11. Version A

Given the following schedule S:

R1(x), W1(x), R2(x), R3(z), W1(y), R2(y), W3(y), R4(x), R4(z)

Explain whether this schedule is conflict serializable. If the schedule is conflict equivalent to a serial schedule, write an equivalent serial order.

Solution:

There are 5 conflicts in this case.

First conflict is between W1(x) and R2(x). Because of this conflict we need to put T1 before T2 i.e T1 --->T2.

Second conflict is between W1(x) and R4(x). Because of this conflict we need to put T1 before T4 i.e T1 --->T4.

Third conflict is between W1(y) and R2(y). Because of this conflict we need to put T1 before T2 i.e T1 --->T2

Fourth conflict is between W1(y) and W3(y). Because of this conflict we need to put T1 before T3 i.e T1 --->T3

Fifth conflict is between R2(y) and W3(y). Because of this conflict we need to put T2 before T3 i.e T2 --->T3

Since we have no cycle, so this schedule is conflict serializable. An equivalent serial schedule will be one of the following:

T1, T4, T2, T3

T1, T2, T4, T3

T1, T2, T3, T4

Q11: Version B

Given the following schedule S:

R1(x), W1(z), R2(y), R1(y), R3(z), W4(z), W2(x), R3(y), W2(y), W4(x)

Explain whether this schedule is conflict serializable. If the schedule is conflict equivalent to a serial schedule, write an equivalent serial order.

Solution:

There are 8 conflicts in this case.

First conflict is between R1(x) and W2(x). Because of this conflict we need to put T1 before T2 i.e T1 --->T2.

Second conflict is between R1(x) and W4(x). Because of this conflict we need to put T1 before T4 i.e T1 --->T4.

Third conflict is between W2(x) and W4(x). Because of this conflict we need to put T2 before T4 i.e T2 --->T4.

Fourth conflict is between W1(z) and R3(z). Because of this conflict we need to put T1 before T3 i.e T1 --->T3

Fifth conflict is between W1(z) and W4(z). Because of this conflict we need to put T1 before T4 i.e T1 --->T4

Sixth conflict is between R3(z) and W4(z). Because of this conflict we need to put T3 before T4 i.e T3 --->T4

Seventh conflict is between R1(y) and W2(y). Because of this conflict we need to put T1 before T2 i.e T1 --->T2

Eighth conflict is between R3(y) and W2(y). Because of this conflict we need to put T3 before T2 i.e T3 --->T2

Since we have no cycle, this schedule is conflict serializable. The equivalent serial schedule is:

T1, T3, T2, T4

12. Version A

The crew members who will be onboard a flight are recorded in an un-normalised relation with the following schema:

Crew (flightCode, destination, date, tailCode, acName, airline, emp_id, name, title)

For example, the row ('NZ700', 'Auckland', '31-10-2021', 'VH-EBU', 'City of Broken Hill', 'Air New Zealand', 78354, 'Roger Wilco', 'Navigator') indicates that on 31st October 2021, flight NZ700 departed for Auckland on Air New Zealand aircraft VH-EBU ("City of Broken Hill") with navigator Roger Wilco (employee ID 78354) on board. Note that tailCode is the government-issued registration number used to identify an aircraft.

You are told that the following functional dependencies apply in this table.

flightCode --> destination

flightCode, date --> tailCode

tailCode --> acName, airline

emp_id --> name, title

flightCode	**destination**	**date**	**tailCode**	**acName**	**airline**	**emp_id**	**name**	**title**
NZ700	Auckland	31-10-2021	VH-EBU	City of Broken Hill	Air New Zealand	78354	Roger Wilco	Navigator
CX110	Hong Kong	1-11-2021	P2-ANH	City of Broken Hill	Cathay Pacific	78354	John Smith	Navigator
NZ700	Auckland	31-10-2021	VH-EBU	City of Broken Hill	Air New Zealand	78965	John Smith	Navigator

a) Given the relation instance above, list all those functional dependencies (listed above) that are violated.

b) Find a key to the relation above.

c) Is the relation in BCNF? If not, give a lossless-join and dependency preserving decomposition of the table Crew into a set of BCNF relations. Show that your final relations are in BCNF, and no FDs have been lost.

Solution:

a) emp_id --> name, title is violated by emp_id = 78354, which does not functionally determine name (it has two names, Roger Wilco and John Smith) in row 1 and row 2.
b) The full set of attributes can be obtained from the closure of (flightCode, date, emp_id) which is the key.
c) The relation is not in BCNF. flightCode --> destination violates BCNF.
A lossless and dependency preserving decomposition in BCNF is:

Crew (flightCode, date, emp_id)
Destination (flightCode, destination)
Flight (flightCode, date, tailCode)
Plane (tailCode, acName, airline)
Employee (emp_id, name, title)

All relations above have only functional dependencies of X -> Y where X can only be a superkey. This decomposition is also dependency preserving because no FDs have been lost in the decomposition.

