Sample Exam (2019) for CSE 480: Database Systems KEY

Instructions:

• DO NOT START THE EXAM UNTIL TOLD TO DO SO

• You need to answer 6 of the 7 questions.

• On one of the questions make a large slash through it, which indicates that it should not be graded.

• Answer the questions in the spaces provided on the page. If you run out of room for an answer, continue on the back of that page.

• Show us the crossed out question and present your MSU ID (or other form of ID) when you hand in your exam.

Figure 1: http://xkcd.com/1129/
Question 1: SQL statement.......................................................50 points

I have 2 tables containing information about athletes and teams. The table *athletes* has the following columns: name (the athlete's name, unique), year (the graduating year of the athlete), team (the id of the team the athlete is part of), and grade (the athlete's GPA). The table *teams* has the following columns: name (the team's name), id (the team's identification number, unique), and school (the name of the team's school).

I want to know (for each team) what the was the average grade of the students who played for that team. However, I am only concerned with teams with more than one player. Order the result from the highest average grade to the least.

Here is some example data:

<table>
<thead>
<tr>
<th>Table: <em>athletes</em></th>
<th>name</th>
<th>year</th>
<th>team</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Josh</td>
<td>2017</td>
<td>7</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>Emily</td>
<td>2016</td>
<td>7</td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td>Tyler</td>
<td>2015</td>
<td>8</td>
<td>3.5</td>
<td></td>
</tr>
<tr>
<td>Grant</td>
<td>2015</td>
<td>8</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>Charles</td>
<td>2016</td>
<td>9</td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td>Laura</td>
<td>2016</td>
<td>10</td>
<td>1.0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table: <em>teams</em></th>
<th>name</th>
<th>id</th>
<th>school</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparties</td>
<td>7</td>
<td>MSU</td>
<td></td>
</tr>
<tr>
<td>Greens</td>
<td>8</td>
<td>MSU</td>
<td></td>
</tr>
<tr>
<td>Blues</td>
<td>9</td>
<td>UM</td>
<td></td>
</tr>
<tr>
<td>Whites</td>
<td>10</td>
<td>MSU</td>
<td></td>
</tr>
</tbody>
</table>

And the desired results for this example data:

<table>
<thead>
<tr>
<th>team</th>
<th>average grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparties</td>
<td>3.5</td>
</tr>
<tr>
<td>Greens</td>
<td>2.75</td>
</tr>
</tbody>
</table>

Fill in the blanks when the clause should be used in the SELECT statement (you must always use qualified names):

```
SELECT       teams.name,  avg(athletes.grade)
FROM         athletes
             INNER JOIN teams
ON           athletes.team = teams.id
WHERE
GROUP BY     teams.id
HAVING       count() > 1
ORDER BY     avg(athletes.grade) DESC
```
Question 2: Corrupted Database ........................................50 points

Unfortunately, Pikachu got a bit upset too close to our database, and it erased some of
the values. The data isn’t essential, but we would like to recover what we can. Here’s
what we know:

- The relation has 6 attributes (A-F) as shown below.
- The relation has one multivalued dependency: $AB \rightarrow F$
- The relation has two functional dependencies: $C \rightarrow D$
- and $AC \rightarrow E$

Use the data given in the relation below to fill in the missing values. If the value is
unknown, put a question mark (?) instead of a number.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>9</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>
Question 3: Schedules ................................................................. 50 points

(a) (10 points) What is the difference between a serial and a serializable schedule?

**Solution:** Serial schedules don’t intersperse actions from different transactions. Serializable schedules can, but the final result will be the same as some ordering of a serial schedule.

(b) (10 points) Which of the four ACID principles are violated by a non-serializable schedule?

○ Atomicity  ○ Consistency  √ **Isolation**  ○ Durability

(c) (10 points) When does a conflict occur between two transactions?

**Solution:** A conflict occurs between two actions if both transactions are involved with the same element in the database and one of them is a write.

