21.9 What is a serial schedule? What is a serializable schedule? Why is a serial schedule considered correct? Why is a serializable schedule considered correct?

Serial Schedule.
A schedule "S" is referred as serial, for each transaction "T" participating in schedule, the operations of T must be executed consecutively in schedule. So from this perspective, it is clear that only one transaction at a time is active and whenever if that transaction is committed, then it initiates the execution of next transaction.

Serializable schedule.
The schedule is referred as "serializable schedule. When a schedule t T be a set of n transactions (T_1, T_2, …, Tn), is serializable and if it is equivalent to n transactions executed serially.
Consider that possibly there are "n" serial schedule of "n" transactions and moreover there are possibly non-serial schedules. If two disjoined groups of the nonserial schedules are formed then it is equivalent to one or more of the serial schedules. Hence, the schedule is referred as serializable.

Reason for the correctness of serial schedule.
A serial schedule is said to be correct on the assumption of that each transactions is independent of each other. So according to the "consistency preservation" property, when the transaction runs in isolation, it is executed from the beginning to end from the other transaction .Thus, the output is correct on the database.
Therefore a set of transaction executed one at a time is correct.

Reason for the correctness of the serializable schedule.
The simple method to prove the correctness of serializable schedule is that to prove the satisfactory definition. In this definition, it compares the results of the schedules on the database, if both produce same final state of database. Then, two schedules are equivalent and it is proved to be serializable. Therefore, the serializable schedule is correct when the two schedules are in the same order.

21.22 Which of the following schedules is (conflict) serializable? For each serializable schedule, determine the equivalent serial schedules.

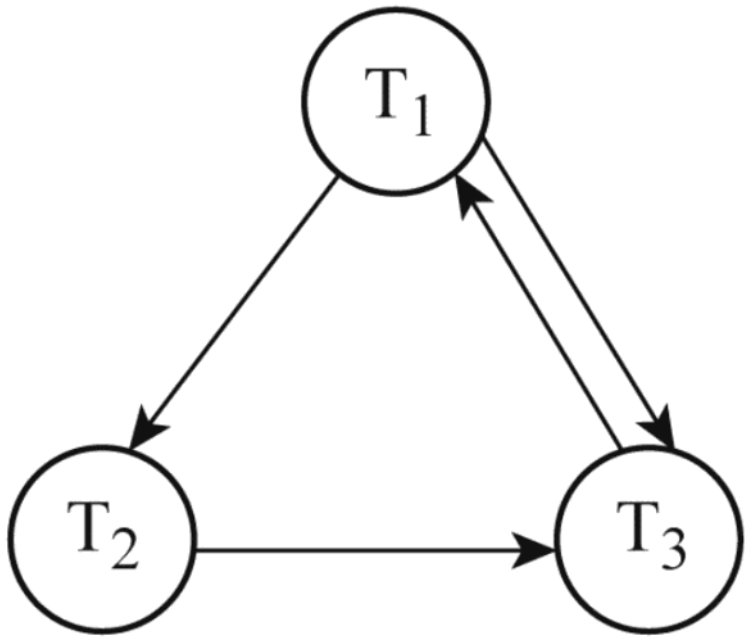
Serializable schedule.

A conflict graph corresponding to a schedule decides whether given schedule is conflict serializable or not. If conflict graph contains cycle, then the schedule is not serializable. The drawing sequence of conflict graph

1) Create a node labeled T_i in graph for each of the transaction T_i which participates in schedule S.
2) An edge is created from T_i to T_j in graph, where a write_item(X) is executed by T_i and then a read_item(X) is executed by T_j
3) Create an edge in graph from T_i to T_j , where a read_item(X) is executed by T_i and then a write_item(X) is executed by T_j
4) Create an edge in graph from T_i to T_j , where a write_item(X) is executed by T_i and then a write_item(X) is executed by T_j.
5) If no cycles are present in conflict graph, then it is a serializable schedule.

a. r1(X); r3(X); w1(X); r2(X); w3(X);

