

Library Database

Advanced Queries and Views

Curt Ireton

Library Database Requirements – Advanced Queries and Views

Curt Ireton

masc0771

Ed Johnson

masc0778

Requirement 1. Set date_due in Book_copy for checked_out books (the simpler way). Date_due is as follows: add to date_out 90 days for friends, 30 days for regular, 20 days for children. Show the 4 update commands and the resulting Book_copy table after updates (Lit_id, copynum, PersId, date_out, date_due, order by Lit_id, copynum). The type of update to use is the one at the bottom of page 92. Commit.

```
SQL> set echo on
SQL> set linesiz 200
SQL> set pagesiz 200
SQL> update Book_copy
  2 set date_due=date_out+90
  3 where PersId in
  4 (select PersId
  5 from Customer where cust_type='Friend');
```

13 rows updated.

```
SQL> update Book_copy
  2 set date_due=date_out+30
  3 where PersId in
  4 (select PersId
  5 from Customer where cust_type='R');
```

2 rows updated.

```
SQL> update Book_copy
  2 set date_due=date_out+20
  3 where PersId in
  4 (select PersId
  5 from Customer where cust_type='Child');
```

3 rows updated.

```
SQL> select Lit_Id, CopyNum, PersId, date_out, date_due
2 from Book_copy
3 order by Lit_Id, CopyNum;
```

```
LIT_  COPYNUM PER DATE_OUT  DATE_DUE
```

```
-----
1001      1
1001      2 001 02-FEB-08 02-MAY-08
1001      3
1002      1 010 30-JAN-08 19-FEB-08
1002      2 001 01-FEB-08 01-MAY-08
1002      3
1003      1
1003      2 009 04-FEB-08 04-MAY-08
1003      3 011 11-FEB-08 02-MAR-08
1004      1 009 09-JAN-08 08-APR-08
1004      2 001 19-FEB-08 19-MAY-08
1005      1 006 29-JAN-08 28-APR-08
1005      2 001 07-MAR-08 05-JUN-08
1005      3
1006      1 001 26-JAN-08 25-APR-08
1006      2
1006      3 005 25-MAR-08 14-APR-08
1007      1 009 09-JAN-08 08-APR-08
1007      2 008 25-MAR-08 24-APR-08
1008      1
1008      2 003 10-FEB-08 10-MAY-08
1008      3
1009      1 003 20-FEB-08 20-MAY-08
1010      1 009 12-FEB-08 12-MAY-08
1010      2
1011      1
1011      2 008 16-FEB-08 17-MAR-08
1012      1 009 13-FEB-08 13-MAY-08
```

28 rows selected.

```
SQL> commit;
Commit complete.
```

```
SQL> spool off;
```

Requirement 2. Assuming for this project that "today's" date is 01-MAY-2008, show all overdue BOOK_copies. The output will show bookid, copy_num, customer last name and first name, num of days overdue. Order by bookid, copy_num. A BOOK_COPY is overdue when the date_due attribute precedes (is smaller than) 01-MAY-2008; you should compute the difference to_date('01-MAY-2008') - date_due. The difference of two dates is a number of days; the to_date function transforms a data of type VARCHAR2 into a data of type DATE. PLEASE, DO NOT update the BALANCE_DUEs in this Requirement.

```
SQL> set echo on
SQL> set linesiz 200
SQL> set pagesiz 200
SQL> select Lit_Id, CopyNum, lname, fname, to_date('01-MAY-2008')-date_due
 2 from Book_copy, Customer
 3 where Book_copy.PersId=Customer.PersId and date_due<'01-MAY-2008'
 4 order by Lit_Id, CopyNum;
```

LIT_	COPYNUM	LNAME	FNAME	TO_DATE('01-MAY-2008')-DATE_DUE
1002	1	RodGreg	Tina	72
1003	3	Castro	Andy	60
1004	1	Celine	Rachel	23
1005	1	Midler	Greg	3
1006	1	Ireton	Ron	6
1006	3	Tatum	Dina	17
1007	1	Celine	Rachel	23
1007	2	Midler	Darren	7
1011	2	Midler	Darren	45