(d) (10 points) Define what conflict-serializable means.

**Solution:** Two schedule are called conflict-equivalent if one can be transformed into the other by a sequence of non-conflicting swaps of adjacent actions. Conflict-serializable means that a schedule is conflict-equivalent to a serial schedule.

(e) (10 points) Are all serializable schedules conflict-serializable?

**Solution:** No, all conflict-serializable schedules are serializable by definition, but not all serializable schedules are conflict-serializable. If there are arithmetic coincidences, then a serializable schedule may not be conflict-serializable.
Question 4: Precedence ............................................................... 50 points

(a) (10 points) What are the 3 conditions that determine if an action \((A_1)\) from one transaction takes precedence over an action \((A_2)\) in a different transaction?

Solution:

1. \(A_1\) is ahead of \(A_2\) in the schedule
2. Both \(A_1\) and \(A_2\) involve the same database element
3. At least one of \(A_1\) and \(A_2\) is a write action

(b) (20 points) What are the conflicts and implied transaction precedence for the following schedule:
\[ S: r_1(A); r_2(B); w_1(A); w_2(A); w_3(A); w_3(B); r_1(B); \]

Solution:

\[ r_1(A); w_2(A); T_1 < T_2 \]
\[ r_1(A); w_3(A); T_1 < T_3 \]
\[ r_2(B); w_3(B); T_2 < T_3 \]
\[ w_1(A); w_2(A); T_1 < T_2 \]
\[ w_1(A); w_3(A); T_1 < T_3 \]
\[ w_2(A); w_3(A); T_2 < T_3 \]
\[ w_3(B); r_1(B); T_3 < T_1 \]

(c) (20 points) Draw the Precedence Graph for the schedule and indicate if it is conflict-serializable.

Solution:
Not conflict-serializable (cycles present).
Question 5: Simple Locks................................................................. 50 points

(a) (10 points) According to two-phase locking, when can locks not be acquired?
   - Before all read and write actions
   - Before the last read or write action
   - After the first unlock action
   √ Only upon commit or rollback

(b) (10 points) According to strict two-phase locking, when can unlocks be performed?
   - Before all read and write actions
   - Before the last read or write action
   - After the first unlock action
   √ Only upon commit or rollback

(c) (30 points) For this question there is only one type of lock (an exclusive lock).
   For the following schedule (S), output all of the read and write actions with the
   needed lock and unlock actions (i.e. \( l_1(A) \) and \( u_1(A) \)). Only lock/unlock when
   such actions are required. You can assume a commit occurs when a transaction has
   completed all of its actions.

   S: \( r_1(A); w_1(B); w_2(C); r_1(A); w_2(A); r_3(B) \);

   **Solution:**

   1. \( l_1(A) \);
   2. \( r_1(A) \);
   3. \( l_1(B) \);
   4. \( w_1(B) \);
   5. \( l_2(C) \);
   6. \( w_2(C) \);
   7. \( r_1(A) \);
   8. \( u_1(A) \);
   9. \( l_2(A) \);
   10. \( w_2(A) \);
   11. \( u_1(B) \);
   12. \( l_3(B) \);
   13. \( r_3(B) \);
   14. \( u_2(A) \);
   15. \( u_2(C) \);
   16. \( u_3(B) \);

   Note there are some minor swaps that would also result in a correct answer.
Question 6: Multiple Types of Locks .................................................................... 50 points

Below are three transactions (and six total actions). Unlocks must happen after all the actions in a transaction have taken place (strict two-phase locking). For this problem, there are shared locks \( sl_1(A) \) and exclusive locks \( xl_1(A) \) and either/both are unlocked with \( u_1(A) \). Below, the order of each of the actions are interleaved. For each part, output the necessary locks and unlocks that the transaction should perform to make the action take place (don’t forget to include the action itself).

