7. WORK WITH TRANSACTIONS

Goal: learn the basics of the transactional mechanism using the MySQL database.

Warning! It is recommended to create the temporary database using the queries shown in the laboratory work 2. Use this temporary database in this laboratory work.

7.1. Create a query that demonstrates usage of transactions to add data into a single table

Consider the sequence of actions when creating and using a query that triggers a transaction, a new entry is added to the table, and then the situation of the incorrect or correct completion of the transaction is simulated. The table status is controlled before the transaction begins, during the execution of the transaction and after it is completed. To do this you need to do the following sequence of actions.

```
SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
   supplier.supplier_address, contract.contract_date
FROM supplied, contract, supplier
WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
AND contract.contract_number = 1;

SET AUTOCOMMIT = 0;
START TRANSACTION;
INSERT INTO supplied VALUES (1, 'Vacuum cleaner', 22, 390);

SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
   supplier.supplier_address, contract.contract_date
FROM supplied, contract, supplier
WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
AND contract.contract_number = 1;

ROLLBACK;

SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
   supplier.supplier_address, contract.contract_date
FROM supplied, contract, supplier
WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
AND contract.contract_number = 1;
```

The SELECT queries can output data that illustrates the state of the table before the transaction begins (Figure 7.1), during the execution of the transaction, and after the transaction is completed.

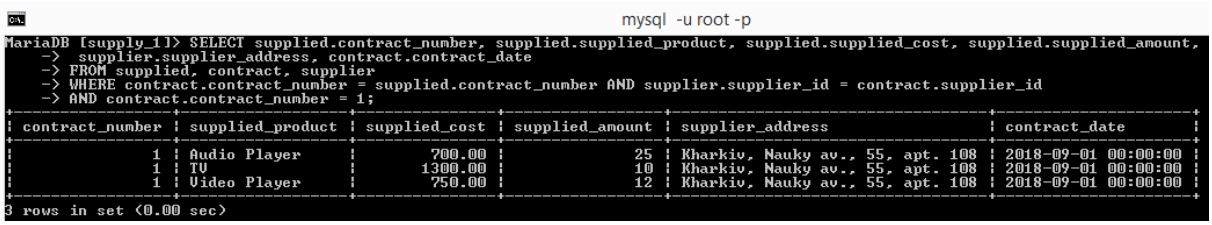

Figure 7.1

As can be seen from the data shown, a new entry in the table appears (Figure 7.2).

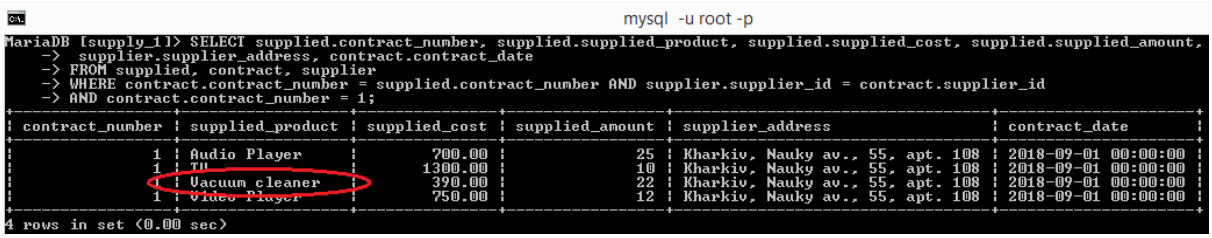

Figure 7.2

And then the new entry disappears (Figure 7.3)

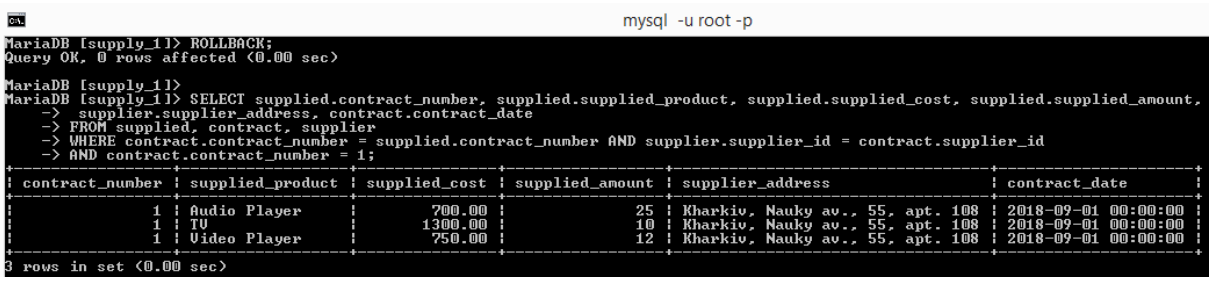

Figure 7.3

Now it is necessary to consider the situation of the correct completion of the transaction. To do this, in the text of the query, you need to change the ROLLBACK statement to COMMIT. Perform the statement and analyze the results.

