Uppsala University

Department of Information Technology

Database Design II - 1DL400

Assignment 1 - Database Integrity: Assertions, Triggers, and Stored Procedures

Group 9: Efthymia Chantzi[1] || Huijie Shen[2] || Eleftherios Anagnostopoulos[3]

February 4, 2015

Submitted Source Files

ex3.sql | ex4.sql | ex5.sql | ex6.sql | ex7.sql

Exercises 1 & 2

Our first steps included the setup of *"JonsonBrothers"* database by following the instructions on the course home page:

```
mysql> CREATE DATABASE JohnsonBrothers;
mysql> CONNECT JohnsonBrothers;
mysql> SOURCE JohnsonSchema.sql;
mysql> SOURCE JohnsonData.sql;
mysql> SOURCE extension.sql;
```

Thus, the following tables and views have been created and added to the database:

```
mysql> SHOW TABLES;
```

Tables_in_JohnsonBrothers: | account | carries | city | customer | dav (view) | debit | dept | employee | item | manager | months | parts | sale | sale_supply (view) | store | supplier | supply | transact

[1] Efthymia.Chantzi.0787@student.uu.se
[2] Huijie.Shen.6288@student.uu.se
[3] Eleftherios.Anagnostopoulos.8127@student.uu.se

Exercise 3

File: ex3.sql

Procedure: createCustomer

Inputs: name, street address, city

Description: Procedure for creating and adding customers to the database. It must be mentioned that selected city must be stored at the database, since there is an added foreign key constraint for the attribute *city* of table *customer* related to the attribute *name* of table *city*.

mysql> SELECT name FROM city;

name: | Amherst | Atlanta | Boston | Dallas | Denver | El Cerrito | Hickville | Los Angeles | Madison | New York | Oakland | Paxton | Salt Lake City | San Diego | San Francisco | Seattle | White Plains

mysql> SELECT * FROM customer;

number	name	address	city
1	John Smith	Park Row, 1	New York

Table 1: customer

mysql> SOURCE ex3.sql;

mysql> CALL createCustomer('Efthymia', 'Sunset Boulevard, 1', 'Los Angeles');

mysql> CALL createCustomer('Huijie', 'Park Avenue, 1', 'New York');

mysql> CALL createCustomer('Eleftherios', 'Newbury Street, 1', 'Boston');

mysql> SELECT * FROM customer;

number	name	address	city
1	John Smith	Park Row, 1	New York
2	**Efthymia**	**Sunset Boulevard, 1**	**Los Angeles**
3	**Huijie**	**Park Avenue, 1**	**New York**
4	**Eleftherios**	**Newbury Street, 1**	**Boston**

Table 2: customer → New customers have been added to the database

Exercise 4

File: ex4.sql

Procedure: createAccount

Inputs: customer_number, credit

Description: Procedure for creating a new customer account. **If credit is less than zero or selected customer does not exist, then new account is aborted and an exception is thrown.**

mysql> SELECT * FROM account;

number	customer	balance	credit
11652133	1	0	0
12591815	1	0	0
14096831	1	0	0
14356540	1	0	0

Table 3: account

mysql> SOURCE ex4.sql;

mysql> CALL createAccount(0, 10);

ERROR 1644 (ERROR): Procedure: createAccount - There is no customer with this number.

mysql> CALL createAccount(1, -1);

ERROR 1644 (ERROR): Procedure: createAccount - Credit must be greater or equal to zero.

mysql> CALL createAccount(2, 200);

mysql> CALL createAccount(3, 300);

mysql> CALL createAccount(4, 400);

mysql> SELECT * FROM account;

number	customer	balance	credit
11652133	1	0	0
12591815	1	0	0
14096831	1	0	0
14356540	1	0	0
14356541	**2**	**0**	**200**
14356542	**3**	**0**	**300**
14356543	**4**	**0**	**400**

Table 4: account → New accounts have been added to the database

Exercise 5

File: ex5.sql

Procedure: depositMoney

Inputs: account_number, amount

Description: Procedure for depositing money into a customers account. The procedure should take an account number and a positive amount as arguments. **If amount is not positive or selected account does not exist, then the deposit is aborted and an exception is thrown.**

mysql> SOURCE ex5.sql;

mysql> CALL depositMoney(14356541, 0);

ERROR 1644 (ERROR): Procedure: depositMoney - Input amount must be positive.

mysql> CALL depositMoney(0, 100);

ERROR 1644 (ERROR): Procedure: depositMoney - There is no account with this number.