<table>
<thead>
<tr>
<th>( T_1 )</th>
<th>( T_2 )</th>
<th>( T_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_1(X) )</td>
<td>( w_2(Y) )</td>
<td>( w_3(Z) )</td>
</tr>
<tr>
<td>( w_1(Y) )</td>
<td>(   )</td>
<td>( w_3(Y) )</td>
</tr>
<tr>
<td>( w_1(X) )</td>
<td>(   )</td>
<td>(   )</td>
</tr>
</tbody>
</table>

(a) \( T_1: r_1(X) \)

**Solution:** \( sl_1(X); r_1(X); \)

(b) \( T_2: w_2(Y) \)

**Solution:** \( xl_2(Y); w_2(Y); u_2(Y); \)

(c) \( T_1: w_1(Y) \)

**Solution:** \( xl_1(Y); w_1(Y); \)

(d) \( T_3: w_3(Z) \)

**Solution:** \( xl_3(Z); w_3(Z); \)

(e) \( T_1: w_1(X) \)

**Solution:** \( xl_1(X); w_1(X); u_1(X); u_1(Y); \)

(f) \( T_3: w_3(Y) \)

**Solution:** \( xl_3(Y); w_3(Y); u_3(Z); u_3(Y); \)
Question 7: Transaction Modes ........................................ 50 points

After the statement is completed by the associated connection, write which connections are holding each type of lock in the table. If no connection holds a type of lock, leave it blank. A connection can only hold one lock at a time. If an action can’t be granted the necessary locks, cross out that action and proceed as if that action wasn’t in the schedule (don’t perform a rollback).

<table>
<thead>
<tr>
<th>ID</th>
<th>SQL statement</th>
<th>Shared</th>
<th>Reserved</th>
<th>Exclusive</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CREATE TABLE ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>BEGIN TRANSACTION;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>INSERT INTO ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SELECT ...</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>BEGIN IMMEDIATE TRANSACTION;</td>
<td>1 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>DELETE FROM ...</td>
<td>1 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ROLLBACK TRANSACTION;</td>
<td>1 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BEGIN TRANSACTION;</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>BEGIN TRANSACTION;</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>COMMIT TRANSACTION;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SELECT ...</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SELECT ...</td>
<td>2 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>INSERT ...</td>
<td>2 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>INSERT ...</td>
<td>2 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>COMMIT TRANSACTION;</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>COMMIT TRANSACTION;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>BEGIN EXCLUSIVE TRANSACTION;</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SELECT ...</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>UPDATE ...</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>COMMIT TRANSACTION;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question 8: Deadlocks

Our database is running into problems, as many transactions are waiting for locks held by other transactions. Here is what each transaction is waiting for:

- $T_1$ is waiting on $T_4$
- $T_2$ is waiting on $T_7$
- $T_3$ is waiting on $T_2$
- $T_4$ is waiting on $T_1$
- $T_5$ is waiting on $T_8$
- $T_6$ is waiting on $T_2$
- $T_7$ is waiting on $T_6$
- $T_8$ is not waiting

(a) (20 points) Draw the Wait-For graph for transactions $T_1$–$T_8$.

Solution:

(b) (10 points) Which transactions are deadlocked?

\[ \checkmark T_1 \quad \checkmark T_2 \quad \checkmark T_3 \quad \checkmark T_4 \quad \quad \circ T_5 \quad \checkmark T_6 \quad \checkmark T_7 \quad \circ T_8 \]

(c) (20 points) What needs to be done to resolve the deadlock?

Solution: Either $T_1$ or $T_4$ needs to be rolled back. And one of $T_2$, $T_6$, or $T_7$ needs to be rolled back.
Question 9: Deadlock Resolution ................................................. 50 points

Below is a time line as to when transactions were started, received locks, and requested a lock. For this problem you can assume all locks are exclusive. You can also assume transaction restarts are handled after the events described.

