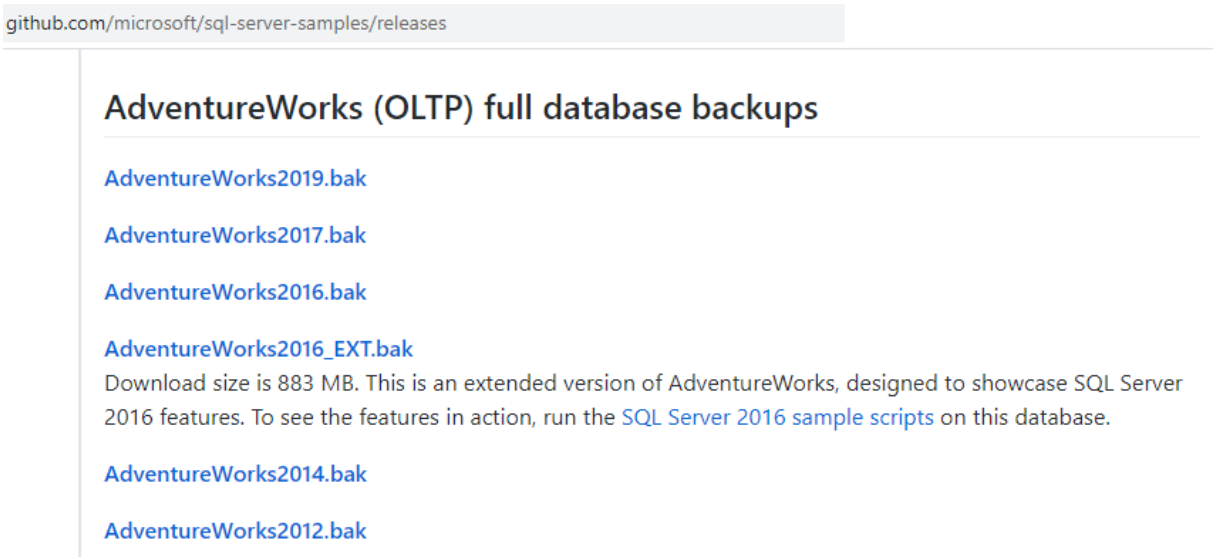
SE 308 Advanced Topics in Database Systems

TERM PROJECT 1

This is the first term project. You will have a second one until the end of the semester. In this project, you are required to make some simulations.

PLEASE READ THE REST OF THE DOCUMENT CAREFULLY!

1. First of all, you will need to download **2012 version (or later) of AdventureWorks full database** from GitHub page of Microsoft (https://github.com/microsoft/sql-server-samples/releases). Please be careful not to download the LT version! If you cannot use 2012 version with the SQL Server version installed on your computer, you can try using a newer and compatible version. The files on this page are BACKUP files, therefore, you will need to RESTORE the database from the backup file.



2. Once you restore the database from the backup, you will make some simulations on the database and will measure the time it takes to complete the transactions of users against the database.
3. In the simulation, there will be 2 types of users with 2 different transactions, Type A and Type B.
4. Type A users update the database tables with exactly the same update queries inside exactly the same transaction structure. You can find the thread structure of a Type A user at the end of the document.
5. Type B users read the database records with exactly the same select queries inside exactly the same transaction structure. You can find the thread structure of a Type B user at the end of the document.
6. You will develop a simulation application (a Windows forms application will be easier than a web application but the choice is yours). In this simulation you will be able to determine the number of users of Type A and Type B independent of each other. For example, you can determine that there will be 5 Type A users and 8 Type B users connected to the database concurrently. When you start the simulation, the program will create and start individual and independent threads

that correspond to individual users. For the previous example, there will be 5+8 = 13 concurrent threads running their respective transactions 100 times in their own loops.

7. While executing the transactions, your program will measure the time it takes to complete 100 runs of each transaction by each thread. You will present these measurements in your report in a format specified in a later item in this document.
8. One other parameter of the simulation is the Transaction Isolation Level. You will perform exactly the same simulations for each different isolation level from READ UNCOMMITTED to SERIALIZABLE, and then make a comparison in your report.
9. You will report your measurements in tables like below **for each isolation level**:

Isolation Level	READ UNCOMMITTED				
Number of Type A Users	**Number of Type B Users**	**Average Duration of Type A Threads**	**Number of Deadlocks Encountered by Type A Users**	**Average Duration of Type B Threads**	**Number of Deadlocks Encountered by Type B Users**
1	1	Xxxxx	Yyyyy	Yyyyy	Yyyyy
2	2	Xxxxx	Yyyyy	Yyyyy	Yyyyy
….	….	….	….	….	….
20	20	Xxxxx	Yyyyy	Yyyyy	Yyyyy
….	….	….	….	….	….
100	100	Xxxxx	Yyyyy	Yyyyy	Yyyyy
….	….	….	…	…	…
1000	1000	Xxxxx	Yyyyy	Yyyyy	Yyyyy

Note that you can run the simulations with as many users as you wish. I prefer higher number of users like 100s or 1000s if possible.