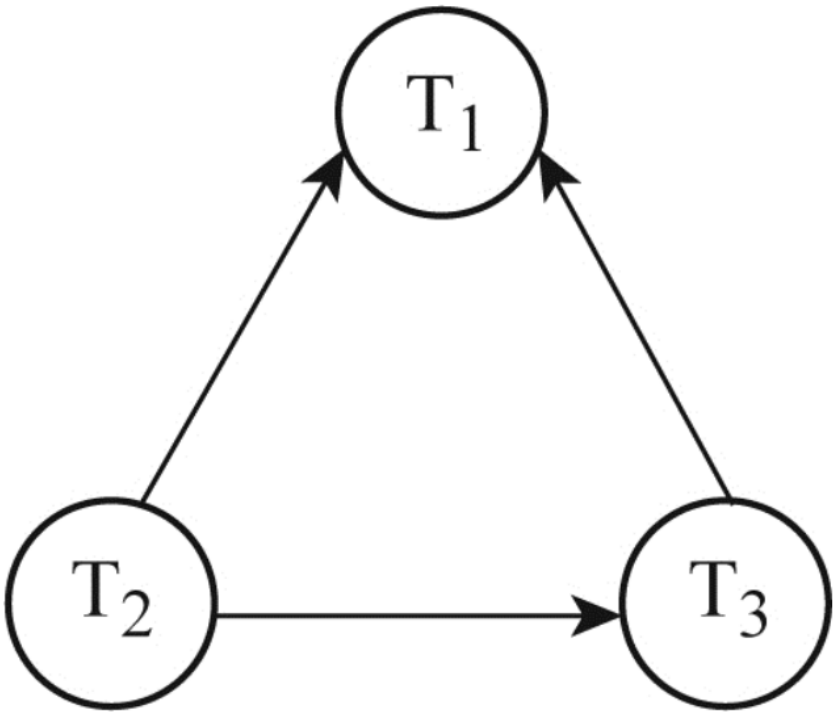
Conflict graph:



The conflict graph has cycle, in T_1-T_3. Hence, given schedule is not serializable .

c. r3(X); r2(X); w3(X); r1(X); w1(X);

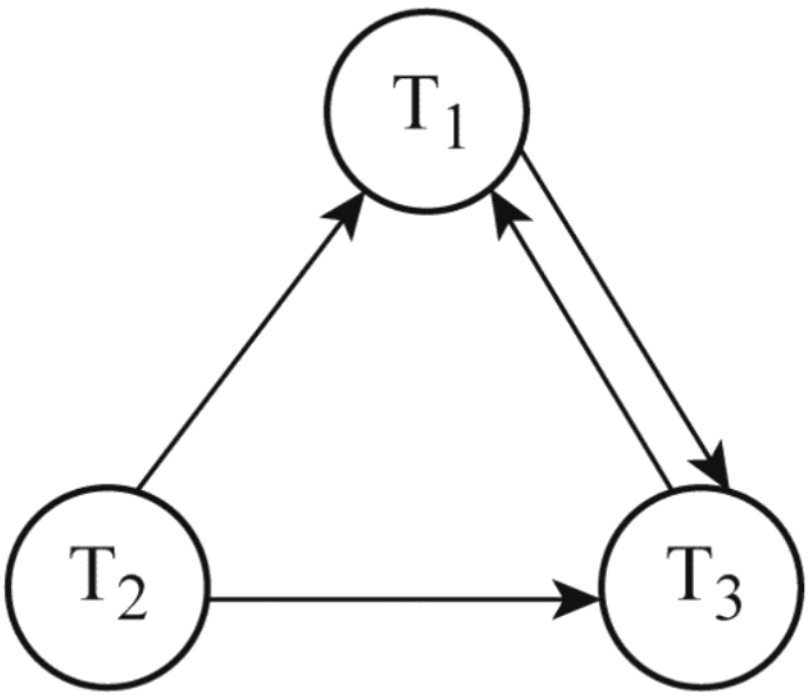
Conflict graph:



The graph contains no cycles. Hence, the schedule S is serializable .

d. r3(X); r2(X); r1(X); w3(X); w1(X);

Conflict graph:



The conflict graph has cycle, in T_1-T_3. Hence, the schedule S is not serializable.

