

Ch10 Transaction Management

Exercises (Please disregard question numbers – copy/paste error) 😊

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

Table P10.1 The Database for Problem 1

Table name: PRODUCT

PROD_CODE	PROD_QOH
ABC	1,205

Table name: PART

PART_CODE	PART_QOH
A	567
B	98
C	549

How many database requests can you identify for an inventory update for both PRODUCT and PART?

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

Table P10.1 The Database for Problem 1

Table name: PRODUCT

PROD_CODE	PROD_QOH
ABC	1,205

Table name: PART

PART_CODE	PART_QOH
A	567
B	98
C	549

How many database requests can you identify for an inventory update for both PRODUCT and PART?

Depending in how the SQL statements are written, there are two correct answers: 4 or 2.

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

Table name: PRODUCT

PROD_CODE	PROD_QOH
ABC	1,205

Table name: PART

PART_CODE	PART_QOH
A	567
B	98
C	549

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

The database requests are shown in the following table.

Four SQL statements	Two SQL statements
<pre>UPDATE PRODUCT SET PROD_QOH = PROD_OQH + 1 WHERE PROD_CODE = 'ABC' UPDATE PART SET PART_QOH = PART_OQH - 1 WHERE PART_CODE = 'A' UPDATE PART SET PART_QOH = PART_OQH - 1 WHERE PART_CODE = 'B' UPDATE PART SET PART_QOH = PART_OQH - 1 WHERE PART_CODE = 'C'</pre>	<pre>UPDATE PRODUCT SET PROD_QOH = PROD_OQH + 1 WHERE PROD_CODE = 'ABC' UPDATE PART SET PART_QOH = PART_OQH - 1 WHERE PART_CODE = 'A' OR PART_CODE = 'B' OR PART_CODE = 'C'</pre>

Write the complete transaction(s).

Write the complete transaction(s).

The transactions are shown in the following table.

Four SQL statements	Two SQL statements
<pre>BEGIN TRANSACTION UPDATE PRODUCT SET PROD_QOH = PROD_QOH + 1 WHERE PROD_CODE = 'ABC' UPDATE PART SET PART_QOH = PART_QOH - 1 WHERE PART_CODE = 'A' UPDATE PART SET PART_QOH = PART_QOH - 1 WHERE PART_CODE = 'B' UPDATE PART SET PART_QOH = PART_QOH - 1 WHERE PART_CODE = 'C' COMMIT;</pre>	<pre>BEGIN TRANSACTION UPDATE PRODUCT SET PROD_QOH = PROD_QOH + 1 WHERE PROD_CODE = 'ABC' UPDATE PART SET PART_QOH = PART_QOH - 1 WHERE PART_CODE = 'A' OR PART_CODE = 'B' OR PART_CODE = 'C' COMMIT;</pre>

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

TABLE 10.1

A TRANSACTION LOG

TRL_ID	TRX_NUM	PREV_PTR	NEXT_PTR	OPERATION	TABLE	ROW ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE
341	101	Null	352	START	****Start Transaction				
352	101	341	363	UPDATE	PRODUCT	1558-QW1	PROD_QOH	25	23
363	101	352	365	UPDATE	CUSTOMER	10011	CUST_BALANCE	525.75	615.73
365	101	363	Null	COMMIT	**** End of Transaction				




TRL_ID = Transaction log record ID
 TRX_NUM = Transaction number
 PTR = Pointer to a transaction log record ID
 (Note: The transaction number is automatically assigned by the DBMS.)

TABLE 10.1

A TRANSACTION LOG

TRL_ID	TRX_NUM	PREV_PTR	NEXT_PTR	OPERATION	TABLE	ROW ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE
341	101	Null	352	START	****Start Transaction				
352	101	341	363	UPDATE	PRODUCT	1558-QW1	PROD_QOH	25	23
363	101	352	365	UPDATE	CUSTOMER	10011	CUST_BALANCE	525.75	615.73
365	101	363	Null	COMMIT	**** End of Transaction				


 TRL_ID = Transaction log record ID
 TRX_NUM = Transaction number
 PTR = Pointer to a transaction log record ID
 (Note: The transaction number is automatically assigned by the DBMS.)

