**ISS4014 – Database Systems and Web Integration**

**Chapter 10 – Activities and Homework**

Name: Logan Strong

Date: March 13th, 2024

**Chapter 10 REVIEW (5 points)**

Respond to the following Chapter 10 review questions (you may have discussed some of these questions in class, so be sure to include ideas and information you gained from the class discussion).

1. (2 pts) List and discuss the five transaction properties.

| **#** | **Transaction Property** | **Describe in your own words.** |
| --- | --- | --- |
| 1 | Atomicity | This property means that everything in a transaction must be treated totally individually, and everything in the transaction must be completed or else the transaction is aborted. |
| 2 | Consistency | This is a condition where all data integrity constraints have been satisfied. Databases that want to ensure consistency will set these constraints to do so, or else there could be transactions that yield conflicting results. |
| 3 | Isolation | Is when an item being used by one transaction is not available to any other transaction until the first transaction has been completed. |
| 4 | Durability | Is the property that makes sure when a transaction is committed that it can’t be undone no matter what. |
| 5 | Serializability | Property where the specified order of each transaction can create the same database as if it were produced in a serial fashion |

1. What is a lock, and how, in general, does it work?

|  |
| --- |
| A tool that ensures the unique use of data in a certain transaction. Transactions require them prior to data access, and it is released after the operation. To say succinctly, they just work to lock transactions like you’d lock anything, to protect it (in this case protect the data). |

1. What are the different levels of lock granularity?

|  |
| --- |
| Database Level, Table Level, Page Level, Row Level, and Field Level |

1. What is a deadlock? Discuss **three specific** strategies for dealing/avoiding deadlocks.

|  |  |
| --- | --- |
| What is a deadlock? | Deadlock is when two transactions wait indefinitely for one another to unlock data. |

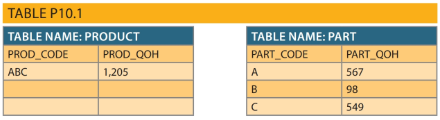
| **#** | **Strategy Name** | **Describe in your own words.** |
| --- | --- | --- |
| 1 | Deadlock Prevention: | Any transaction that requests a new lock is aborted when a deadlock can occur. |
| 2 | Deadlock Detection: | The DBMS will periodically test the database for deadlocks. |
| 3 | Deadlock Avoidance: | Transactions must obtain all of the locks needed before it is allowed to execute. |

**Chapter 10 PROBLEMS (25 points)**

See the text for details for each question.

1. (5 pts) **Suppose you are a manufacturer of product ABC, which is composed of parts A, B, and C. Each time a new product is created, it must be added to the product inventory using the PROD\_QOH in a table named PRODUCT. And each time the product ABC is created, the parts inventory, using PART\_QOH in a table named PART, must be reduced by one each of parts A, B, and C.**

**The sample database contents are shown in Table P10.1**



**You may wish to create a simple database with the two tables in this problem and actually write and test the SQL statements in parts b and c. to ensure they are correct.**

a. How many database requests can you identify for an inventory update involving PRODUCT and PART tables?

|  |
| --- |
| 4 requests |

b. Using SQL, write each database request you identified in step a.

|  |
| --- |
| Insert Into Product (prod\_code, prod\_qoh)  Values (ABC, LAST\_INSERT\_ID() + 1);  Update Part  Set part\_qoh = part\_qoh – 1  Where part\_code In (‘A’, ‘B’, ‘C’); |

c. Write the SQL statements to make them a single and complete transaction(s).

|  |
| --- |
| Begin Transaction;  Insert Into Product (prod\_code, prod\_qoh)  Values (ABC, LAST\_INSERT\_ID() + 1);  Update Part  Set part\_qoh = part\_qoh – 1  Where part\_code In (‘A’, ‘B’, ‘C’);  Commit; |

d. Write the transaction log using Table 10.1 as your template.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TRL ID** | **TRX Num** | **PREV PTR** | **NEXT PTR** | **Operation** | **Table** | **Row ID** | **Attribute** | **Before value** | **After value** |
| 1 | 1 | NULL |  | START |  |  |  |  |  |
| 2 | 1 | 1 | 2 | INSERT | PRODUCT | ABC | PROD\_CODE, PROD\_QOH | NULL | 1,205 |
| 3 | 1 | 2 | 3 | UPDATE | PART | A | PROD\_QOH | 568 | 567 |
| 4 | 1 | 3 | 4 | UPDATE | PART | B | PROD\_QOH | 99 | 98 |
| 5 | 1 | 4 | 5 | UPDATE | PART | C | PROD\_QOH | 550 | 549 |
| 6 | 1 | 5 | NULL | COMMIT |  |  |  |  |  |

e. Using the transaction log created in Step d, trace its use in database recovery. Assume the database experienced a crash just before the final commit was executed. What are the TRL IDs (from step d) that must be executed, and what specific action must be done to reverse the transaction and return the database to a stable state before the transaction starts? The truncations should be in reverse order, and the actions should be specific.