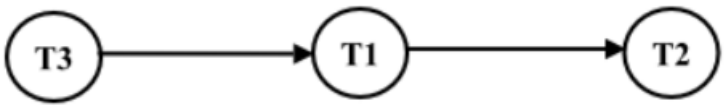
21.23 Consider the three transactions T1, T2, and T3, and the schedules S1 and S2 given below. Draw the serializability (precedence) graphs for S1 and S2 and state whether each schedule is serializable or not. If a schedule is serializable, write down the equivalent serial schedule(s).
T1: r1(X); r1(Z); w(X);
T2: r2(Z); r2(Y); w12(Z); w(Y);
T3: r3(X); r3(Y); w(Y);
S1: r1(X); r2(Z); r1(Z); r3(X); r3(Y); w1(X); w3(Y); r2(Y); w2(Z); w2(Y);
S2: r1(X); r2(Z); r3(X); r1(Z); r2(Y); r3(Y); w1(X); w2(Z); w3(Y); w2(Y);

The schedule S1 is as follows:

S1: r1(X); r2(Z); r1(Z); r3(X); r3(Y); w1(X); w3(Y); r2(Y); w2(Z); w2(Y);

T1	T2	T3
r1(X)		
	r2(Z)	
r1(Z)		
		r3(X)
		r3(Y)
w1(X)		
		w3(Y)
	r2(Y)	
	w2(Z)	
	w2(Y)	

The precedence graph for S1 is as follows:



The schedule S1 is a serializable schedule as there is no cycle in the precedence graph.
• T3 reads X before X is modified by T1.
• T1 reads Z before Z is modified by T2.
• T2 reads Y and writes it only after T3 has written to it.
The equivalent serializable schedule is as follows:

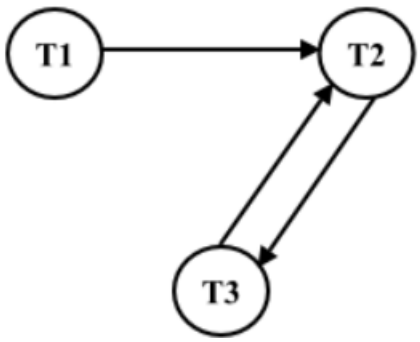
T1	T2	T3
r1(X)		
		r3(X)
	r2(Z)	
r1(Z)		
		r3(Y)
		w3(Y)
	r2(Y)	
w1(X)		
	w2(Y)	
	w2(Z)	

The schedule S2 is as follows:

S2: r1(X); r2(Z); r3(X); r1(Z); r2(Y); r3(Y); w1(X); w2(Z); w3(Y); w2(Y);

T1	T2	T3
$r_1(X)$		
	$r_2(Z)$	
		$r_3(X)$
$r_1(Z)$		
	$r_2(Y)$	
		$r_3(Y)$
$w_1(X)$		
	$w_2(Z)$	
		$w_3(Y)$
	$w_2(Y)$	

The precedence graph for S2 is as follows:



The schedule S2 is not a serializable schedule as there is cycle in the precedence graph.

- T2 reads Y before T3 reads it and modifies Y.
- T3 reads Y which is later modified by T2.

16.32 Consider the relation REFRIG(Model#, Year, Price, Manuf_plant, Color), which is abbreviated as REFRIG(M, Y, P, MP, C), and the following set F of functional dependencies:
F = { M →MP,{M, Y} →P, MP →C}
a. Evaluate each of the following as a candidate key for REFRIG, giving reasons why it can or cannot be a key: {M}, {M, Y}, {M, C}.
b. Based on the above key determination, state whether the relation REFRIG is in 3NF and in BCNF, giving proper reasons.
c. Consider the decomposition of REFRIG into D = {R1(M, Y, P), R2(M, MP,C)}. Is this decomposition lossless? Show why. (You may consult the testunder Property NJB in Section 16.2.4.)

