O#,I#}.
- O# is a partial primary key and it functionally determines Odate, Cust# and Total_amt%.

Step 4 of 4

According to the third normal form, the relation must be in second normal form and any non-key attribute should not describe any non-key attribute.

The relation RES is not in third normal form as it is not in second normal form.

14.30. Consider the following relation:
CAR_SALE(Car#, Date_sold, Salesperson#, Commission%, Discount_amt)
Assume that a car may be sold by multiple salespeople, and hence {Car#, Salesperson#} is the primary key. Additional dependencies are
Date_sold → Discount_amt and
Salesperson# → 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?

Step 1 of 4

The relation CAR_SALE is in first normal form (1NF) but not in second normal form.

• According to the first normal form, the relation should contain only atomic values.

• The primary key is {Car#, Salesperson#}.

• As the relation CAR_SALE contains only atomic values, the relation CAR_SALE is in the first normal form.

Comment

Step 2 of 4

The relation CAR_SALE is not in second normal form as partial dependencies exist in the relation.

• According to the second normal form, each non-key attribute must depend only on primary key.

• Salesperson# is a partial primary key and it functionally determines Commission%.

• As partial dependency exists in the relation, the relation CAR_SALE is not in second normal form.

• In order to satisfy second normal form, remove the partial dependencies by decomposing the relation as shown below:

CAR_SALE1(Car#, Date_sold, Salesperson#, Discount_amt)

CAR_SALE2 (Salesperson#, Commission%)

• The relations CAR_SALE1, and CAR_SALE2 are in second normal form.

Step 3 of 4

The relation CAR_SALE2 is in third normal form but the relation CAR_SALE1 is not in third normal form as transitive dependencies exist in the relation.

• According to the third normal form, the relation must be in second normal form and any non-key attribute should not describe any non-key attribute.

• In relations CAR_SALE1, Date_sold is a non-key attribute which functionally determines Discount_amt.

• As transitive dependency exists in the relation, the relation CAR_SALE1 is not in third normal form.

• In order to satisfy third normal form, remove the transitive dependencies by decomposing the relation CAR_SALE1as shown below:

CAR_SALE3 (Car#, Date_sold, Salesperson#)

CAR_SALE4 (Date_sold, Discount_amt)

• The relations CAR_SALE3 and CAR_SALE4 are now in third normal form.

Comment

Step 4 of 4

The final set of relations that are in third normal are as follows:

CAR_SALE2 (Salesperson#, Commission%)

CAR_SALE3 (Car#, Date_sold, Salesperson#)

CAR_SALE4 (Date_sold, Discount_amt)

14.31. Consider the following relation for published books:
BOOK (Book_title, Author_name, Book_type, List_price, Author_affil, Publisher)
Author_affil refers to the affiliation of author. Suppose the following dependencies exist:

Book_title → Publisher, Book_type
Book_type → List_price
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.

Step 1 of 4

a.

The relation Book is in first normal form (1NF) but not in second normal form.

Explanation:

- According to the first normal form, the relation should contain only atomic values.
- The primary key is (Book_Title, Author_Name).
- As the relation Book contains only atomic values, the relation Book is in the first normal form.
- According to the second normal form, each non-key attribute must depend only on primary key.
- Author_Name is a partial primary key and it functionally determines Author_affil.
- Book_title is a partial primary key and it functionally determines Publisher and Book_type.
- As partial dependency exists in the relation, the relation Book is not in second normal form.

Comment

Step 2 of 4

b.

The relation Book is in first normal form. It is not in second normal form as partial dependencies exist in the relation.

In order to satisfy second normal form, remove the partial dependencies by decomposing the relation as shown below:

Book_author (Book_title, Author_name)

Book_publisher(Book_title, Publisher, Book_type,

List_price)

Author(Author_name, Author_affil)

The relations Book_author, Book_publisher and Author are in second normal form.

Step 3 of 4

According to the third normal form, the relation must be in second normal form and any non-key attribute should not describe any non-key attribute.

• The relations Book_author and Author is in third normal form.

• The relations Book_publisher is not in third normal form as transitive dependency exists in the relation.

• Book_type is a non-key attribute which functionally determines List_price.

• In order to satisfy third normal form, remove the transitive dependencies by decomposing the relation Book_publisher as shown below:

Book_details(Book_title, Publisher,Book_type)

Book_price (Book_type, List_price)

The relations Book_author, Book_details, Book_price and Author are in third normal form.

Comment

Step 4 of 4

The final set of relations that are in third normal are as follows:

Book_author (Book_title, Author_name)

Book_details (Book_title, Publisher,Book_type)

Book_price (Book_type, List_price)

Author(Author_name, Author_affil)

14.32. This exercise asks you to convert business statements into dependencies. Consider the relation DISK_DRIVE (Serial_number, Manufacturer, Model, Batch, Capacity, Retailer). Each tuple in the relation DISK_DRIVE contains information about a disk drive with a unique Serial_number, made by a manufacturer, with a particular model number, released in a certain batch, which has a certain storage capacity and is sold by a certain retailer. For example, the tuple Disk_drive ('1978619', 'WesternDigital', 'A2235X', '765234', 500, 'CompUSA') specifies that WesternDigital made a disk drive with serial number 1978619 and model number A2235X, released in batch 765234; it is 500GB and 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 therefore 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 certain model of a particular manufacturer have exactly the same capacity.