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

We assume that product 'ABC' has a PROD_QOH = 23 at the start of the transaction and that the transaction is representing the addition of 1 new product. We also assume that PART components "A", "B" and "C" have a PROD_QOH equal to 56, 12, and 45 respectively.

TRL ID	TRX NUM	PREV PTR	NEXT PTR	OPERATION	TABLE	ROW ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE
1	1A3	NULL	2	START	**START TRANSACTION				
2	1A3	1	3	UPDATE	PRODUCT	'ABC'	PROD_QOH	23	24
3	1A3	2	4	UPDATE	PART	'A'	PART_QOH	56	55
4	1A3	3	5	UPDATE	PART	'B'	PART_QOH	12	11
5	1A3	4	6	UPDATE	PART	'C'	PART_QOH	45	44
6	1A3	5	NULL	COMMIT	** END TRANSACTION				

Using the transaction log you created in Step d, trace its use in database recovery.

TRL ID	TRX NUM	PREV PTR	NEXT PTR	OPERATION	TABLE	ROW ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE
1	1A3	NULL	2	START	**START TRANSACTION				
2	1A3	1	3	UPDATE	PRODUCT	'ABC'	PROD_QOH	23	24
3	1A3	2	4	UPDATE	PART	'A'	PART_QOH	56	55
4	1A3	3	5	UPDATE	PART	'B'	PART_QOH	12	11
5	1A3	4	6	UPDATE	PART	'C'	PART_QOH	45	44
6	1A3	5	NULL	COMMIT	** END TRANSACTION				

Using the transaction log you created in Step d, trace its use in database recovery.

TRL ID	TRX NUM	PREV PTR	NEXT PTR	OPERATION	TABLE	ROW ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE
1	1A3	NULL	2	START	**START TRANSACTION				
2	1A3	1	3	UPDATE	PRODUCT	'ABC'	PROD_QOH	23	24
3	1A3	2	4	UPDATE	PART	'A'	PART_QOH	56	55
4	1A3	3	5	UPDATE	PART	'B'	PART_QOH	12	11
5	1A3	4	6	UPDATE	PART	'C'	PART_QOH	45	44
6	1A3	5	NULL	COMMIT	** END TRANSACTION				

Begin with the last trl_id (trl_id 6) for the transaction (trx_num 1A3) and work backward using the prev_ptr to identify the next step to undo moving from the end of the transaction back to the beginning.

Trl_ID 6: Nothing to change because it is a end of transaction marker.

Trl_ID 5: Change PART_QOH from 44 to 45 for ROW_ID 'C' in PART table.

Trl_ID 4: Change PART_QOH from 11 to 12 for ROW_ID 'B' in PART table.

Trl_ID 3: Change PART_QOH from 55 to 56 for ROW_ID 'A' in PART table.

Trl_ID 2: Change PROD_QOH from 24 to 23 for ROW_ID 'ABC' in PRODUCT table.