mysql> CALL depositMoney(14356541, 2000);

mysql> CALL depositMoney(14356542, 3000);

mysql> CALL depositMoney(14356543, 4000);

mysql> SELECT * FROM account;

number	customer	balance	credit
11652133	1	0	0
12591815	1	0	0
14096831	1	0	0
14356540	1	0	0
14356541	2	**2000**	200
14356542	3	**3000**	300
14356543	4	**4000**	400

Table 5: account → Balance entries have been correctly modified

Exercise 6

File: ex6.sql

Procedure: finalizeSale

Inputs: transaction_number

Description: Procedure that finalizes a sale given by its transaction number. The procedure charges customer's account the total cost of the sale, reduces qoh of the sold items (*carries* table) and removes the corresponding entries from tables *transact*, *debit* and *sale*. Uses a view in order to calculate the total cost of the sale (as suggested) and finally, **in case of NULL transaction account, the sale is aborted and an exception is thrown.**

mysql> SELECT * FROM transact;

number	date	time	employee	account	amount
100581	1995-01-15	NULL	157	NULL	2050
100582	1995-01-15	NULL	1110	14356540	1000
100586	1995-01-16	NULL	35	14096831	13446
100592	1995-01-17	NULL	129	NULL	650
100593	1995-01-18	NULL	13	11652133	430
100594	1995-01-18	NULL	215	12591815	3295

Table 6: transact

mysql> SELECT * FROM sale;

debit	item	quantity	dept
100581	118	5	26
100581	120	1	26
100582	26	1	14
100586	106	2	1
100586	127	3	65
100592	258	1	58
100593	23	2	10
100594	52	1	60

Table 7: sale

mysql> SELECT * FROM carries;

item	dept	qoh
1	14	220
11	1	575
19	43	600
21	1	405
21	49	120
23	10	100
25	10	75
26	14	20
43	49	200
52	60	300
101	28	125
101	63	325
101	70	225
106	1	175
106	49	150
107	35	225
115	14	10
118	26	1000
119	49	400
120	26	750
121	26	600
127	65	125
165	65	500
258	58	1200
301	43	500
301	73	100

Table 8: carries

mysql> SELECT * FROM item;

number	name	dept	price	supplier
11	Wash Cloth	63	75	213
19	Bellbottoms	63	450	33
21	ABC Blocks	43	198	125
23	1 lb Box	34	215	42
25	2 lb Box, Mix	34	450	42
26	Earrings	14	1000	199
43	Maze	43	325	89
52	Jacket	60	3295	15
101	Slacks	58	1600	15
106	Clock Book	43	198	125
107	The ‘Feel’ Book	35	225	89
115	Gold Ring	14	4995	199
118	Towels, Bath	26	250	213
119	Squeeze Ball	1	250	89
120	Twin Sheet	26	800	213
121	Queen Sheet	26	1375	213
127	Ski Jumpsuit	60	4350	15
165	Jean	60	825	33
258	Shirt	58	650	33
301	Boy’s Jean Suit	43	1250	33

Table 9: item

mysql> SELECT * FROM debit;

number	sdate	employee	account
100581	1995-01-15	157	NULL
100582	1995-01-15	1110	14356540
100586	1995-01-16	35	14096831
100592	1995-01-17	129	NULL
100593	1995-01-18	13	11652133
100594	1995-01-18	215	12591815

Table 10: debit

In the beginning, it is important to create a view in order to be used in the calculation of the total cost for each sale (as suggested):

mysql> CREATE VIEW v AS SELECT debit, item, quantity, dept, (SELECT price FROM item WHERE number = item) AS price FROM sale;

mysql> SELECT * FROM v;

debit	item	quantity	dept	price
100581	118	5	26	250
100581	120	1	26	800
100582	26	1	14	1000
100586	106	2	1	198
100586	127	3	65	4350
100592	258	1	58	650
100593	23	2	10	215
100594	52	1	60	3295

Table 11: v → A new view has been created

Using this view, the total cost of a particular sale can now be calculated. For example:

mysql> SELECT SUM(quantity * price) AS Total_Cost_of_100581 FROM v WHERE debit = 100581;

Total_Cost_of_100581: 2050

This can also be confirmed by the table *transact*:

mysql> SELECT amount AS Confirmation FROM transact WHERE number = 100581;