Step 1 of 1

a)

manufacturer, serialNumber → model, batch, capacity, retailer

b)

model → manufacturer

c)

manufacturer, batch → model

d)

model → capacity

14.33. Consider the following relation:

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

In the above 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

Step 1 of 1

Let the relation R (Doctor#, Patient#, Date, Diagnosis, Treat_code , Change)

Functional dependencies of relation R is

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

{Treat_code}→{Charge}

Here there is no partial dependencies, So, the given relation is in 2NF. And it is not 3NF because the Charge is a nonkey attribute that is determined by another nonkey attribute, Treat_code.

We must decompose this as:

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 (Car_id, Option_type, Option_listprice, Sale_date, Option_discountedprice)
This relation refers to options installed in cars (e.g., cruise control) that were sold at a dealership, and the list and discounted prices of the options.
If CarID → Sale_date and Option_type → Option_listprice and CarID, Option_type → Option_discountedprice, 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 even in 2NF.

Step 1 of 3

The relation CAR_SALE is as shown below:

CAR_SALE(Car_id, Option_type, Option_listprice,

Sale_date, Option_discountedprice)

The functional dependencies are as given below:

Car_id Sale_date

Option_type Option_listprice

Car_id, Option_type Option_discountedprice

Comment

Step 2 of 3

In order for a relation to be in third normal form, all nontrivial functional dependencies must be fully dependent on the primary key and any non-key attribute should not describe any non-key attribute. In other words, there should not be any partial dependency and transitive dependency.

• For the relation CAR_SALE, Car_id, Option_type is a primary key.

• In functional dependency Car_id Sale_date, Car_id is a partial key that determines Sale_date. Hence, there exists partial dependency in the relation.

• In functional dependency Option_type Option_listprice, Option_type is a partial key that determines Option_type. Hence, there exists partial dependency in the relation.

Therefore, the relation CAR_SALE is not in third normal form.

Step 3 of 3

According to the second normal form, the relation must be in first normal form and each non-key attribute must depend only on primary key. In other words, there should not be any partial dependency.

- For the relation CAR_SALE, Car_id, Option_type is a primary key.
- In functional dependency Car_id Sale_date, Car_id is a partial key that determines Sale_date. Hence, there exists partial dependency in the relation.
- In functional dependency Option_type Option_listprice, Option_type is a partial key that determines Option_type. Hence, there exists partial dependency in the relation.

Therefore, the relation CAR_SALE is not in second normal form.

14.35. Consider the relation:
BOOK (Book_Name, Author, Edition, Year)
with the data:
Book_Name Author Edition Copyright_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. 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.

Step 1 of 3

Candidate Key

A candidate key may be a single attribute or a set of attribute that uniquely identify tuples or record in a database. Subset of candidate key are called prime attributes and rest of the attributes in the table are called non-prime attributes.

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

Book_Name is same in all rows so this can't be consider as a part of candidate key.

a.

Possible candidate keys:

(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.

Multi Valued Dependency (MVD):

MVD occurs when the presence of one or more tuples in the table implies the presence of one or more other rows in the same table. If at least two rows of table agree on all implying attributes, then there components might be swapped, and the resulting tuples must be in the table. MVD plays very important role in 4NF.

Consider the MVD $(Book\_Name) \rightarrow (Author) | (Edition, Year)$.

The relationship $(Book\_Name) \rightarrow (Author)$ indicates that the relationship between Book_Name and Author is independent of the relationship between Book_Name and (Edition, Copyright_Year).

By the definition of MVD, Book_Name is implying more than one Author and (Edition, Copyright_Year). If the components of Author, Edition and Copyright are swapped than the resulting rows would be present in the table. Therefore, the relation has MVD $(Book\_Name) \rightarrow (Author) | (Edition, Year)$.

Comment

Step 3 of 3

c.

Decomposition on the basis of MVD:

If a relation has MVD then redundant values will be there in the tuples and hence functional dependency would not exist in that relation. Therefore, the relation will be in BCNF. So relation can be decomposed into the following relations:

BOOK1 (Book_Name, Author, Edition)

BOOK2 (Edition, Copyright_Year)

Again BOOK1 is following MVD. Decompose it further and the final schema will be holding highest normal form.

BOOK1_1 (Book_Name, Author)

BOOK1_2 (Book_Name, Edition)

BOOK2 (Edition, Copyright_Year)

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.
b. Show how you will go about normalizing the relation.

Step 1 of 2

Relation TRIP has unique attribute Trip_id and particular Trip_id has single Start_date of the trip. So Start_date is fully functionally dependent on Trip_id.

a.

FDs and MVDs that exist in the relation are:

FD1: ($Trip\_id \rightarrow Start\_date$ **)**

Cities_visited and Cards_used may repeat for particular Start_date or Trip_id. Cities_visited and Cards_used are independent of each other and they also have multiple values. Also, both Cities_visited and Cards_used are dependent on Trip_id and Start_date, so the MVDs present in the relation are as follows:

MVD1: ($Trip\_id \rightarrow Cities\_visited \mid Cards\_used$)

MVD2: ($Start\_date \rightarrow Cities\_visited \mid Cards\_used$ **)**

Step 2 of 2

b.

Normalizing relation

Relation is having one FD and two MVDs, so first split the relation to remove functional dependency FD1.

TRIP1 (Trip_id, Start_date**)**

Now split relation to remove multi valued functional dependency. Cities_visited and Cards_used are independent of each other, if their components are swapped then relation will remain unchanged. On the basis of Start_date, the relation can be decomposed as follows:

TRIP2 (Start_date, Cities_visited)

TRIP3 (Start_date, Cards_used)

Following is the final schema for the table provided.

TRIP1 (Trip_id, Start_date**)**

TRIP2 (Start_date, Cities_visited)

TRIP3 (Start_date, Cards_used)