(a)
- {M} IS NOT a candidate key since it does not functionally determine attributes Y or P.
- {M, Y} IS a candidate key since it functionally determines the remaining attributes P, MP, and C. i.e.
{M, Y} P, But M MP
By augmentation {M, Y} MP
Since MP C, by transitivity M MP, MP C, gives M C
By augmentation {M, Y} C
Thus {M, Y} P, MP, C and {M, Y} can be a candidiate key
- {M, C} IS NOT a candidate key since it does not functionally determine attributes Y or P.

(b)
REFRIG is not in 2NF, due to the partial dependency {M, Y} MP (since {M} MP holds). Therefore REFRIG is neither in 3NF nor in BCNF.
Alternatively: BCNF can be directly tested by using all of the given dependencies and finding out if the left hand side of each is a superkey (or if the right hand side is a prime attribute). In the two fields in REFRIG: M MP and MP C. Since neither M nor MP is a superkey, we can conclude that REFRIG is is neither in 3NF nor in BCNF.

(c)
R={M,Y,MP,C}
R1={M.Y,P}
R2={M,MP,C}
F={M,MP, {M,Y},P,MP,C},
F+ = { {M}+ {M,MP,C},{M,Y}+ {M,Y,MP,C},
{MP}+ {M[,C}}
R1 intersection R2 =M
R2- R1 ={MP,C}
D(R1,R2)has the lossless join property since Property :LJ1:FD((R1 intersection R2) (R2-R1)) is in F+ is satisfied (due to M {MP,C}).

15.2 Discuss insertion, deletion, and modification anomalies. Why are they considered bad? Illustrate With examples.

- 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.

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.

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.

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.

15.24 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.

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}

15.25 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} }.

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: 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: 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}.

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.

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,}
R6{A, D, G, H,}
The key for R3 is { A, D}.
R7{H, J}
The key for R7 is {H}.

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}

15.28 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?

$R = \{$ course no (A), sec-no (B), offering dept (C), credit – hours (D), course level (E), Instructor ssn (F), Semester (G), year (H), Days-hours (I), Room-no (J), no of students (K) $\}$

$\Rightarrow R = \{A, B, C, D, E, F, G, H, I, J, K\}$

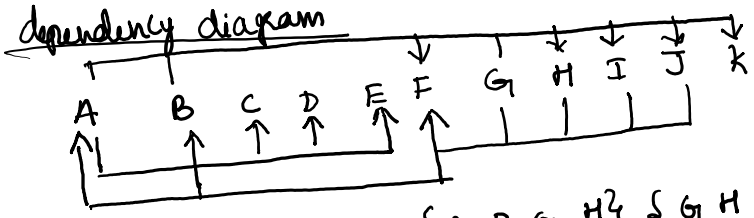
from given fds

$fd_1 = A \rightarrow \{c, d, e\}$

$fd_2 = \{A, B, g, h\} \rightarrow \{i, j, k, t\}$

$fd_3 = \{J, I, G, H\} \rightarrow \{F, A, B\}$.

dependency diagram



the keys of the R are :- $\{A, B, G, H\}, \{G\ H\ I\ J\}$

∴ candidate keys are

{course no, sec-no, semester, year} {

{ Semester, year, Days-hours, Room-no}

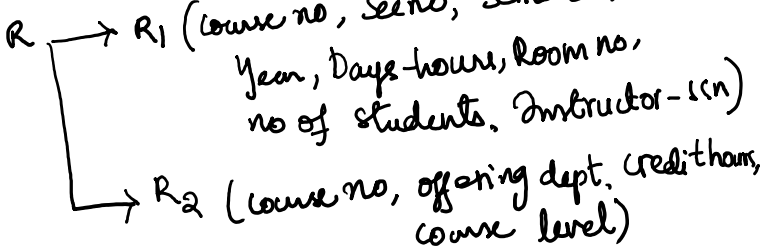
∴ there are six prime attributes. It is not in 2NF due to fd_1, so, decomposing it into relations than

$R \rightarrow R_1 (A\ B\ I\ J\ G\ H\ K\ F)$

$R_2 (A\ C\ D\ E)$

the above decomposition is also in 3NF as there is no transitive dependencies.