9 rows selected.

```
SQL> spool off
```

Requirement 3. List each child customer (by lname, fname) with his/her sponsor (also by lname, fname). (this SQL command will use the join of a table with itself, similar to the SQL command shown page 85).

```
SQL> set echo on
SQL> set linesiz 200
SQL> set pagesiz 200
SQL> select e1.lname, e1.fname, e2.lname, e2.fname
 2 from Customer e1, Customer e2
 3 where e1.Spons_id=e2.PersId;
```

LNAME	FNAME	LNAME	FNAME
Ireton	Bil	Ireton	Ron
Rivers	Jenny	Clooney	Marie
Tatum	Dina	Clooney	Marie
RodGreg	Tina	Celine	Rachel
Castro	Andy	Celine	Rachel

```
SQL> spool off
```

Requirement 4. Show how many requests and how many copies exist for each book **that is requested**. Show title, number of requests, number of copies of the book. Show in order by title. The SQL query for this Requirement mixes a join with count. To understand such a query, be aware that the join must be processed before the count function. Be also aware that the join may create duplicate data, so that you should use the count(distinct ..) form of count where needed.

```
SQL> set echo on
SQL> set pagesiz 200
SQL> set linesiz 200
SQL> select Book.Btitle "Title",
2 count(distinct Book_copy.CopyNum) "# of Copies",
3 count(distinct Request.Rdate) "# of Requests"
4 from Book, Book_copy, Request
5 where Book.Lit_Id=Book_copy.Lit_Id
6 and Book.Lit_Id=Request.Lit_Id
7 group by Btitle
8 order by Btitle;
```

Title	# of Copies	# of Requests
0-0 Analysis	2	2
CRM Basics	3	1
Dating Clients	2	1
Dirt Road	2	1
Justine	1	2

```
SQL> spool off
```

Requirement 5. List the persid of friend customers who have more than two book_copies checked out, together with the number of book_copies checked-out (use HAVING, p. 73)

```
SQL> set echo on
SQL> set pagesiz 200
SQL> set linesiz 200
SQL> select PersId, (count(date_out)-count(time_due)) from Book_copy
 2 where PersId in
 3 (select PersId from Customer
 4 where cust_type in ('Friend'))
 5 group by PersId
 6 having (count(date_out)-count(time_due))>2;
```

```
PER (COUNT(DATE_OUT)-COUNT(TIME_DUE))
```

```
-----
001                5
009                5
```

```
SQL> spool off
```

Requirement 6. Customer Greg Middler (Persid = 006) returns the books he checked out before moving out to Amsterdam. The librarian will remove all his data from the database. Explain and show in order the SQL commands that the librarian will perform. Do not commit those changes. (+2 extra points if you indicate commands that may affect other customers as a result of the commands affecting Greg Middler).

```
SQL> set echo on
SQL> set pagesiz 200
SQL> set linesiz 200
SQL> update Book_copy
  2 set date_due=null
  3 where PersId='006';
```

1 row updated.

This affects other customers by making books available for checkout.

```
SQL> update Customer
  2 set balance_due=null
  3 where PersId='006';
```

1 row updated.

```
SQL> delete from Lecture where Speaker_Id='006';
```

1 row deleted.

```
SQL> delete from Request where PersId='006';
```

2 rows deleted.

This affects other customers by clearing these requests and making books more available to request.

```
SQL> delete from Book_copy where PersIdF='006';
```

1 row deleted.

This clears holds and makes books more readily available to other customers.

```
SQL> delete from Book_copy where PersId='006';
```

1 row deleted.

```
SQL> delete from Customer where PersId='006';
```


1 row deleted.

SQL> rollback;

Rollback complete.