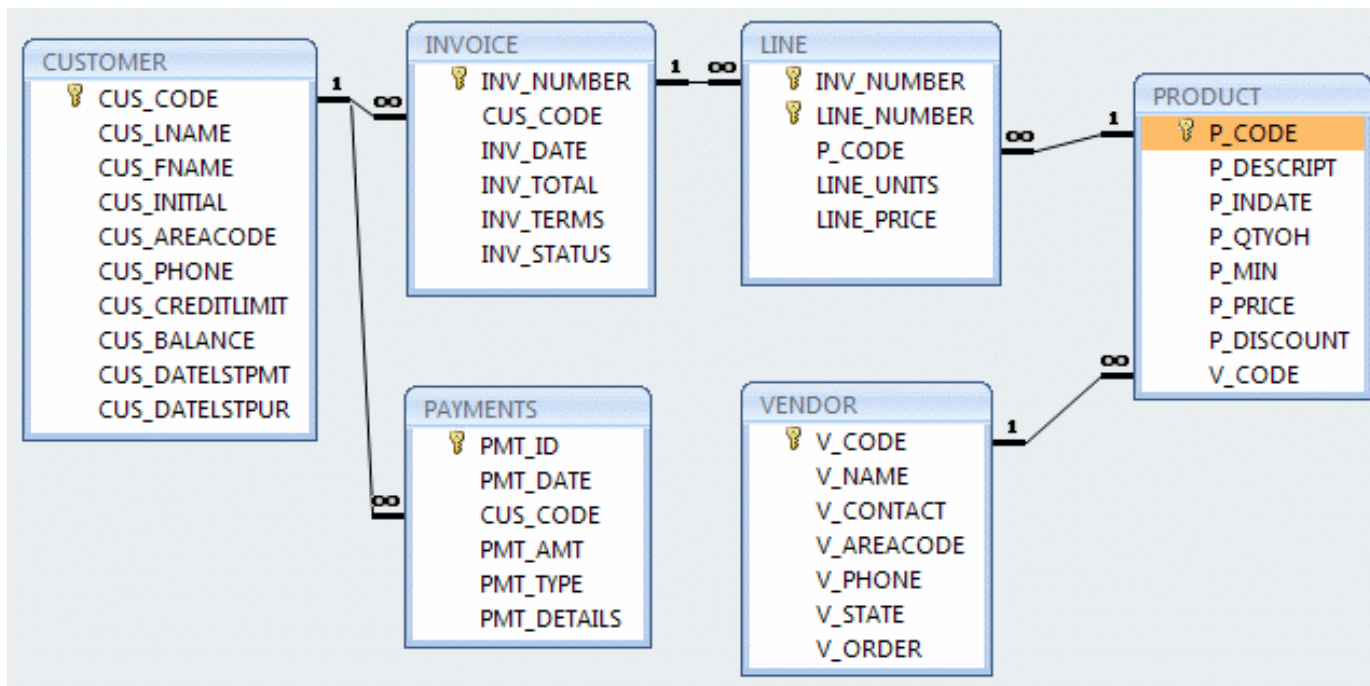
Trl_ID 1: Nothing to change because it is a beginning of transaction marker.

ABC Markets sell products to customers. The relational diagram shown in Figure P10.6 represents the main entities for ABC's database. Note the following important characteristics:

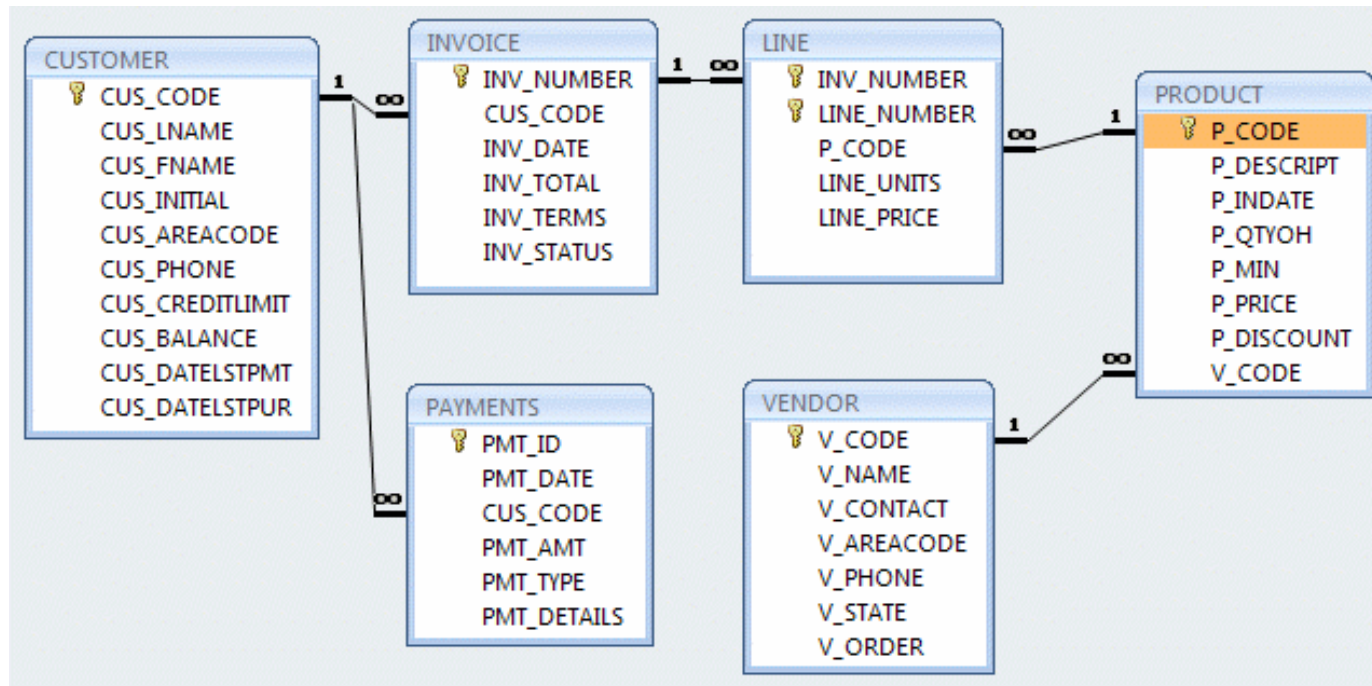
- A customer may make many purchases, each one represented by an invoice.
 - The CUS_BALANCE is updated with each credit purchase or payment and represents the amount the customer owes.
 - The CUS_BALANCE is increased (+) with every credit purchase and decreased (-) with every customer payment.
 - The date of last purchase is updated with each new purchase made by the customer.
 - The date of last payment is updated with each new payment made by the customer.
- An invoice represents a product purchase by a customer.
 - An INVOICE can have many invoice LINES, one for each product purchased.
 - The INV_TOTAL represents the total cost of invoice including taxes.
 - The INV_TERMS can be "30," "60," or "90" (representing the number of days of credit) or "CASH," "CHECK," or "CC."
 - The invoice status can be "OPEN," "PAID," or "CANCEL."
- A product's quantity on hand (P_QTYOH) is updated (decreased) with each product sale.
- A customer may make many payments. The payment type (PMT_TYPE) can be one of the following:
 - "CASH" for cash payments.
 - "CHECK" for check payments
 - "CC" for credit card payments
- The payment details (PMT_DETAILS) are used to record data about check or credit card payments:
 - The bank, account number, and check number for check payments
 - The issuer, credit card number, and expiration date for credit card payments.

Note: Not all entities and attributes are represented in this example. Use only the attributes indicated.

Using this database, write the SQL code to represent each one of the following transactions. Use BEGIN TRANSACTION and COMMIT to group the SQL statements in logical transactions.