1. $T_W$ starts
2. $T_W$ gets lock on A
3. $T_X$ starts
4. $T_X$ gets lock on B
5. $T_W$ wants lock on B, waits on $T_X$
6. $T_Z$ starts
7. $T_Z$ gets lock on C
8. $T_Z$ wants lock on B, rollback!

(a) (10 points) Which deadlock resolution strategy is being used?

√ Wait-Die  ○ Wound-Wait

(b) (20 points) If the same events occurred using the other deadlock resolution strategy, what would the time line be?

Solution:

<table>
<thead>
<tr>
<th>Step</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$T_W$ starts</td>
</tr>
<tr>
<td>2</td>
<td>$T_W$ gets lock on A</td>
</tr>
<tr>
<td>3</td>
<td>$T_X$ starts</td>
</tr>
<tr>
<td>4</td>
<td>$T_X$ gets lock on B</td>
</tr>
<tr>
<td>5</td>
<td>$T_W$ wants lock on B, waits on $T_X$</td>
</tr>
<tr>
<td>6</td>
<td>$T_Z$ starts</td>
</tr>
<tr>
<td>7</td>
<td>$T_Z$ gets lock on C</td>
</tr>
<tr>
<td>8</td>
<td>$T_Z$ wants lock on B, rollback!</td>
</tr>
</tbody>
</table>

(c) (10 points) Before the rollback on the original Step 8, draw the Wait-For Graph for the transactions.

Solution:

```
$T_W$ -> $T_X$ -> $T_Z$
```

(d) (10 points) If you want to have fewer rollbacks (but perhaps undoing more work) which strategy should you use?

○ Wait-Die  √ Wound-Wait
Question 10: Optimistic Scheduling........................................50 points
There are two transactions ($T_i$ and $T_j$). $T_i$ started before $T_j$. Both $T_i$ and $T_j$ performing reads and/or writes on database element $E$.

(a) (10 points) Give an example of a Read-too-late event and why it is a problem.

**Solution:** $T_i$ tries to read element $E$, but $T_j$ wrote to element $E$ first. Because $T_i$ started first it should have read the old value of $E$, but it can’t because $T_j$ changed it.

(b) (10 points) Give an example of a Write-too-late event.

**Solution:** $T_i$ tries to write element $E$, but $T_j$ read from the element $E$ first. Because $T_i$ started first it should have changed the old value of $E$, before $T_j$ had the opportunity to read it. Now $T_j$ has read the wrong (old) value.

(c) (10 points) Why are physically unrealizable behaviors a problem?

**Solution:** Because they result in non-serializable schedules and hence break isolation.

(d) (10 points) What is the Thomas Write Rule?

**Solution:** If a transaction needed to write to an element that has already been written to by a transaction with a later time stamp, then the write doesn’t have to take place. It doesn’t matter because it would have been overwritten anyways.

(e) (10 points) When does an optimistic scheduler (time stamp/validating) outperform a pessimistic scheduler (locking)?

**Solution:** If conflicts between transactions are rare (mostly reads or transactions which touch different parts of the database) than optimistic schedulers will have fewer transaction delays. However, it may perform more rollbacks, so rollback performance should not be a factor.
Question 11: Legal Optimistic Schedules ........................................ 50 points

Below is a time line of when two transactions ($T_i$ and $T_j$) performed actions on the database.

1. $T_i$ begins transaction
2. $r_i(A)$
3. $T_j$ begins transaction
4. $w_i(B)$
5. $w_j(A)$
6. $T_i$ commits transaction
7. $T_j$ commits transaction

For proposed action, indicate if that action were added (in isolation from the other proposed actions) would the resulting schedule result in physically unrealizable behavior.