SQL> spool off

Requirement 7. You would like to create a view showing all book copies checked out by friends. You would like to use this view for the following purpose, whenever possible:

1.) retrieve the data from the view: title of book, copy_num, persid of customer, lname, fname, date_out, date_due, balance_due

2) when a book is returned, modify the view to show that that book is no longer checked out

3) when a book is checked out by a friend, modify the view directly (not possible)

a.) create the view (when a view attribute exists in two tables used to create the view, make sure you get the attribute from the correct table). Explain how you go about it.

```
SQL> set echo on
```

```
SQL> set pagesiz 200
```

```
SQL> set linesiz 200
```

```
SQL> create view bookco
```

```
2 as select Book.Btitle, Book_copy.CopyNum,  
3 Customer.PersId, Customer.lname, Customer.fname,  
4 Book_copy.date_out, Book_copy.date_due,  
5 Customer.balance_due  
6 from Book, Book_copy, Customer  
7 where Book.Lit_Id=Book_copy.Lit_Id  
8 and Book_copy.PersId=Customer.PersId  
9 and Customer.cust_type in ('Friend');
```

View created.

```
SQL> spool off
```

I used attributes from three tables: Book, Book_copy, and Customer to create this view. I used the attribute PersId from the Customer table, so the data would not be redundant. Since the PersId of Book_copy (foreign key) matched the PersId of Customer (primary key), only the customer that had a book checked out would appear on the view.

b.) explain (asserting is not explaining) clearly what sort of association exists a) between the view and customer; b) between the view and book_copy (association could be 1-1, 1-M, M-M)

The association between the view and Customer is 1-M, so modifying the Customer attributes (PersId, balance_due, lname, fname) is not permitted on the view. The view and Book_copy have a 1-1 relationship, that is, each row is a unique Book_copy, so the attributes of table Book_copy can be modified on the view – however, if a book_copy is modified it will affect other tables, i.e. if a book is deleted (returned) it will remove the copy of that book from the data base.

c.) list the data in the view order by customer persid

```
SQL> set echo on
SQL> set pagesiz 200
SQL> set linesiz 200
SQL> select * from bookco
  2 order by PersId;
```

BTITLE	COPYNUM	PER	LNAME	FNAME	DATE_OUT	DATE_DUE	BALANCE_DUE
E-Business	2 001	Ireton	Ron	02-FEB-08	02-MAY-08		
CRM Basics	2 001	Ireton	Ron	01-FEB-08	01-MAY-08		
Java Cooking	2 001	Ireton	Ron	07-MAR-08	05-JUN-08		
Free Downloads	1 001	Ireton	Ron	26-JAN-08	25-APR-08		
Dirt Road	2 001	Ireton	Ron	19-FEB-08	19-MAY-08		
C# for All	2 003	Clooney	Marie	10-FEB-08	10-MAY-08		5.25
Easy Calculus	1 003	Clooney	Marie	20-FEB-08	20-MAY-08		5.25
Java Cooking	1 006	Midler	Greg	29-JAN-08	28-APR-08		1.25
Justine	1 009	Celine	Rachel	13-FEB-08	13-MAY-08		5.5
Managers	1 009	Celine	Rachel	12-FEB-08	12-MAY-08		5.5
0-0 Analysis	1 009	Celine	Rachel	09-JAN-08	08-APR-08		5.5
Dirt Road	1 009	Celine	Rachel	09-JAN-08	08-APR-08		5.5
Easy Java	2 009	Celine	Rachel	04-FEB-08	05-MAR-08		5.5

13 rows selected.

```
SQL> spool off
```

d.) Rachel Celine (009) returns her copy (copy 2) of book 1003. Show the sql command you execute on the view to achieve it. Which sql did you choose: delete, insert, update? Explain. Show the result on the view and the affected table(s). Rollback

```
SQL> set echo on
SQL> set pagesiz 200
SQL> set linesiz 200
SQL> update Book_copy
  2 set date_out=null
  3 where Lit_Id='1003' and CopyNum=2;
```