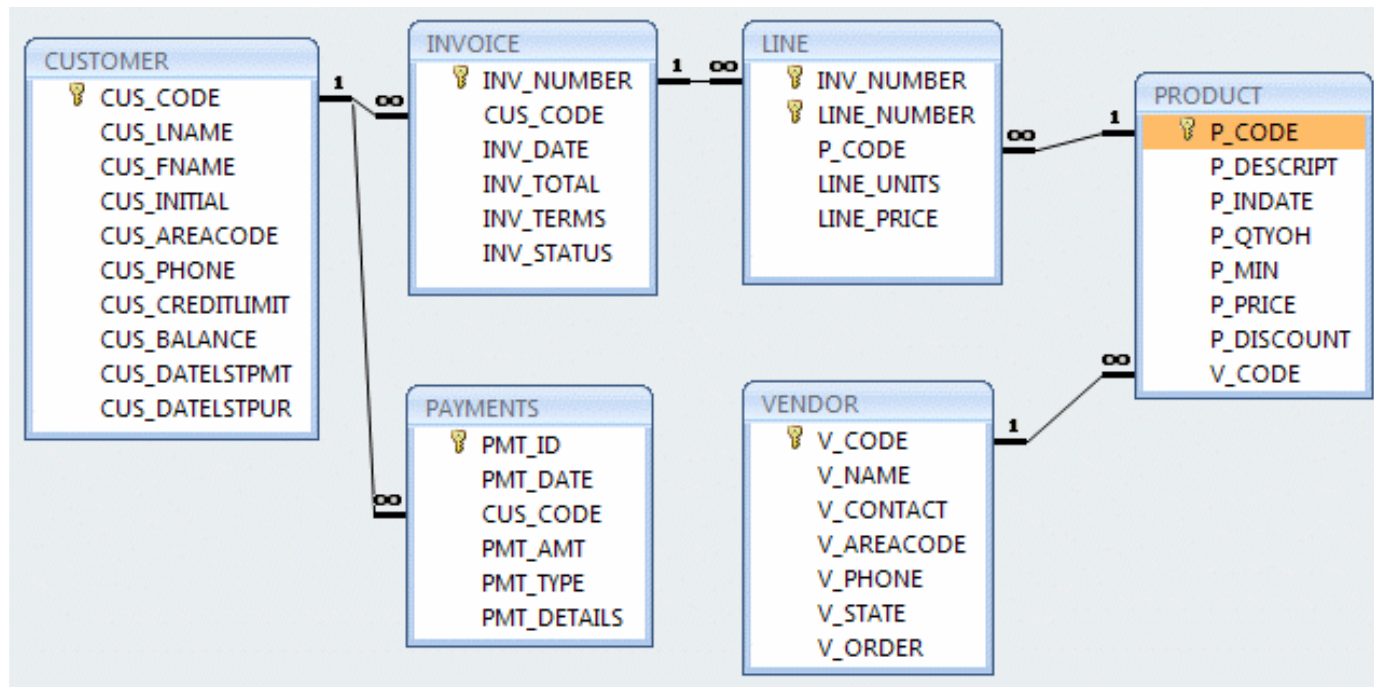
On May 11, 2016, customer '10010' makes a credit purchase (30 days) of one unit of product '11QER/31' with a unit price of \$110.00; the tax rate is 8 percent. The invoice number is 10983, and this invoice has only one product line.



On May 11, 2016, customer '10010' makes a credit purchase (30 days) of one unit of product '11QER/31' with a unit price of \$110.00; the tax rate is 8 percent. The invoice number is 10983, and this invoice has only one product line.

- a. BEGIN TRANSACTION
- b. INSERT INTO INVOICE
 - i. VALUES (10983, '10010', '11-May-2016', 118.80, '30', 'OPEN');
- c. INSERT INTO LINE
 - i. VALUES (10983, 1, '11QER/31', 1, 110.00);
- d. UPDATE PRODUCT
 - i. SET P_QTYOH = P_QTYOH - 1
 - ii. WHERE P_CODE = '11QER/31';
- e. UPDATE CUSTOMER
- f. SET CUS_DATELSTPUR = '11-May-2016', CUS_BALANCE = CUS_BALANCE + 118.80
- g. WHERE CUS_CODE = '10010';
- h. COMMIT;

On June 3, 2016, customer '10010' makes a payment of \$100 in cash. The payment ID is 3428.



On June 3, 2016, customer '10010' makes a payment of \$100 in cash. The payment ID is 3428.

```
a. BEGIN TRANSACTION
b. INSERT INTO PAYMENTS
    VALUES (3428, '03-Jun-2016', '10010', 100.00, 'CASH', 'None');
UPDATE CUSTOMER;
SET CUS_DATELSTPMT = '03-Jun-2016', CUS_BALANCE = CUS_BALANCE -100.00
WHERE CUS_CODE = '10010';
COMMIT
```

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 6a.

On May 11, 2016, customer '10010' makes a credit purchase (30 days) of one unit of product '11QER/31' with a unit price of \$110.00; the tax rate is 8 percent. The invoice number is 10983, and this invoice has only one product line.

- a. BEGIN TRANSACTION
- b. INSERT INTO INVOICE
 - i. VALUES (10983, '10010', '11-May-2016', 118.80, '30', 'OPEN');
- c. INSERT INTO LINE
 - i. VALUES (10983, 1, '11QER/31', 1, 110.00);
- d. UPDATE PRODUCT
 - i. SET P_QTYOH = P_QTYOH - 1
 - ii. WHERE P_CODE = '11QER/31';
- e. UPDATE CUSTOMER
- f. SET CUS_DATELSTPUR = '11-May-2016', CUS_BALANCE = CUS_BALANCE + 118.80
- g. WHERE CUS_CODE = '10010';
- h. COMMIT;

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 6a.

