**Take Assessment: Exercise 5**

窗体顶端

|  |
| --- |
|  |

|  |  |  |
| --- | --- | --- |
|  | Please answer the following question(s). If the assessment includes multiple-choice questions, click the "Submit Answers" button when you have completed those questions. |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1. |  | [Go to bottom of question.](https://www.icarnegie.com/takeassmcmd.php?course_section_id=9788469&assm_id=6860314#6860317#6860317) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Programming with Transactions**  **Testing Commits and Rollbacks**  Almost all commercial-strength DBMS software supports the ability to undo the effects of a transaction before those effects are committed to the database. This is particularly useful when an operation must be performed to completion or not performed at all. You may undo these effects on the data using rollbacks. This portion of the exercise will familiarize you with rollbacks and commits.   1. Run the [SQL script](https://www.icarnegie.com/content/SSD/SSD7/1.5.1/normal/pg-performance/pg-transactionmanagement/qn-pr-transaction/qn-pr-transaction/handout/build_trans_db.sql) provided to build a small bank database consisting of two columns: an account id number and a balance. There are two accounts, a savings account and a checking account. The savings account has *id = 1* and the checking account has *id = 2*. 2. Begin a new transaction. 3. Select balance of the checking account and paste the output in a file named *rollback.txt*. 4. Delete both the savings and the checking accounts. Place the SQL query to perform this operation in *rollback.txt*. 5. Select all of the data in the account table and paste the output in *rollback.txt*. 6. Roll back the transaction. 7. Select the balance of the savings and checking accounts and paste the output in *rollback.txt*. 8. Begin a new transaction. 9. Delete the savings account. Place the SQL code to do this in *rollback.txt*. 10. Select all rows from the account table and paste the output in *rollback.txt*. 11. Commit the transaction. 12. Attempt to rollback the transaction that you have just committed. What values are stored in the tables? Please explain the effects of transaction commits and rollbacks you have observed above. Place your explanation in *rollback.txt*.   **Testing Isolation Levels**  PostgreSQL supports two transaction isolation levels. To become familiar with PostgreSQL isolation levels, perform the following tasks:   1. Run the [SQL script](https://www.icarnegie.com/content/SSD/SSD7/1.5.1/normal/pg-performance/pg-transactionmanagement/qn-pr-transaction/qn-pr-transaction/handout/build_trans_db.sql) provided to build a small bank database consisting of two columns: an account id number and a balance. There are two accounts, a savings account and a checking account. The savings account has *id = 1* and the checking account has *id = 2*. 2. Begin two sessions of the PostgreSQL client in two separate console windows. 3. Begin new transactions in both windows. 4. In the first window, update the checking account to have a balance of 455.66. 5. In the first window, select all of the data from the account table and place the output into the file *isolation.txt*. Be sure to label the data so that it is clear this data is from the first table. 6. In the second window, select all of the data from the account table and place the output into the file *isolation.txt*. Be sure to label the data so that it is clear this is data from the second window. 7. Commit the transaction in the first window to update the account table. 8. In the second window, select all of the data from the account table and place the output into the file *isolation.txt.* Be sure to label the data so that it is clear this is data from the second window. What has changed about the data and why? Place the answer to this question in *isolation.txt*. 9. Commit the transaction in second window. 10. In both windows, begin new transactions. 11. Set the transaction isolation level of the transaction in the second window to serializable. Place the code to do this in *isolation.txt*. 12. In the first window, set the balance of checking account balance to 1400.00. 13. In the second window, set the balance of savings account balance to 1. 14. Select all data from the account table in the first and second window and place it into *isolation.txt*. Be sure to label the data clearly to denote what data came from the first and second windows. 15. Commit the data in the first window. 16. Select all of the data from the account table in the second window. Has the data for the checking account changed? Why or why not? Has the data for the savings account changed in the first window? Why or why not? Places the answers to these questions in *isolation.txt* 17. Commit the data in the second window. Display all of the data from the account table in both windows. Place the output of both windows into *isolation.txt*. Clearly label which data is from the first window and which data is from the second window. What do you notice now about the balances of the checking and savings accounts? Please explain. Place your explanation in *isolation.txt*.   **Blocking and Deadlocks**  In this portion of the exercise, you will cause two transactions to block and deadlock two transactions in PostgreSQL.   1. Run the [SQL script](https://www.icarnegie.com/content/SSD/SSD7/1.5.1/normal/pg-performance/pg-transactionmanagement/qn-pr-transaction/qn-pr-transaction/handout/build_trans_db.sql) provided to build a small bank database consisting of two columns: an account id number and a balance. There are two accounts, a savings account and a checking account. The savings account has *id = 1* and the checking account has *id = 2*. 2. Begin two sessions of the PostgreSQL client in two separate console windows. 3. Begin new transactions in both windows. 4. Update the checking account's balance to 455.75 in the second window. 5. Update the checking account's balance to 1400.00 in the first window. Does the update occur? Why or why not? Place your answer in *deadlock.txt*. 6. Commit the transaction in the second window. What do you notice happening in the first window? Please explain. Place your answer in *deadlock.txt*. 7. Commit the transaction in the first window. 8. Begin a new transaction in each window. 9. Update the savings account balance to 2400.00 in the second window. 10. Update the checking account balance to 2000.00 in the first window. 11. Update the savings account balance to 1400.00 in the first window. 12. Update the checking account balance to 1000.50 in the second window. 13. What happens to the transactions? Why? Place your answer in *deadlock.txt*. 14. Try selecting the all of the data from the account table in each window. What do you notice? Place your answer in *deadlock.txt*.   **Application**  The final exercises will test your knowledge of transactions and their behavior in real world situations.   1. Suppose William and Julie share a savings and a checking account at a bank. The bank has many automatic teller machines. William banks from one ATM while Julie banks at another ATM. William wishes first to deposit four hundred dollars into the savings account and then to transfer three hundred dollars from the checking account to the savings account. Julie wishes first to deposit a check for five hundred dollars into the checking account and then withdraw one hundred dollars from the savings account.   The sequence of operations for a deposit is:   * + Select the current account balance for the customer from the database.   + Increase this account balance by the deposit amount.   + Update the customer's account balance in the database to the new amount.   The sequence of operations for a withdrawal is:   * + Select the current account balance for the customer from the database.   + Decrease this account balance by the withdrawal amount.   + Update the customer's account balance in the database to the new amount.   The sequence of operations for a transfer is:   * + Select the current account balance for the account from where the funds are transferred.   + Decrease the current balance by the amount transferred.   + Select the balance of the account to where the funds are being transferred.   + Increase the balance of the account where money is transferred.   + Update both account balances in the database.   Suppose the account activities occur as an interleaved execution of transactions updating the database. You will now analyze a number of sample scenarios involving these account activities. Each scenario fixes one or more problems with the previous scenario, but there still may be problems in the scenario. You must identify what problem was fixed as well as why the fix leads to a more correct execution. Note that in the scenario below, bulleted operations are those that occur in the DBMS, non-bulleted items are those that occur at the ATM machine. Also, you should assume that these scenarios are run on PostgreSQL meaning that you may use only those isolation levels supported by PostgreSQL. You may assume that no other transactions are occurring in the database except these transactions. This means that you do not need to consider the potential effects of other types of transactions on the data. Also note that the ordering of these operations must be consistent for each scenario (all of William's operations occur in the proper order for his operations and all of Julie's operations occur in the proper order for her set of operations, that is the time flows from top to bottom for each user's transactions). You cannot assume that all of William's transaction will occur before all of Julie's or vice-versa. You also cannot assume that the operations will always be interleaved. It is possible that all of Julie's operations will occur before all of Williams, all of William's will occur before all of Julie's, and any interleaving of the operations is also possible.   * + Scenario A  |  |  | | --- | --- | | William | Julie | | * + - Begin transaction (read committed isolation level) | * + - Begin transaction (read committed isolation level) | | ATM prompts user for operation. | ATM prompts user for operation. | | William Chooses Deposit into Savings. | Julie Chooses Deposit into Checking. | | ATM prompts user for amount. | ATM prompts user for amount. | | William enters 400. | Julie enters 500. | | * + - Select balance of savings | * + - Select balance of checking | | savings\_balance = savings\_balance + $400 | checking\_balance = checking\_balance + $500 | | * + - Update savings balance in database. | * + - Update checking balance in database. | | ATM displays confirmation of deposit. | ATM displays confirmation of deposit. | | ATM prompts user for operation. | ATM prompts user for operation. | | William chooses transfer from checking to savings. | Julie chooses withdrawal from savings. | | ATM prompts user for amount. | ATM prompts user for amount. | | William enters 300. | Julie enters 100. | | * + - Select the checking balance. | * + - Select the savings checking balance in database. | | checking\_balance = checking\_balance - $300 | savings\_balance = savings\_balance - $100 | | * + - Select the savings balance. | * + - Update savings balance in database. | | savings\_balance = savings\_balance + $300 | ATM displays confirmation of withdrawal. | | * + - Update savings balance in database. | ATM prompts user for operation. | | * + - Update checking balance in database. | Julie chooses no more operations. | | ATM displays confirmation of transfer. | * + - End Transaction | | ATM prompts user for operation. |  | | William chooses no more operations. |  | | * + - End Transaction |  |  * + We will help you with this first scenario. The following problems exist:     - Incorrect Isolation Level     - Confirmation Messages are not in the correct positions     - Incorrect Transaction Boundaries     - Pausing for user input within transaction boundaries   For each of these problems, you must state why they are a problem. We will provide with help on one of them and you must explain the rest.   The confirmation messages are placed such that a user can receive a confirmation message before the transaction ends. This is a problem because a transaction may roll back but the user believes his transaction has ended successfully. All confirmation messages must be displayed after a transaction has been committed.   * + Scenario B  |  |  | | --- | --- | | William | Julie | | ATM prompts user for operation. | ATM prompts user for operation. | | William Chooses Deposit into Savings. | Julie Chooses Deposit into Checking. | | * + - Begin transaction (read committed isolation level) | * + - Begin transaction (read committed isolation level) | | ATM prompts user for amount. | ATM prompts user for amount. | | William enters 400. | Julie enters 500. | | * + - Select balance of savings | * + - Select balance of checking | | savings\_balance = savings\_balance + $400 | checking\_balance = checking\_balance + $500 | | * + - Update savings balance in database. | * + - Update checking balance in database. | |  |  | | * + - End Transaction | * + - End Transaction | | ATM displays confirmation of deposit. | ATM displays confirmation of deposit. | | ATM prompts user for operation. | ATM prompts user for operation. | | William chooses transfer from checking to savings. | Julie chooses withdrawal from savings. | | * + - Begin transaction (read committed isolation level) | * + - Begin transaction (read committed isolation level) | | ATM prompts user for amount. | ATM prompts user for amount. | | William enters 300. | Julie enters 100. | | * + - Select the checking balance. | * + - Select the savings checking balance in database. | | checking\_balance = checking\_balance - $300 | savings\_balance = savings\_balance - $100 | | * + - Select the savings balance. | * + - Update savings balance in database. | | savings\_balance = savings\_balance + $300 | * + - End Transaction | | * + - Update savings balance in database. | ATM displays confirmation of withdrawal. | | * + - Update checking balance in database. | ATM prompts user for operation. | | * + - End Transaction | Julie chooses no more operations. | | ATM displays confirmation of transfer. |  | | ATM prompts user for operation. |  | | William chooses no more operations. |  |  * + Scenario C  |  |  | | --- | --- | | William | Julie | | ATM prompts user for operation. | ATM prompts user for operation. | | William Chooses Deposit into Savings. | Julie Chooses Deposit into Checking. | | * + - Begin transaction (serializable isolation level) | * + - Begin transaction (serializable isolation level) | | ATM prompts user for amount. | ATM prompts user for amount. | | William enters 400. | Julie enters 500. | | * + - Select balance of savings | * + - Select balance of checking | | savings\_balance = savings\_balance + $400 | checking\_balance = checking\_balance + $500 | | * + - Update savings balance in database. | * + - Update checking balance in database. | |  |  | | * + - End Transaction | * + - End Transaction | | ATM displays confirmation of deposit. | ATM displays confirmation of deposit. | | ATM prompts user for operation. | ATM prompts user for operation. | | William chooses transfer from checking to savings. | Julie chooses withdrawal from savings. | | * + - Begin transaction (serializable isolation level) | * + - Begin transaction (serializable isolation level) | | ATM prompts user for amount. | ATM prompts user for amount. | | William enters 300. | Julie enters 100. | | * + - Select the checking balance. | * + - Select the savings checking balance in database. | | checking\_balance = checking\_balance - $300 | savings\_balance = savings\_balance - $100 | | * + - Select the savings balance. | * + - Update savings balance in database. | | savings\_balance = savings\_balance + $300 | * + - End Transaction | | * + - Update savings balance in database. | ATM displays confirmation of withdrawal. | | * + - Update checking balance in database. | ATM prompts user for operation. | | * + - End Transaction | Julie chooses no more operations. | | ATM displays confirmation of transfer. |  | | ATM prompts user for operation. |  | | William chooses no more operations. |  |  * + You must write the final sequence of events that lead to a correct execution while allowing the maximum amount of concurrency. Be sure to state each step for both William and Julie. Also, state what actions are handled by the DBMS and what actions are handled by the ATM machine.  1. Suppose you are asked to write a Web-based database application where tickets may be purchased for a local movie theater by a large number of Web clients. You may assume that the movie theater customers usually wait until thirty minutes or less before a show begins, thus the system has many users accessing the site concurrently. The sequence of events for a typical customer:    * The system queries the database for a list of current movies and displays them.    * The user selects the movie he wants to see from a list.    * There are typically many showings of the same movie throughout the day and the user must select which showing he will attend. To this end, the system queries the database for all of the daily showings of the chose movie and displays them. The user must pick one of these times.    * The system queries the DBMS and returns how many seats are available for the selected show.    * If there are seats available, the system displays the count to the user.    * The user must enter how many tickets he wants to buy.    * Finally, the user confirms his order and completes payment.   The table structure for this scenario along with some sample data is given below. Create a new database in PostgreSQL named movie and run the [SQL script](https://www.icarnegie.com/content/SSD/SSD7/1.5.1/normal/pg-performance/pg-transactionmanagement/qn-pr-transaction/qn-pr-transaction/handout/create_movie.sql) provided to build these tables.   |  |  | | --- | --- | | Movies | | | Movie Id (PK) | Movie Title | | 001 | *The Terminator* | | 002 | *James Bond* | | 003 | *The Matrix* |  |  |  |  |  | | --- | --- | --- | --- | | Movie Showings | | | | | Movie Id (PK,FK) | Movie Time(PK) | Total Seats | Available Standard Seats | | 001 | 1:00 PM | 40 | 25 | | 001 | 4:00 PM | 30 | 27 | | 002 | 12:30 PM | 50 | 32 | | 002 | 7:30 PM | 75 | 17 | | 003 | 9:30 PM | 100 | 10 |   You must modify the [Java console application provided](https://www.icarnegie.com/content/SSD/SSD7/1.5.1/normal/pg-performance/pg-transactionmanagement/qn-pr-transaction/qn-pr-transaction/handout/MovieTransaction.java) that simulates this scenario using PostgreSQL meaning that you can only use those isolation levels supported by PostgreSQL. This application sets incorrect transaction boundaries and uses an incorrect isolation level. Your task is to correct these mistakes by moving the transaction boundaries to where they allow the best performance for all system users. Note that you may use more than one transaction in your solution. For each transaction that is used in your solution, you must include the following:   * + An explanation as to where the transaction begins and ends (make specific references to the code immediately before and after it).   + State why these boundaries allow for the best performance.   + State why the isolation level you have selected will lead to a consistent view of the data to all users.   To help yourself do your best on this assessment, consult this general list of [grading guidelines](https://www.icarnegie.com/content/SSD/SSD7/1.5.1/normal/pg-performance/pg-transactionmanagement/qn-pr-transaction/qn-pr-transaction/handout/rubrics.html). |  |

|  |  |  |
| --- | --- | --- |
|  | [Go to top of question.](https://www.icarnegie.com/takeassmcmd.php?course_section_id=9788469&assm_id=6860314#top_6860317#top_6860317) |  |

窗体底端