(a) $w_i(A)$ between steps 3 and 4 √ Legal ○ Physically Unrealizable
(b) $w_j(B)$ between steps 3 and 4 ○ Legal √ Physically Unrealizable
(c) $w_j(B)$ between steps 4 and 5 ○ Legal ○ Physically Unrealizable
(d) $r_j(B)$ between steps 4 and 5 ○ Legal ○ Physically Unrealizable
(e) $w_j(C)$ between steps 4 and 5 ○ Legal ○ Physically Unrealizable
(f) $r_i(A)$ between steps 5 and 6 ○ Legal √ Physically Unrealizable
(g) $w_i(A)$ between steps 5 and 6 ○ Legal √ Physically Unrealizable
(h) $r_j(A)$ between steps 6 and 7 ○ Legal ○ Physically Unrealizable
We are creating a database to record which farms raise which types of animals. Below are some facts that need to be represented in the database:

- Each farm has a name and a unique address
- Each farm may raise multiple animals, and each animal can be uniquely identified by the combination of its species and id number.
- Each animal is raised by one farm.
- Each animal also has a nickname, but this may not be unique to each.
- Some animals are breeders, meaning that we know its sex and the season in which it can be breed.
- Some animals are producers, meaning that they produce one or more products (e.g. eggs, milk, wool).
- The products that can be produced have a unique id, as well as, a name and price.

Draw an E/R diagram illustrating the structure of a database capturing the above information.

Solution:

The weird arrow (between raises and Farm) is suppose to be a rounded arrow (representing exactly one).
Question 13: Converting E/R Diagram .................................................. 50 points

Below is an E/R representing information about pets and their owners.

(a) (40 points) Write a relational database schema representing the E/R diagram. Be sure to combine relations when possible. Use the Object-Oriented method to create relations for the subclasses.

Solution:

- Pets(Name, Age, AdoptionDate, OwnerFirstName, OwnerLastName)
- Purebred(Name, Age, AdoptionDate, OwnerFirstName, OwnerLastName, Registration)
- Trained(Name, Age, AdoptionDate, OwnerFirstName, OwnerLastName, Teacher)
- PurebredAndTrained(Name, Age, AdoptionDate, OwnerFirstName, OwnerLastName, Registration, Teacher)
- Knows(TrickName, PetName, OwnerFirstName, OwnerLastName)

(b) (10 points) Into what relation(s) would you put a pet that is purebred and trained?

Solution: Only into the relation named PurebredAndTrained.
Question 14: Dependencies ........................................... 50 points

Below is a relation about pets and owners.

<table>
<thead>
<tr>
<th>Pet</th>
<th>Species</th>
<th>Age</th>
<th>Owner</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zoe</td>
<td>Dog</td>
<td>8</td>
<td>Josh</td>
<td>123-4567</td>
</tr>
<tr>
<td>River</td>
<td>Cat</td>
<td>11</td>
<td>Josh</td>
<td>123-4567</td>
</tr>
<tr>
<td>Harry</td>
<td>Rat</td>
<td>4</td>
<td>Emily</td>
<td>246-1357</td>
</tr>
<tr>
<td>Ron</td>
<td>Rat</td>
<td>4</td>
<td>Emily</td>
<td>246-1357</td>
</tr>
<tr>
<td>Snape</td>
<td>Rat</td>
<td>3</td>
<td>Emily</td>
<td>246-1357</td>
</tr>
</tbody>
</table>

(a) (30 points) Which functional dependencies are obeyed?

√ Pet → Species
√ Species → Owner Phone
√ Phone → Owner Phone
○ Species → Pet
√ Pet → Phone
√ Age → Owner

(b) (20 points) The above relation obeys this multivalued dependency (Owner → Phone).

What other rows must also be added if the row below is added? Fill in the additional dependent rows as needed (you may not need them all).