Time	Action
1	Lock INVOICE
2	Insert row 10983 into INVOICE
3	Unlock INVOICE
4	Lock LINE
5	Insert tow 10983, 1 into LINE
6	Unlock LINE
7	Lock PRODUCT
8	Update PRODUCT 11QER/31, P_QTYOH from 47 to 46
9	Unlock PRODUCT
10	Lock CUSTOMER
11	Update CUSTOMER 10010, CUS_BALANCE from 345.67 to 464.47
12	Update CUSTOMER 10010, CUS_DATELSTPUR from 05-May-2016 to 11-May-2016
13	Unlock CUSTOMER

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 6a.

On May 11, 2016, customer '10010' makes a credit purchase (30 days) of one unit of product '11QER/31' with a unit price of \$110.00; the tax rate is 8 percent. The invoice number is 10983, and this invoice has only one product line.

- a. BEGIN TRANSACTION
- b. INSERT INTO INVOICE
 - i. VALUES (10983, '10010', '11-May-2016', 118.80, '30', 'OPEN');
- c. INSERT INTO LINE
 - i. VALUES (10983, 1, '11QER/31', 1, 110.00);
- d. UPDATE PRODUCT
 - i. SET P_QTYOH = P_QTYOH - 1
 - ii. WHERE P_CODE = '11QER/31';
- e. UPDATE CUSTOMER
- f. SET CUS_DATELSTPUR = '11-May-2016', CUS_BALANCE = CUS_BALANCE + 118.80
- g. WHERE CUS_CODE = '10010';
- h. COMMIT;

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 6a.

Time	Action
1	Lock INVOICE
2	Lock LINE
3	Lock PRODUCT
4	Lock CUSTOMER
5	Insert row 10983 into INVOICE
6	Insert tow 10983, 1 into LINE
7	Update PRODUCT 11QER/31, P_QTYOH from 47 to 46
8	Update CUSTOMER 10010, CUS_BALANCE from 345.67 to 464.47
9	Update CUSTOMER 10010, CUS_DATELSTPUR from 05-May-2016 to 11-May-2016
10	Unlock INVOICE
11	Unlock LINE
12	Unlock PRODUCT
13	Unlock CUSTOMER

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 6b.

On June 3, 2016, customer '10010' makes a payment of \$100 in cash. The payment ID is 3428.

```
BEGIN TRANSACTION
INSERT INTO PAYMENTS
VALUES (3428, '03-Jun-2016', '10010', 100.00, 'CASH', 'None');
UPDATE CUSTOMER;
SET CUS_DATELSTPMT = '03-Jun-2016', CUS_BALANCE = CUS_BALANCE -100.00
WHERE CUS_CODE = '10010';
COMMIT
```

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 6b.

Time	Action
1	Lock PAYMENT
2	Insert row 3428 into PAYMENT
3	Unlock PAYMENT
4	Lock CUSTOMER
5	Update CUSTOMER 10010, CUS_BALANCE from 464.47 to 364.47
6	Update CUSTOMER 10010, CUS_DATELSTPMT from 02-May-2016 to 03-Jun-2016
7	Unlock CUSTOMER

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 6b.

On June 3, 2016, customer '10010' makes a payment of \$100 in cash. The payment ID is 3428.

a. BEGIN TRANSACTION

b. INSERT INTO PAYMENTS

VALUES (3428, '03-Jun-2016', '10010', 100.00, 'CASH', 'None');

UPDATE CUSTOMER;

SET CUS_DATELSTPMT = '03-Jun-2016', CUS_BALANCE = CUS_BALANCE -100.00

WHERE CUS_CODE = '10010';

COMMIT

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 6b.

Time	Action
1	Lock PAYMENT
2	Lock CUSTOMER
3	Insert row 3428 into PAYMENT
4	Update CUSTOMER 10010, CUS_ BALANCE from 464.47 to 364.47
5	Update CUSTOMER 10010, CUS_ DATELSTPMT from 02-May-2016 to 03-Jun-2016
6	Unlock PAYMENT
7	Unlock CUSTOMER