| **TRL ID** | **Action** |
| --- | --- |
| 6 | Doesn’t need to be executed |
| 5 | Add 1 back to Part C’s PROD\_QOH |
| 4 | Add 1 back to Part B’s PROD\_QOH |
| 3 | Add 1 back to Part A’s PROD\_QOH |
| 2 | Delete the last entry from the Product Table |
| 1 | Doesn’t need to be executed |

1. (2 pts) Describe the three most common problems with concurrent transaction execution. Explain how concurrency control can be used to avoid those problems.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Problem Name** | **Describe in your own words** | **How can concurrency control be used to avoid the identified 3 problems?** |
| 1 | Lost Updates | This problem happens when two transactions running at the same time are manipulating the same data and one transaction ends up overwriting the other. | Concurrency control can be used to avoid this problem by not letting another transaction (by locking) update information that needs to be read. |
| 2 | Uncommitted Data | A problem when a transaction is using data that hasn’t been committed yet meaning it is inaccurate data. | Concurrency control can be used to avoid this problem by locking the value(s) being changed until that transaction is committed. |
| 3 | Inconsistent Retrievals | When a transaction accesses data before or after other transactions are finished working with the same data. | Concurrency control can be used to avoid this problem by locking the value(s) till each one is completed in a serial manner. |

1. (2 pts) What DBMS component is responsible for concurrency control? How is this feature used to resolve conflicts?

|  |  |
| --- | --- |
| **DBMS Component** | **How is it used to resolve conflicts?** |
| The Scheduler | It establishes the order operations are executed within concurrent transactions. |

1. (3 pts) Suppose that your database system has failed. Describe the database recovery process and the use of deferred-write and write-through techniques.

|  |  |
| --- | --- |
| Describe the database recovery process using the…  **Deferred-Write Technique** | Describe the database recovery process using the…  **Write-Through Technique** |
| You identify the last checkpoint in the transaction log, but since nothing was fully updated then you don’t need to revert anything, just identify the problem. | You identify the last checkpoint in the transaction log, and since the values in the database had been modified the changes will have to be reverted and done in reverse order. |

|  |  |
| --- | --- |
| **Technique** | **Describe how the technique works in your own words.** |
| **Deferred-Write** | This technique used in the recovery process handles transaction operations uniquely. Instead of immediately updating the database, only the transaction log is updated. |
| **Write-Through** | With this technique, the database is immediately, even before the transaction commits. |

**Create a ch10\_abc\_markets database and use the Ch10\_ABC\_Markets\_MySQL file to create and load the tables for use in problems 5-10. Be sure to set ch10\_abc\_markets as the default schema.**

1. (2 pts) **ABC Markets sell products to customers. The relational diagram shown in Figure P10.6 in the textbook represents the main entities for ABC’s database. See the textbook starting on page 513, problem 6, for important characteristics.**

**Using this database, write the SQL code to represent each one of the following transactions. Make sure to use BEGIN and COMMIT to group the SQL statements in logical transactions.**

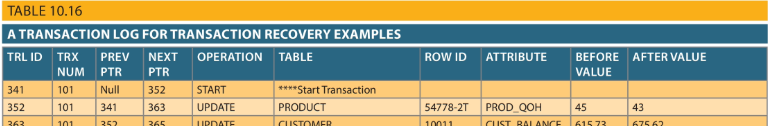
1. On April 15, 2018, customer ‘10011’ makes a credit purchase (30 days) of two units of product ’14-Q1/L3’ with a unit price of $17.49; the tax rate is 10 percent. The invoice number is 1010, and this invoice has only one product line. (NOTE: Be sure to calculate the full invoice value (rounded up) with quantity and tax, but add the extended unit price for the line item).

|  |
| --- |
| START TRANSACTION;  INSERT INTO Invoice VALUES (1010, '10011', '2018-04-15', 38.49, 'NET-30', 'OPEN');  INSERT INTO Line VALUES (1010, 1, '14-Q1/L3', 2, 17.49);  UPDATE Product SET p\_qtyoh = p\_qtyoh - 1 WHERE p\_code = '14-Q1/L3';  UPDATE Customer SET Cus\_datelstpur = '2018-04-15', cus\_balance = cus\_balance + ROUND(2 \* 17.49 \* 1.1, 2)  WHERE cus\_code = '10011';  COMMIT; |