7.2. Create a query that demonstrates usage of transactions to add data into multiple tables

Consider the sequence of actions when creating and using a query that triggers a transaction, and then creates a new supplier, a contract for the supply is concluded with that supplier, the products delivered under this contract. The situation of the incorrect or correct completion of the transaction is simulated. The status of the tables is controlled before the transaction begins, in the process of executing the transaction and after the transaction is completed. To do this you need to do the following sequence of actions.

```
SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;

SET AUTOCOMMIT = 0;
START TRANSACTION;
INSERT INTO supplier (supplier_id, supplier_address, supplier_phone)
   VALUES (6, 'Kyiv, Velyka Vasylkivska st., 55', '');
INSERT INTO contract (contract_date, supplier_id, contract_note)
   VALUES ('2018-12-12', 6, '');
INSERT INTO supplied VALUES (6, 'Vacuum cleaner', 22, 390);
INSERT INTO supplied VALUES (6, 'Coffee machine', 33, 90);

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;

ROLLBACK;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;
```

The SELECT queries allow you to output data that illustrates the status of the tables before the transaction begins, in the process of executing the transaction and after the transaction is completed. As can be seen from the data shown, new entries in the tables appear and then disappear.

Now it is necessary to consider the situation of the correct completion of the transaction. To do this, change the ROLLBACK statement to COMMIT. Perform the query and analyze the results.

7.3. Create a query that demonstrates usage of transactions to update data in multiple tables

Consider the sequence of actions when creating and using the query that triggers the transaction, then the data entered in the table are changed when the previous request is executed. The situation of the incorrect or correct completion of the transaction is simulated. The status of the tables is controlled before the transaction begins, in the process of executing the transaction and after the transaction is completed. To do this you need to do the following sequence of actions.

```
ALTER TABLE contract
DROP FOREIGN KEY contract_ibfk_1;

ALTER TABLE contract
ADD CONSTRAINT contract_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE CASCADE ON UPDATE CASCADE;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;

SET AUTOCOMMIT = 0;
START TRANSACTION;
UPDATE supplier SET supplier_id = 22 WHERE supplier_id = 6;
UPDATE supplied SET supplied_cost = supplied_cost * 1.1 WHERE contract_number = 8;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied WHERE contract_number = 8;

ROLLBACK;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied WHERE contract_number = 8;
```

The SELECT queries allow you to output data that illustrates the status of the tables before the transaction begins, in the process of executing the

transaction and after the transaction is completed. As can be seen from the data shown, new entries in the tables appear and then disappear.

Now it is necessary to consider the situation of the correct completion of the transaction. To do this, change the ROLLBACK statement to COMMIT. Perform the query and analyze the results.

7.4. Create a query that demonstrates usage of transactions to delete data from multiple tables

Consider the sequence of actions when creating and using a query that triggers a transaction that removes the supplier that was created when query 2 was executed and whose data was modified by query 3.

Considering the CASCADE referential integrity control mechanism the data will be deleted in several tables. The situation of the incorrect or correct completion of the transaction is simulated. The status of the tables is controlled before the transaction begins, in the process of executing the transaction and after the transaction is completed. To do this you need to do the following sequence of actions.

```
ALTER TABLE supplied
DROP FOREIGN KEY supplied_ibfk_1;

ALTER TABLE supplied
ADD CONSTRAINT supplied_ibfk_1 FOREIGN KEY (contract_number) REFERENCES contract(contract_number) ON DELETE CASCADE ON UPDATE CASCADE;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;

SET AUTOCOMMIT = 0;
START TRANSACTION;
DELETE FROM supplier WHERE supplier_id = 22;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;

ROLLBACK;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;
```

The SELECT queries allow you to output data that illustrates the status of the tables before the transaction begins, in the process of executing the

transaction and after the transaction is completed. As can be seen from the data shown, new entries in the tables appear and then disappear.

Now it is necessary to consider the situation of the correct completion of the transaction. To do this, change the ROLLBACK statement to COMMIT. Perform the query and analyze the results.

7.5. Questions

1. What is a transaction?
2. Which table types support transactions in the MySQL DBMS?
3. Which table types do not support transactions in the MySQL DBMS?
4. How to disable transaction auto commit in the MySQL DBMS?
5. What operator is used to complete a transaction?
6. What operator is used to cancel changes performed by a transaction?
7. Which command of the MySQL DBMS should be used to enable the transaction auto commit mode for a certain sequence of statements?
8. With which type of tables the SAVEPOINT and ROLLBACK TO SAVEPOINT operators might be used?
9. What is the purpose of the SAVEPOINT and ROLLBACK TO SAVEPOINT operators?
10. Which issues might be caused by parallel execution of transactions?
11. What are the levels of transaction isolation and what problems can each of these levels solve?
12. What table type is used in the MySQL database by default (starting from the version 5.5)?
13. Which transaction isolation levels are supported by InnoDB?
14. Which transaction isolation level is set by default in the InnoDB engine?