1 row updated.

```
SQL> update Book_copy
  2 set date_due=null
  3 where Lit_Id='1003' and CopyNum=2;
```

1 row updated.

```
SQL> update Book_copy
  2 set PersId=null
  3 where Lit_Id='1003' and CopyNum=2;
```

1 row updated.

```
SQL> select * from bookco
  2 order by PersId;
```

BTITLE	COPYNUM	PER	LNAME	FNAME	DATE_OUT	DATE_DUE	BALANCE_DUE
E-Business	2 001	Ireton	Ron		02-FEB-08	02-MAY-08	
CRM Basics	2 001	Ireton	Ron		01-FEB-08	01-MAY-08	
Java Cooking	2 001	Ireton	Ron		07-MAR-08	05-JUN-08	
Free Downloads	1 001	Ireton	Ron		26-JAN-08	25-APR-08	
Dirt Road	2 001	Ireton	Ron		19-FEB-08	19-MAY-08	
C# for All	2 003	Clooney	Marie		10-FEB-08	10-MAY-08	5.25
Easy Calculus	1 003	Clooney	Marie		20-FEB-08	20-MAY-08	5.25
Java Cooking	1 006	Midler	Greg		29-JAN-08	28-APR-08	1.25
Justine	1 009	Celine	Rachel		13-FEB-08	13-MAY-08	5.5
Managers	1 009	Celine	Rachel		12-FEB-08	12-MAY-08	5.5
0-0 Analysis	1 009	Celine	Rachel		09-JAN-08	08-APR-08	5.5
Dirt Road	1 009	Celine	Rachel		09-JAN-08	08-APR-08	5.5

12 rows selected.

```
SQL> select * from Book_copy
2 order by PersId;
```

```
LIT_  COPYNUM BOOK_TYP PER DATE_OUT DATE_DUE TIME_DUE PER HDATE
BNAMEP BNAMEC
```

```
-----
1002    2 regular 001 01-FEB-08 01-MAY-08          U_City
1005    2 regular 001 07-MAR-08 05-JUN-08          LJolla
1006    1 regular 001 26-JAN-08 25-APR-08          Mbeach
1004    2 regular 001 19-FEB-08 19-MAY-08          Mbeach
1001    2 regular 001 02-FEB-08 02-MAY-08          Mbeach
1008    2 regular 003 10-FEB-08 10-MAY-08          Mbeach
1009    1 regular 003 20-FEB-08 20-MAY-08          Mbeach
1006    3 refernce 005 25-MAR-08 14-APR-08    1600    U_City U_City
1005    1 regular 006 29-JAN-08 28-APR-08          Mbeach
1011    2 regular 008 16-FEB-08 17-MAR-08          Mbeach
1007    2 refernce 008 25-MAR-08 24-APR-08    1800    Mbeach Mbeach
1004    1 regular 009 09-JAN-08 08-APR-08          U_City
1010    1 regular 009 12-FEB-08 12-MAY-08          U_City
1012    1 regular 009 13-FEB-08 13-MAY-08          U_City
1007    1 regular 009 09-JAN-08 08-APR-08          LJolla
1002    1 regular 010 30-JAN-08 19-FEB-08          Mbeach
1003    3 regular 011 11-FEB-08 02-MAR-08          LJolla
1008    3 regular          006 15-APR-08 U_City U_City
1001    1 regular          Mbeach LJolla
1010    2 regular          003 15-APR-08 LJolla U_City
1011    1 regular          003 20-APR-08 Mbeach LJolla
1001    3 regular          009 15-APR-08 U_City U_City
1008    1 regular          001 20-APR-08 Mbeach Mbeach
1006    2 regular          Mbeach Mbeach
1002    3 refernce          LJolla LJolla
1005    3 regular          Mbeach LJolla
1003    2 regular          Mbeach
1003    1 regular          007 29-JAN-08 Mbeach LJolla
```