10. You will do all the experiments described above in two parts as follows:
    a. **PART 1:** Drop all indexes on the tables `Sales.SalesOrderDetail` and `Sales.SalesOrderHeader` before running the experiments.
    b. **PART 2:** Create necessary indexes on the tables `Sales.SalesOrderDetail` and `Sales.SalesOrderHeader` before running the experiments to make the queries faster.
11. It is very important that you do your experiments carefully and that you write a detailed report in a clear and organized way. Please pay attention to your report very much. You need to use standard A4 paper size and 11 pt Calibri font for your report.
12. Please provide 10 screenshots from your simulation program while running with different options.
13. Please put the full source code of your simulation program at the end of your report.
14. You can do this project individually or as a group of maximum four (4) students.
15. You will submit your project report file to the submission point on Blackboard with the following naming format:
    - "SE 308 Term Project 1 - Your First Name Your Last Name - Your Student ID.**pdf**"
    - If you are doing a group work, then put only one name into the file name.
16. **Deadline**: **Sunday 13 April 2025 by 23:59.**

Thread structure of Type A users:

```
beginTime = tic()
for i = 1 to 100 do
       Connect to Database
       Set Transaction Isolation Level to the parameter value
       Begin Transaction
          if (rand() < 0.5)
              Run Update Query for @BeginDate = '20110101', @EndDate = '20111231'
          if (rand() < 0.5)
              Run Update Query for @BeginDate = '20120101', @EndDate = '20121231'
          if (rand() < 0.5)
              Run Update Query for @BeginDate = '20130101', @EndDate = '20131231'
          if (rand() < 0.5)
              Run Update Query for @BeginDate = '20140101', @EndDate = '20141231'
          if (rand() < 0.5)
              Run Update Query for @BeginDate = '20150101', @EndDate = '20151231'
       Commit
       Disconnect from Database
end for
endTime = toc()
elapsed = endtime - beginTime // Record this value for reporting.
```

- As you see, you have to connect to and disconnect from the database every time you begin a transaction. You have to follow this thread structure in your program. This is a strict requirement.
- As you see again, each update query is executed with a 50% chance to make the simulations more realistic and less boring.
- If you encounter a Deadlock, your program must continue gracefully as if the deadlock did not happen.

Update Query to use:

```
UPDATE Sales.SalesOrderDetail
   SET UnitPrice = UnitPrice * 10.0 / 10.0
 WHERE UnitPrice > 100
   AND EXISTS (SELECT * FROM Sales.SalesOrderHeader
               WHERE Sales.SalesOrderHeader.SalesOrderID =
                     Sales.SalesOrderDetail.SalesOrderID
                 AND Sales.SalesOrderHeader.OrderDate
                     BETWEEN @BeginDate AND @EndDate
                 AND Sales.SalesOrderHeader.OnlineOrderFlag = 1)
```

Thread structure of Type B users:

```
beginTime = tic()
for i = 1 to 100 do
       Connect to Database
       Set Transaction Isolation Level to the parameter value
       Begin Transaction
          if (rand() < 0.5)
              Run Select Query for @BeginDate = '20110101', @EndDate = '20111231'
          if (rand() < 0.5)
              Run Select Query for @BeginDate = '20120101', @EndDate = '20121231'
          if (rand() < 0.5)
              Run Select Query for @BeginDate = '20130101', @EndDate = '20131231'
          if (rand() < 0.5)
              Run Select Query for @BeginDate = '20140101', @EndDate = '20141231'
          if (rand() < 0.5)
              Run Select Query for @BeginDate = '20150101', @EndDate = '20151231'
       Commit
       Disconnect from Database
end for
endTime = toc()
elapsed = endtime - beginTime // Record this value for reporting.
```

- As you see, you have to connect to and disconnect from the database every time you begin a transaction. You have to follow this thread structure in your program. This is a strict requirement.
- As you see again, each select query is executed with a 50% chance to make the simulations more realistic and less boring.
- If you encounter a Deadlock, your program must continue gracefully as if the deadlock did not happen.

Select Query to use:

```
SELECT SUM(Sales.SalesOrderDetail.OrderQty)
  FROM Sales.SalesOrderDetail
 WHERE UnitPrice > 100
   AND EXISTS (SELECT * FROM Sales.SalesOrderHeader
               WHERE Sales.SalesOrderHeader.SalesOrderID =
                     Sales.SalesOrderDetail.SalesOrderID
                 AND Sales.SalesOrderHeader.OrderDate
                     BETWEEN @BeginDate AND @EndDate
                 AND Sales.SalesOrderHeader.OnlineOrderFlag = 1)
```