<table>
<thead>
<tr>
<th>Pet</th>
<th>Species</th>
<th>Age</th>
<th>Owner</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mal</td>
<td>Bird</td>
<td>12</td>
<td>Josh</td>
<td>555-5555</td>
</tr>
<tr>
<td>Mal</td>
<td>Bird</td>
<td>12</td>
<td>Josh</td>
<td>123-4567</td>
</tr>
<tr>
<td>Zoe</td>
<td>Dog</td>
<td>8</td>
<td>Josh</td>
<td>555-5555</td>
</tr>
<tr>
<td>River</td>
<td>Cat</td>
<td>11</td>
<td>Josh</td>
<td>555-5555</td>
</tr>
</tbody>
</table>

Page 14
Question 15: Closures ............................................................... 50 points

Here’s a relation (R), its attributes and its functional dependencies (F):

R(A, B, C, D, E)
C D → B
A → D
E → C

(a) (10 points) Which of the following are in the attribute set closure \{AB\}+?

√ \{A\}  √ \{AB\}  √ \{D\}
○ \{BC\}  ○ \{CD\}  ○ \{ABD\}

(b) (10 points) Which of the following are in the functional dependency closure of F (F+)?

√ C D → B  √ A → A  ○ D → B
○ E → C D  √ A C → B  √ A E → B

(c) (20 points) Which of the following are superkeys?

√ \{ABCDE\}  √ \{AE\}
○ \{A\}  ○ \{AB\}
○ \{BCE\}  ○ \{ABE\}

(d) (10 points) Which of the following are keys?

○ \{ABCDE\}  ○ \{AB\}
○ \{A\}  ○ \{ABE\}
○ \{BCE\}
Question 16: Lossless Joins ................................................................. 50 points
Here’s a relation (R), its attributes and its functional dependencies (F):
\[ R(A, B, C, D, E) \]
\[ C \rightarrow D \]
\[ A \rightarrow D \]
\[ E \rightarrow C \]
(a) (10 points) Which of the following sets of relations maintain the lossless join property?
\[ \checkmark R_1(ABCDE), R_2(ABCD) \]
\[ \checkmark R_1(AD), R_2(ABCE) \]
\[ \bigcirc R_1(AB), R_2(BCDE) \]
\[ \bigcirc R_1(CD), R_2(ABCE) \]
(b) (20 points) Which of the following sets of relations are entirely in Boyce-Codd Normal Form?
\[ \bigcirc R_1(ABCDE) \]
\[ \bigcirc R_1(AD), R_2(ABCE) \]
\[ \bigcirc R_1(CDB), R_2(ACDE) \]
\[ \checkmark R_1(CDB), R_2(AD), R_3(CE), R_4(AE) \]
(c) (20 points) Which of the following sets of relations are dependency preserving?
\[ \checkmark R_1(ABCDE) \]
\[ \bigcirc R_1(AD), R_2(ABCE) \]
\[ \checkmark R_1(CDB), R_2(ACDE) \]
\[ \checkmark R_1(CDB), R_2(AD), R_3(CE), R_4(AE) \]
Question 17: Decomposition ............................................................ 50 points

Here’s a relation (R), its attributes and its functional dependencies (F):

R(A, B, C, D, E)

C D → B
A → D
E → C

(a) (30 points) Decompose the above relation using the Boyce-Codd Normal Form decomposition. Use the order of F when checking for violations.

Solution:

1. $R_0$(ABCDE) not in BCNF (C D → B; violated), break into $R_1$ and $R_2$
2. $R_1$(CDB) in BCNF
3. $R_2$(ACDE) not in BCNF (A → D; violated), break into $R_3$ and $R_4$
4. $R_3$(AD) in BCNF
5. $R_4$(ACE) not in BCNF (E → C; violated), break into $R_5$ and $R_6$
6. $R_5$(EC) in BCNF
7. $R_6$(AE) in BCNF

Answer = \{CDB\}, \{AD\}, \{EC\}, \{AE\}

(b) (20 points) What is the key for the relation $R$?

Solution:

1. \{ABCDE\} is a superkey
2. \{ACDE\} is a superkey
3. \{ACE\} is a superkey
4. \{AE\} is a superkey
5. \{A\} is not a superkey
6. \{E\} is not a superkey

Answer = \{AE\} is a key