1. On May 10, 2018, customer ‘10011’ makes a payment of $38.48 in cash. The payment ID is 1012. (Note: Be sure to update the Invoice status to PAID.)

|  |
| --- |
| 1. START TRANSACTION; 2. INSERT INTO Payments VALUES (1012, '10011', '2018-05-10', 38.48, 'Cash', 'None'); 3. UPDATE Invoice SET Inv\_status = 'PAID' WHERE Cus\_code = '10011' AND Inv\_status= 'UNPAID'; 4. COMMIT; |

1. (3 pts) Create a simple transaction log (using the format shown in **Table 10.16**) to represent the actions of the two previous transactions.



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TRL ID** | **TRX Num** | **PREV PTR** | **NEXT PTR** | **Operation** | **Table** | **Row ID** | **Attribute** | **Before value** | **After value** |
| 101 | 555 | NULL | 102 | Start |  |  |  |  |  |
| 102 | 555 | 101 | 103 | Insert | Invoice | 1010 | Cus\_code |  | 10011 |
| 103 | 555 | 102 | 104 | Insert | Invoice | 1010 | Inv\_date |  | 2018-04-15 |
| 104 | 55 | 103 | 105 | Update | Product | 10011 | P\_qtyoh |  |  |
| 105 | 555 | 104 | 106 | Update | Customer | 10011 | Cus\_datelstpur |  | 2018-05-10 |
| 106 | 555 | 105 | 107 | Update | Customer | 10011 | Cus\_balance |  | 38.48 |
| 107 | 555 | 106 | Null | Commit |  |  |  |  |  |

1. (2 pts) **Assuming that pessimistic locking is being used, but the two-phase locking protocol is not, create a chronological list of the locking, unlocking, and data manipulation activities that would occur during the complete processing of the transaction described in Problem 5a.**

|  |  |
| --- | --- |
| **Step#** | **Action** |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | START TRANSACTION;  Lock table invoice;  INSERT INTO Invoice VALUES (1010, '10011', '2018-04-15', 38.49, 'NET-30', 'OPEN');  Unlock table invoice;  Lock table line;  INSERT INTO Line VALUES (1010, 1, '14-Q1/L3', 2, 17.49);  Unlock table line  Lock table product  UPDATE Product SET p\_qtyoh = p\_qtyoh - 1 WHERE p\_code = '14-Q1/L3';  Unlock table product  Lock table customer  UPDATE Customer SET Cus\_datelstpur = '2018-04-15', cus\_balance = cus\_balance + ROUND(2 \* 17.49 \* 1.1, 2)  WHERE cus\_code = '10011';  Unlock table customer;  COMMIT; |
| … |  |

1. (2 pts) **Assuming that pessimistic locking with the two-phase locking protocol is being used, create a chronological list of the locking, unlocking, and data manipulation activities that would occur during the complete processing of the transaction described in Problem 5a.**

|  |  |
| --- | --- |
| **Step#** | **Action** |
| 1  2  3  4  5  6  7  8 | START TRANSACTION;  Lock tables line, product, invoice, customer;  INSERT INTO Invoice VALUES (1010, '10011', '2018-04-15', 38.49, 'NET-30', 'OPEN');  INSERT INTO Line VALUES (1010, 1, '14-Q1/L3', 2, 17.49);  UPDATE Product SET p\_qtyoh = p\_qtyoh - 1 WHERE p\_code = '14-Q1/L3';  UPDATE Customer SET Cus\_datelstpur = '2018-04-15', cus\_balance = cus\_balance + ROUND(2 \* 17.49 \* 1.1, 2)  WHERE cus\_code = '10011';  Unlock tables  COMMIT; |
| … |  |

1. (2 pts) Assuming that pessimistic locking is being used, but the two-phase locking protocol is not, create a chronological list of the locking, unlocking, and data manipulation activities that would occur during the complete processing of the transaction described in Problem 5b.

|  |  |
| --- | --- |
| **Step#** | **Action** |
|  |  |
| … |  |

1. (2 pts) Assuming that pessimistic locking with the two-phase locking protocol is being used, create a chronological list of the locking, unlocking, and data manipulation activities that would occur during the complete processing of the transaction described in Problem 5b.

|  |  |
| --- | --- |
| **Step#** | **Action** |
|  |  |
| … |  |