28 rows selected.

```
SQL> select * from Customer
2 order by PersId;
```

```
PER LNAME      FNAME BALANCE_DUE CUST_TY BNAME SPO
-----
001 Ireton     Ron      Friend Mbeach
002 Ireton     Bil      2.6 Child Mbeach 001
003 Clooney    Marie    5.25 Friend U_City
004 Rivers     Jenny    4.5 Child U_City 003
005 Tatum      Dina     3.1 Child U_City 003
006 Midler     Greg     1.25 Friend LJolla
007 Midler     Will     1.75 Friend LJolla
008 Midler     Darren   3 R      LJolla
009 Celine     Rachel   5.5 Friend U_City
010 RodGreg    Tina     10 Child U_City 009
011 Castro     Andy     Child U_City 009
```

11 rows selected.

```
SQL> rollback;
```

Rollback complete.

```
SQL> spool off
```

At first I tried to delete the checkout from the view and was successful, however it deleted the book copy from the entire database, so I rolled back the deletion. Since PersId is a primary key and CopyNum is a foreign key, I couldn't change those attributes by updating the view, so I updated the base table Book_copy and was successful at keeping the copy in the database. I did not update the balance due from the Customer table as Celine had more than one book checked out.

e.) Rachel Celine (009) checks out copy 3 of 1005 on 01-MAY-2008. Why is it not possible that a sql command be executed on the view to perform this action? Explain.
Rollback

```
SQL> set echo on
SQL> set pagesiz 200
SQL> set linesiz 200
SQL> insert into bookco
  2 values ('Java Cooking', 3, '009', 'Celine', 'Rachel',
  3 '01-MAY-2008', '31-MAY-2008', 5.5);
insert into bookco
*
```

```
ERROR at line 1:
ORA-01779: cannot modify a column which maps to a non key-preserved table
```

```
SQL> rollback;
```

Rollback complete.

```
SQL> spool off
```

I was unable to update the table because there is not a 1-1 association between Customer and the view, the same values from the table Customer may appear in several rows of the view.

f.) can you set the balance_due of a customer to 0 on the view? Explain.

```
SQL> set echo on
SQL> set pagesiz 200
SQL> set linesiz 200
SQL> update bookco
  2 set balance_due=0
  3 where PersId='009';
set balance_due=0
*
```

```
ERROR at line 2:
ORA-01779: cannot modify a column which maps to a non key-preserved table
```

```
SQL> spool off
```

I was unable to update the balance from the view because there is not a 1-1 relationship between the table Customer and the view bookco. The same balance_due may appear in several rows of the view.

g.) Rachel Celine (009) changes her status from friend to regular customer. Update the appropriate table and show the effect on the view. Explain. Rollback.

```
SQL> set echo on
SQL> set pagesiz 200
SQL> set linesiz 200
SQL> update Customer
  2 set cust_type='R'
  3 where PersId='009';
```

1 row updated.

```
SQL> select * from bookco
  2 order by PersId;
```

BTITLE	COPYNUM	PER	LNAME	FNAME	DATE_OUT	DATE_DUE	BALANCE_DUE
E-Business	2 001	Ireton	Ron	02-FEB-08	02-MAY-08		
CRM Basics	2 001	Ireton	Ron	01-FEB-08	01-MAY-08		
Dirt Road	2 001	Ireton	Ron	19-FEB-08	19-MAY-08		
Java Cooking	2 001	Ireton	Ron	07-MAR-08	05-JUN-08		
Free Downloads	1 001	Ireton	Ron	26-JAN-08	25-APR-08		
Easy Calculus	1 003	Clooney	Marie	20-FEB-08	20-MAY-08		5.25
C# for All	2 003	Clooney	Marie	10-FEB-08	10-MAY-08		5.25
Java Cooking	1 006	Midler	Greg	29-JAN-08	28-APR-08		1.25

8 rows selected.

```
SQL> rollback;
```

Rollback complete.

```
SQL> spool off
```

I updated the table Customer to change Celine from Friend to Regular ('R') customer. It deleted her data from the view because the view was designed to permit only 'Friend' customers.

