Jackie Diep
CSC 411 Databases
Assignment #5
4/14/2021

-Chapter 9 [20 points for each]

Exercises (P456): 9.13

html

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta charset="utf-8" />
   <title>Ask for N</title>
</head>
<body>
   Enter a value "n" for how many stars will be printed
   <br />
   <form action="nServlet" method="get">
      <input type="text" name="n" placeholder="Enter a value of N" />
      <input type=submit value="submit" />
   </form>
</body>
</html>
```

Servlet

```
import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;
@WebServlet("nServlet")
public class nServlet extends HTTPServlet {
        public void doGet(HttpServletRequest request,
                                                HttpServletResponse response)
                throws ServletException, IOException
        {
                response.setContentType("text/html");
                int n = Integer.parseInt(request.getParameter("n"));
                PrintWriter out = response.getWriter();
                for(int i = 0; i < n; i++)
                {
                out.print("<html><body>");
                out.print("*");
                out.print("</body></html>");
```

```
            }
            out.close();
        }
}
```

-Chapter 17 [10 points for each]

Exercises (P832-833):

17.12

1. Atomicity
    - Ensures that all effects of a transaction are reflected in the database or none are. Makes it so that failures cannot partially execute in the database.
2. Consistency
    - Makes it such that initially consistent databases are still consistent after transaction execution
3. Isolation
    - Ensures that concurrently executing transactions are isolated. Leaves the impression that no other transactions are executed concurrently
4. Durability
    - Ensures that transactional updates do not get lost after a transaction is committed

17.14

Serial schedule = all of a transactions instructions appear together

Serializable schedule = instructions of various transactions may be mixed together as long as they are equal to some serial schedule

17.18

Concurrent execution of transactions improves throughput of transactions and system utilization and also reduces the waiting time of transactions.

-Chapter 17[30 points for each]

Exercises (P832): 17.15

a. Two possible serial executions: T_{13} then T_{14} or T_{14} then T_{13}.
   In both cases, the second transaction to occur has no effect on the first transaction to occur.
   Because the requirement for consistency is A = 0 or B = 0, then consistency must be true because one attribute will always be equal to 0.

	A	B
Before transactions	0	0
After T_{13}	0	1
After T_{14}	0	1

	A	B
Before transactions	0	0
After T_{14}	1	0
After T_{13}	1	0

b.

T13	T14
Read A	
	Read B
Read B	
	Read A
If A = 0...	
Write B	
	If B = 0...
	Write A

c. There is no concurrent execution of the two that produces a serializable schedule. Consistency requirements are such that at least one attribute is = 0 at the end. Once either of the first reads of the transactions occur, then the value of the write will be changed such that the write of one transaction cannot be swapped with the first read of the other transaction. This means that no concurrent execution of the two transactions can produce a serializable schedule.

-Chapter 4 [10 points for each]

Exercises (P179-180):

4.16

```
SELECT student.ID
FROM student
LEFT OUTER JOIN takes
WHERE student.tot_cred = 0 AND course_id = NULL
```

4.20

```
CREATE VIEW tot_credits
AS (SELECT year, sum(credits)
FROM takes NATURAL JOIN course
GROUP BY year
ORDER BY year DESC)
```