Q12: Version B

The passengers who will be travelling in a flight are recorded in a relation with the following schema:

Passenger (flightNum, destination, date, tailNum, acName, airline, ticketNum, name, title)

For example, the row ('NZ800', 'Wellington', '31-10-2021', 'VH-EBU', 'City of Broken Hill', 'Air New Zealand', 95346, 'Phil James, 'Doctor) indicates that on 31st October 2021, flight NZ800 departed for Wellington on Air New Zealand aircraft VH-EBU ("City of Broken Hill") with Doctor Phil James (ticketNum 95346) on board. Note that tailNum is the government-issued registration number used to identify an aircraft.

You are told that the following functional dependencies apply in this table.

flightNum --> destination

tailNum --> acName, airline

flightNum, date --> tailNum

ticketNum --> name, title

flightNum	**destination**	**date**	**tailNum**	**acName**	**airline**	**ticketNum**	**name**	**title**
NZ800	Wellington	31-10-2021	VH-EBU	City of Broken Hill	Air New Zealand	95346	Phil James	Doctor
CX120	Hong Kong	1-11-2021	P2-ANH	City of Broken Hill	Cathay Pacific	95346	Marlon Dumas	Doctor
NZ800	Wellington	31-10-2021	VH-EBU	City of Broken Hill	Air New Zealand	85626	Marlon Dumas	Doctor

a) Given the relation instance above, list all those functional dependencies (listed above) that are violated.

b) Find a key to the relation above.

c) Is the relation in BCNF? If not, give a lossless-join and dependency preserving decomposition of the table Crew into a set of BCNF relations. Show that your final relations are in BCNF, and no FDs have been lost.

Solution:

a) ticketNum --> name, title is violated by ticketNum = 95346, which does not functionally determine name (it has two names, Phil James and Marlon Dumas) in row 1 and row 2.
b) The full set of attributes can be obtained from the closure of (flightNum, date, ticketNum) which is the key.
c) The relation is not in BCNF. flightCode --> destination violates BCNF.
A lossless and dependency preserving decomposition in BCNF is:

Passenger (flightNum, date, ticketNum)
Destination (flightNum, destination)
Flight (flightNum, date, tailNum)
Plane (tailNum, acName, airline)
Person (ticketNum, name, title)

All relations above have only functional dependencies of X -> Y where X can only be a superkey. This decomposition is also dependency preserving because no FDs have been lost in the decomposition.

13. Version A

Assume that there are 10,000 records in Product(product_id, name, price, manufacturer_id) table and each record is 200 bytes long. Each page is 4K bytes, of which 196 bytes are reserved for header and array of record pointer. How many pages do we need to store this Product table?

We also have another table called Manufacturer(manufacturer_id, address, owner) which has 2000 records and all those records are stored in 5 pages.

If we use Block-Nested loop join for the following query then how many disk I/0 do we need? Consider product as the outer table in this case.

```
SELECT *

FROM Product P, Manufacturer M

WHERE P.manufacturer_id = M.manufacturer_id
```

Solution:

Empty space in each page is (4*1024 – 196) = 3900 bytes

Number of records per page = floor(3900 / 200) = 19 records

Number of pages required to store the Product table = ceil(10,000/19)) = 527 pages

Required disk I/O = $\text{Pages}_{\text{product}} + \text{Pages}_{\text{product}} * \text{Pages}_{\text{manufacturer}}$

$= 527 + 527*5$

$= 3162$

Q13: Version B

Assume that there are 10000 records in Product(product_id, name, price, manufacturer_id) table and each record is 200 bytes long. Each page is 4K bytes, of which 196 bytes are reserved for header and array of record pointer. How many pages do we need to store this Product table?

We also have another table called Manufacturer(manufacturer_id, address, owner) which has 2000 records and all those records are stored in 5 pages.

If we use Nested loop join for the following query, then how many disk I/0 do we need? Consider Manufacturer as the outer table in this case.

```
SELECT *

FROM Manufacturer M, Product P

WHERE M.manufacturer_id = P.manufacturer_id
```

Solution:

Empty space in each page is (4*1024 – 196) = 3900 bytes

Number of records per page = floor(3900 / 200) = 19 records

Number of pages required to store the Product table = ceil(10,000/19)) = 527 pages

Required disk I/O $= \text{Pages}_{\text{manufacturer}} + \text{Records}_{\text{manufacturer}} * \text{Pagesproduct}$

$= 5 + 2000*527$

$= 1,054,005$