Requirement 8. GRANT command

Requirements 8 and 9 require you to work in teams of 2 from your two ORACLE accounts. Open two sessions at the same time from one UNIX account. You can do it executing SSH secure shell twice for the same UNIX account

Log on from session 1 to one ORACLE account (which must have the database data).

Log on from session 2 to the second ORACLE account (it needs not to have the database). Be very careful to use two different spool file names in these two ORACLE accounts, since both files will be saved in the same rohan/unix account.

a. (2) In ORACLE account 1, create a table called CLIENT as a copy of CUSTOMER, with attributes PERSID, Lname, Fname, Balance_due. Before any GRANT are issued, try to do a select on CLIENT from the second account. Note the following syntax to access tables stored in one ORACLE account from another ORACLE account. When accessing the table called CLIENT stored in account insc444 from a different account, you must use the fully qualified table name: insc444.CLIENT (i.e. : SELECT * FROM insc444.CLIENT;). Show what happens and explain.

b. (2) Grant the following permission to the second account from the first one on the table CLIENT:

grant select, update(balance_due) on CLIENT to second_account (use the actual user name)

Do a select on CLIENT from the second account. Do an update on balance_due from the second account. Do an update on last name. What happens each time? **Explain each time.** Rollback.

masc0771 Curt Ireton

masc0778 Ed Johnson

masc0771

```
SQL> set echo on
```

```
SQL> set pagesiz 200
```

```
SQL> set linesiz 200
```

```
SQL> create table Client as
```

```
2 select PersId, lname, fname, balance_due from Customer;
```

Table created.

```
SQL> spool off
```

masc0778

```
SQL> set echo on
SQL> set linesiz 200
SQL> select * from masc0771.Client;
select * from masc0771.Client
      *
```

```
ERROR at line 1:
ORA-01031: insufficient privileges
```

Since no privileges were granted to masc0778, he was unable to access the account.

masc0771

```
SQL> set echo on
SQL> set pagesiz 200
SQL> set linesiz 200
SQL> grant select, update(balance_due)
  2 on Client
  3 to masc0778;
```

Grant succeeded.

The privileges of select and update (balance_due) were granted to masc0778.

masc0778

```
SQL> select * from masc0771.Client;
```

```
PER LNAME      FNAME BALANCE_DUE
```

```
-----  
001 Ireton     Ron  
002 Ireton     Bil      2.6  
003 Clooney    Marie    5.25  
004 Rivers     Jenny    4.5  
005 Tatum      Dina     3.1  
006 Midler     Greg     1.25  
007 Midler     Will     1.75  
008 Midler     Darren   3  
009 Celine     Rachel   5.5  
010 RodGreg    Tina     10  
011 Castro     Andy
```

11 rows selected.

Since masc0778 now has privileges he can access the data.

masc0778

```
SQL> update masc0771.Client  
2 set balance_due = balance_due + 10;
```

11 rows updated.

masc0778

```
SQL> select * from masc0771.Client;
```

```
PER LNAME      FNAME BALANCE_DUE
```

```
-----  
001 Ireton     Ron  
002 Ireton     Bil      12.6  
003 Clooney    Marie    15.25  
004 Rivers     Jenny    14.5  
005 Tatum      Dina     13.1  
006 Midler     Greg     11.25  
007 Midler     Will     11.75  
008 Midler     Darren   13  
009 Celine     Rachel   15.5  
010 RodGreg    Tina     20  
011 Castro     Andy
```

11 rows selected.

masc0778 can update balance_due on the view because he has permission.

masc0778

```
SQL> update masc0771.Client
```

```
  2 set lname=null;
```

```
update masc0771.Client
```

```
*
```

ERROR at line 1:

ORA-01031: insufficient privileges

Since no permission to update customer last name was given to masc0771, he was denied access.

masc0778

```
SQL> rollback;
```

Rollback complete.

masc0771

```
SQL> rollback;
```

Rollback complete.