∴ Final relation are



15.30 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?

Given primary key is:
{Car#,Salesperson#}
Functional dependencies are:
Date_sold → Discount_amt
Salesperson# → Commission%
1NF:
A table is said to be in 1NF if it is free from multivalues and composite values.
So it is in 1NF
2NF:
A table is said to be in 2NF if it is free from partial dependencies.
partial dependency is nothing but a part of primary key deriving one or more attributes.
Here Salesperson# is part of primary key and it is deriving attribute commission%,so it has partial dependency.
In order to eliminate partial dependency,
We have to decompose tables:
{Salesperson#}+ = {Salesperson#,Commission%}
CAR_SALE(Car#, Date_sold, Salesperson#, Commission%, Discount_amt)
will be decomposed as
CAR_SALE(Car#, Date_sold, Salesperson#, Discount_amt)
Salesperson(Salesperson#, Commission%)
3NF:
A table is said to be in 3NF if it is free from transitive dependencies.
Transitive dependency is nothing but non key attribute deriving one or more attributes
Here Date_sold is non key attribute and it is deriving Discount_amt
So it is transitive dependency.
In order to eliminate transitive dependency,
We have to decompose tables:
{Date_sold}+ = {Date_sold,Discount_amt}
CAR_SALE(Car#, Date_sold, Salesperson#, Discount_amt)
will be decomposed as
CAR_SALE(Car#, Date_sold, Salesperson#)
Date_sold(Date_sold,Discount_amt);
Salesperson(Salesperson#, Commission%)
So tables after 3NF are:
CAR_SALE(Car#, Date_sold, Salesperson#)
Date_sold(Date_sold,Discount_amt);
Salesperson(Salesperson#, Commission%)

15.31 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.

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.

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.

a) 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.

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)

6.16 Specify the following queries on the COMPANYrelational database schema shown in Figure 5.5, using the relational operators discussed in this chapter. Also show the result of each query as it would apply to the database state in Figure 3.6.
a. Retrieve the names of all employees in department 5 who work more than 10 hours per week on the ProductX project.
b. List the names of all employees who have a dependent with the same first name as themselves.

d. For each project, list the project name and the total hours per week (by all employees) spent on that project.
e. Retrieve the names of all employees who work on every project.
f. Retrieve the names of all employees who do not work on any project.
g. For each department, retrieve the department name and the average salary of all employees working in that department.

i. Find the names and addresses of all employees who work on at least one project located in Houston but whose department has no location in Houston.
j. List the last names of all department managers who have no dependents.

a)
Retrieve the names of all employees in department 5 who work more than 10 hours per week on the ProductX project.
First we need to use the database called COMPANY. Since COMPANY is a relational database all he departments have each table represented as department1,department2, etc. So the query for selecting employee names(ename) from department5 is.

```
USE COMPANY;
SELECT first_name, last_name FROM department5
WHERE work > 10
```

b)
List the names of all employees who have a dependent with first name as themselves.

```
SELECT first_name, last_name , Count(*) AS CNT
FROM department5
GROUP BY first_name,
HAVING COUNT(*) > 1
```

It will select the full name of the employees having seimilar first name.

4.12 Specify the following queries in SQL on the database schema of Figure 1.2.
a. Retrieve the names of all senior students majoring in 'CS' (computer science).
b. Retrieve the names of all courses taught by Professor King in 2007 and 2008.
c. For each section taught by Professor King, retrieve the course number, semester, year, and number of students who took the section.
d. Retrieve the name and transcript of each senior student (Class = 4) majoring in CS. A transcript includes course name, course number, credit hours, semester, year, and grade for each course completed by the student.

a) The query to display the names of senior students majoring in CS is as follows:

Query:
SELECT Name FROM STUDENT
WHERE Major = "CS" AND Class = "4";