Confirmation: 2050

Furthermore, we will connect two of the existing transactions and debits with two of our new accounts:

mysql> UPDATE transact SET account = 14356543 WHERE number = 100581;
mysql> UPDATE debit SET account = 14356543 WHERE number = 100581;
mysql> UPDATE transact SET account = 14356542 WHERE number = 100592;
mysql> UPDATE debit SET account = 14356542 WHERE number = 100592;

mysql> SELECT * FROM transact;

number	date	time	employee	account	amount
100581	1995-01-15	NULL	157	**14356543**	2050
100582	1995-01-15	NULL	1110	14356540	1000
100586	1995-01-16	NULL	35	14096831	13446
100592	1995-01-17	NULL	129	**14356542**	650
100593	1995-01-18	NULL	13	11652133	430
100594	1995-01-18	NULL	215	12591815	3295

Table 12: transact → Modified accounts for transactions 100581 and 100592

mysql> SELECT * FROM debit;

number	sdate	employee	account
100581	1995-01-15	157	**14356543**
100582	1995-01-15	1110	14356540
100586	1995-01-16	35	14096831
100592	1995-01-17	129	**14356542**
100593	1995-01-18	13	11652133
100594	1995-01-18	215	12591815

Table 13: debit → Modified accounts for debits 100581 and 100592

So, sales **100581** and **100592** can now be finalized:

```
mysql> SOURCE ex6.sql;
mysql> CALL finalizeSale(100581);
mysql> CALL finalizeSale(100592);
mysql> SELECT * FROM account;
```

number	customer	balance	credit
11652133	1	0	0
12591815	1	0	0
14096831	1	0	0
14356540	1	0	0
14356541	2	2000	200
14356542	3	**2350**	300
14356543	4	**1950**	400

Table 14: account → Balance of corresponding accounts has been correctly modified

```
mysql> SELECT * FROM transact;
```

number	date	time	employee	account	amount
100582	1995-01-15	NULL	1110	14356540	1000
100586	1995-01-16	NULL	35	14096831	13446
100593	1995-01-18	NULL	13	11652133	430
100594	1995-01-18	NULL	215	12591815	3295

Table 15: transact → Transactions 100581 and 100592 have been deleted

mysql> SELECT * FROM sale;

debit	item	quantity	dept
100582	26	1	14
100586	106	2	1
100586	127	3	65
100593	23	2	10
100594	52	1	60

Table 16: sale → Sales 100581 and 100592 have been deleted

mysql> SELECT * FROM debit;

number	sdate	employee	account
100582	1995-01-15	1110	14356540
100586	1995-01-16	35	14096831
100593	1995-01-18	13	11652133
100594	1995-01-18	215	12591815

Table 17: debit → Debits 100581 and 100592 have been deleted

mysql> SELECT * FROM carries;

item	dept	qoh
1	14	220
11	1	575
19	43	600
21	1	405
21	49	120
23	10	100
25	10	75
26	14	20
43	49	200
52	60	300
101	28	125
101	63	325
101	70	225
106	1	175
106	49	150
107	35	225
115	14	10
118	26	**995**
119	49	400
120	26	**749**
121	26	600
127	65	125
165	65	500
258	58	**1199**
301	43	500
301	73	100

Table 18: carries → Qoh of items 118, 120 and 258 have been correctly mofified

Exercise 7

File: ex7.sql

Trigger: overdraftCheck

Created Table: overdraft

Description: Procedure for creating an update trigger that checks if a customer account exceeds its given credit limit. If the credit limit is exceeded by less than 10% the trigger stores the date, account number and the overdraft amount in a new table (*overdraft*). **If the credit limit is exceeded by more than 10%, an exception is thrown, preventing any update to take place.**

Note: *In our implementation, we took as granted that (1.1 * credit) offers a lower limit to the possible negative values of balance. On the other hand, someone else might think that credit refers to a maximum limit of difference between old and new balance, which would lead to a different solution.*

mysql> SOURCE ex7.sql;

mysql> UPDATE account SET balance = -219 WHERE number = 14356541;

mysql> UPDATE account SET balance = -350 WHERE number = 14356542;

ERROR 1644 (ERROR): overdraftCheck - Credit limit exceeded.

mysql> UPDATE account SET balance = -330 WHERE number = 14356542;

mysql> UPDATE account SET balance = -440 WHERE number = 14356543;

mysql> SELECT * FROM account;

number	customer	balance	credit
11652133	1	0	0
12591815	1	0	0
14096831	1	0	0
14356540	1	0	0
14356541	2	**-219**	200
14356542	3	**-330**	300
14356543	4	**-440**	400

Table 19: account → Balance entries have been modified

mysql> SELECT * FROM overdraft;

number	date	account	amount
1	2015-02-04	14356541	19
2	2015-02-04	14356542	30
3	2015-02-04	14356543	40

Table 20: overdraft → New table has been created