Requirement 9. GRANT, COMMIT, LOCK, ROLLBACK, IMPLICIT LOCK. NO credit given to Requirements for which no valid explanations of what happens are provided and that are not properly presented as explained below). Other remarks of Requirement 8 apply also to Requirement 9.

Show all your work and comment it in an appropriate fashion.

- a. Do a GRANT all on CLIENT To account2.
- b. From account2, verify that you can access the CLIENT table located in account1 (e.g. by a select)
- c. from account 1, execute an update on CLIENT: increase all balance_dues by \$300, including when the value is 0 or null. Show the result.
- d. from account2, do a select on same CLIENT table. What do you notice that seems abnormal or in error. Explain.
- e. from account 1, execute the SQL command "COMMIT";
- f. from account 2, do a select again on CLIENT. What is different from Requirement d? Explain
- g. from account1, repeat the command c again.
- h. from account2, execute the command for updating balance_due by 50. What happens? (the system is frozen). Explain in terms of "implicit" lock.
- i. from account 1, execute COMMIT;
- j. what do you notice in account 2. Do a select from account2. Explain. Do a COMMIT.
- k. from account 1, execute the command:
lock table CLIENT in exclusive mode;
- l. from account2, do a select on the same table.
- m. from account 2 update balance_due (increase balance_due by 50). What do you notice?
Explain both m and l
- n. From account 1, set balance_dues to 10. Show the result by a select.
- o. What do you notice in account 2? Explain
- p. Do a COMMIT in account 1
- q. What do you notice in account2; explain. Do a select on same table. What do you notice in the results? Explain.

a.)

masc0771

```
SQL> set echo on
SQL> set pagesiz 200
SQL> set linesiz 200
SQL> grant all on Client to masc0778;
```

Grant succeeded.

b.)

masc0778

```
SQL> set echo on
SQL> set linesiz 200
SQL> set pagesiz 200
SQL> select * from masc0771.Client;
```

PER	LNAME	FNAME	BALANCE_DUE
001	Ireton	Ron	
002	Ireton	Bil	2.6
003	Clooney	Marie	5.25
004	Rivers	Jenny	4.5
005	Tatum	Dina	3.1
006	Midler	Greg	1.25
007	Midler	Will	1.75
008	Midler	Darren	3
009	Celine	Rachel	5.5
010	RodGreg	Tina	10
011	Castro	Andy	

11 rows selected.

c.)

masc0771

```
SQL> update Client
```

```
2 set balance_due=nvl(balance_due,0)+300;
```

```
11 rows updated.
```

```
SQL> select * from Client;
```

```
PER LNAME      FNAME BALANCE_DUE
```

```
-----  
001 Ireton     Ron      300  
002 Ireton     Bil      302.6  
003 Clooney    Marie    305.25  
004 Rivers     Jenny    304.5  
005 Tatum      Dina     303.1  
006 Midler     Greg     301.25  
007 Midler     Will     301.75  
008 Midler     Darren   303  
009 Celine     Rachel   305.5  
010 RodGreg    Tina     310  
011 Castro     Andy     300
```

```
11 rows selected.
```

d.)

masc0778

```
SQL> select * from masc0771.Client;
```

PER	LNAME	FNAME	BALANCE_DUE
001	Ireton	Ron	
002	Ireton	Bil	2.6
003	Clooney	Marie	5.25
004	Rivers	Jenny	4.5
005	Tatum	Dina	3.1
006	Midler	Greg	1.25
007	Midler	Will	1.75
008	Midler	Darren	3
009	Celine	Rachel	5.5
010	RodGreg	Tina	10
011	Castro	Andy	

```
11 rows selected.
```

The balance due did not change in this account

e.)

masc0772

```
SQL> commit;
```

```
Commit complete.
```


f.)

masc0778

```
SQL> select * from masc0771.Client;
```

PER	LNAME	FNAME	BALANCE_DUE
001	Ireton	Ron	300
002	Ireton	Bil	302.6
003	Clooney	Marie	305.25
004	Rivers	Jenny	304.5
005	Tatum	Dina	303.1
006	Midler	Greg	301.25
007	Midler	Will	301.75
008	Midler	Darren	303
009	Celine	Rachel	305.5
010	RodGreg	Tina	310
011	Castro	Andy	300

11 rows selected.