Output:
-

Explanation:
• There are no rows in the database where Class is Senior, and Major is CS.
SELECT is used to query the database and get back the specified fields.
o Name is the columns of STUDENT table.
FROM is used to query the database and get back the preferred information by specifying the table name.
o STUDENT is a table name.
WHERE is used to specify a condition based on which the data is to be retrieved. In the database, Seniors are represented by Class 4. The condition is as follows:
o Major='CS'AND Class = '4'

b)
The query to get the course name that are taught by professor King in year 2007 and 2008 is as follows:
Query:
SELECT Course_name
FROM COURSE, SECTION
WHERE COURSE.Course_number = SECTION.Course_number
AND Instructor = 'King'
AND (Year='07' or Year='08');

Output:

Course Name
Discrete Mathematics

Explanation:

SELECT is used to query the database and get back the specified fields.
o Course_name is the columns of COURSE table.
FROM is used to query the database and get back the preferred information by specifying the table name.
o COURSE, SECTION are table names.
WHERE is used to specify a condition based on which the data is to be retrieved. The conditions are as follows:
o COURSE.Course_number = SECTION.Course_number
o Instructor = 'King'
o (Year='07' or Year='08')
• The conditions are concatenated with AND operator. All the conditions must be satisfied.

c)
The query to retrieve the course number, Semester, Year and number of students who took the section taught by professor King is as follows:

Query:

```
SELECT Course_number, Semester, Year, Count(G.Student_number) AS 'Number of Students'
FROM SECTION AS S, GRADE_REPORT AS G
WHERE S.Instructor= 'King'
AND S.Section_identifier=G.Section_identifier;
```

Output:

Course Number	Semester	Year	Number of students
Math2410	Fall	7	1

Explanation:

SELECT is used to query the database and get back the specified fields.
o Course_number, Semester, Year are the columns of SECTION table.
FROMis used to query the database and get back the preferred information by specifying the table name.
o GRADE_REPORT, SECTION are table names.
WHERE is used to specify a condition based on which the data is to be retrieved. The conditions are as follows:
o S.Instructor= 'King'
o S.Section_identifier=G.Section_identifier

d)
The query to display the name and transcript of each senior students majoring in CS is as follows:
Query:

```
SELECT ST.Name, C.Course_name, C.Course_number, C.Credit_hours, S.Semester, S.Year, G.Grade
FROM STUDENT AS ST, COURSE AS C, SECTION AS S, GRADE_REPORT As G
WHERE Class = 4 AND Major='CS'
AND ST.Student_number= G.Student_number
AND G.Section_identifier= S.Section_identifier
AND S.Course_number= C.Course_number;
```

Explanation
SELECT is used to query the database and get back the specified fields.
Course_number, course_number, credit_hours are the columns of COURSE table
o Semester, Year are the columns of SECTION table.
o Name is the columns of STUDENT table.
o Grade is the columns of GRADE_REPORT table.
FROM is used to query the database and get back the preferred information by specifying the table name.
o STUDENT, COURSE, GRADE_REPORT, SECTION are table names.
o ST is the alias name for STUDENT table.
o G is the alias name for GRADE_REPORT table.
o S is the alias name for SECTION table.
o C is the alias name for COURSE table.
WHERE is used to specify a condition based on which the data is to be retrieved. The conditions are as follows:
o Class = 4
o Major='CS'
o ST.Student_number= G.Student_number
o G.Section_identifier= S.Section_identifier
o S.Course_number= C.Course_number

4.13

Write SQL update statements to do the following on the database schema shown in Figure 1.2.

a. Insert a new student, <‘Johnson’, 25, 1, ‘Math’>, in the database.

b. Change the class of student ‘Smith’ to 2.

c. Insert a new course, <‘Knowledge Engineering’, ‘CS4390’, 3, ‘CS’>.

a.

```
insert into student
values('johnson',25,1,'Math');
```

b.

```
update student
set class=2
where sname='Smith';
```

c.

```
insert into course
values('Knowledge Engineering','CS4390',3,'CS');
```