The balances have changed because the changes were committed from masc0771, so the database was updated.

g.)

masc0771

```
SQL> update Client  
2 set balance_due=nvl(balance_due,0)+300;
```

11 rows updated.

h.)

masc0778

```
SQL> update masc0771.Client  
2 set balance_due=nvl(balance_due,0)+50;
```

The system locked on masc0778 because the operation was not completed. There was still data waiting to be committed from masc0771

i.)

masc0771

```
SQL> commit;
```

Commit complete.

j.)

masc0778

11 rows updated.

```
SQL> select * from masc0771.Client;
```

PER	LNAME	FNAME	BALANCE_DUE
001	Ireton	Ron	650
002	Ireton	Bil	652.6
003	Clooney	Marie	655.25
004	Rivers	Jenny	654.5
005	Tatum	Dina	653.1
006	Midler	Greg	651.25
007	Midler	Will	651.75
008	Midler	Darren	653
009	Celine	Rachel	655.5
010	RodGreg	Tina	660
011	Castro	Andy	650

11 rows selected.

```
SQL> commit;
```

Commit complete.

Once masc0771 committed the data, the implicit lock was removed and masc0778's data was not only available, but was updated again.

k.)

masc0771

```
SQL>
```

```
SQL> lock table Client in exclusive mode;
```

Table(s) Locked.

l.)

masc0778

```
SQL> select * from masc0771.Client;
```

PER	LNAME	FNAME	BALANCE_DUE
001	Ireton	Ron	650
002	Ireton	Bil	652.6
003	Clooney	Marie	655.25
004	Rivers	Jenny	654.5
005	Tatum	Dina	653.1
006	Midler	Greg	651.25
007	Midler	Will	651.75
008	Midler	Darren	653
009	Celine	Rachel	655.5
010	RodGreg	Tina	660
011	Castro	Andy	650

```
11 rows selected.
```

m.)

masc0778

```
SQL> update masc0771.Client
```

```
2 set balance_due=nvl(balance_due,0)+50;
```

masc0778nwas allowed to query the data but when he tried to update it, he was locked out.

n.)

masc0771

SQL> update Client

2 set balance_due=10;

11 rows updated.

SQL> select * from Client;

PER	LNAME	FNAME	BALANCE_DUE
001	Ireton	Ron	10
002	Ireton	Bil	10
003	Clooney	Marie	10
004	Rivers	Jenny	10
005	Tatum	Dina	10
006	Midler	Greg	10
007	Midler	Will	10
008	Midler	Darren	10
009	Celine	Rachel	10
010	RodGreg	Tina	10
011	Castro	Andy	10

11 rows selected.

o.)

masc0778

Still nothing has happened.

p.)

masc0771

SQL> commit;

Commit complete.

q.)

masc0778

11 rows updated.

SQL> select * from masc0771.Client;

PER	LNAME	FNAME	BALANCE_DUE
001	Ireton	Ron	60
002	Ireton	Bil	60
003	Clooney	Marie	60
004	Rivers	Jenny	60
005	Tatum	Dina	60
006	Midler	Greg	60
007	Midler	Will	60
008	Midler	Darren	60
009	Celine	Rachel	60
010	RodGreg	Tina	60
011	Castro	Andy	60

11 rows selected.

When masc0771 committed masc0778 was unlocked and the balance due was set first to 10 from masc0771 and then updated by 50 from masc0778.

masc0771

SQL> spool off;

masc0778

SQL> spool off;