**Лабораторная работа № 11. Особенности использования транзакций**

1. Разработать сценарий, демонстрирующий работу в режиме *неявной* транзакции. Проанализировать пример, приведенный справа, в котором создается таблица Х, и создать сценарий для другой таблицы.

set nocount on

if exists (select \* from SYS.OBJECTS where OBJECT\_ID= object\_id(N'DBO.MyTable'))

drop table MyTable;

DECLARE @rows int;

SET IMPLICIT\_TRANSACTIONS ON;

CREATE table MyTable(

X int UNIQUE

);

INSERT INTO MyTable VALUES

(1),(2),(3),(4);

SET @rows = (SELECT COUNT(\*) FROM MyTable);

print 'Количество строк в таблице MyTable: ' + cast(@rows as varchar(5));

if @rows < 5

commit;

else

rollback;

SET IMPLICIT\_TRANSACTIONS OFF;

if exists (select \* from SYS.OBJECTS where OBJECT\_ID = object\_id(N'DBO.MyTable'))

print 'таблица MyTable есть';

else

print 'таблицы MyTable нет';



1. Разработать сценарий, демонстрирующий свойство *атомарности* *явной* транзакции на примере базы данных X\_UNIVER. В блоке CATCH предусмотреть выдачу соответствующих сообщений об ошибках. Опробовать работу сценария при использовании различных операторов модификации таблиц.

use UNIVER;

BEGIN try

BEGIN tran

DELETE SUBJECT\_T WHERE SUBJECT\_T LIKE 'test\_subj%';

INSERT INTO SUBJECT\_T VALUES('test\_subj5', 'test\_subj5', 'ИСиТ');

UPDATE dbo.SUBJECT\_T

SET SUBJECT\_T.SUBJECT\_NAME = SUBJECT\_T.SUBJECT\_NAME + '(upd)'

FROM dbo.SUBJECT\_T

WHERE SUBJECT\_T.SUBJECT\_NAME LIKE 'test\_subj%';

SELECT \*

FROM dbo.SUBJECT\_T;

RAISERROR('Test error', 1, 1);

commit tran;

END try

BEGIN catch

print 'Ошибка: ' + cast(error\_number() as varchar(5))

print 'Line: ' + cast(error\_line() as varchar(5))

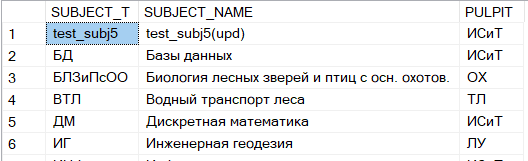
print 'Message: ' + error\_message();

if @@TRANCOUNT > 0

rollback tran;

END catch;





1. Разработать сценарий, демонстрирующий применение оператора SAVE TRAN на примере базы данных X\_UNIVER. В блоке CATCH предусмотреть выдачу соответствующих сообщений об ошибках. Опробовать работу сценария при использовании различных контрольных точек и различных операторов модификации таблиц.

DECLARE @checkpoint varchar(10);

BEGIN try

BEGIN transaction

DELETE SUBJECT\_T WHERE SUBJECT\_T LIKE 'test\_subj%';

SET @checkpoint = 'checkpoint1'; SAVE transaction @checkpoint;

INSERT INTO SUBJECT\_T VALUES('test\_subj5', 'test\_subj5', 'ИСиТ');

SET @checkpoint = 'checkpoint2'; SAVE transaction @checkpoint;

UPDATE dbo.SUBJECT\_T

SET SUBJECT\_T.SUBJECT\_NAME = SUBJECT\_T.SUBJECT\_NAME + '(upd)'

FROM dbo.SUBJECT\_T

WHERE SUBJECT\_T.SUBJECT\_NAME LIKE 'test\_subj%';

SET @checkpoint = 'checkpoint3'; SAVE transaction @checkpoint;

SELECT \*

FROM dbo.SUBJECT\_T;

commit transaction;

END try

BEGIN catch

print 'Ошибка: ' + cast(error\_number() as varchar(5))

print 'Line: ' + cast(error\_line() as varchar(5))

print 'Message: ' + error\_message();

if @@TRANCOUNT > 0

BEGIN

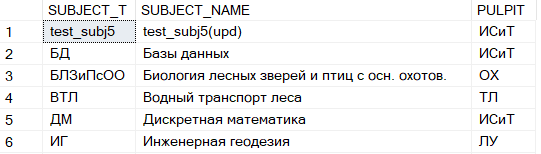
print 'Контрольная точка: ' + @checkpoint;

rollback transaction @checkpoint;

commit transaction;

END;

END catch;



1. Разработать два сценария A и B на примере базы данных X\_UNIVER. Сценарий A представляет собой явную транзакцию с уровнем изолированности READ UNCOMMITED, сценарий B – явную транзакцию с уровнем изолированности READ COMMITED (по умолчанию). Сценарий A должен демонстрировать, что уровень READ UNCOMMITED допускает неподтвержденное, неповторяющееся и фантомное чтение.

--- A ---

SET transaction isolation level READ UNCOMMITTED;

BEGIN transaction

SELECT @@SPID 'SPID', 'insert SUBJECT' 'результат', \*

FROM dbo.SUBJECT\_T

WHERE SUBJECT\_T.PULPIT = 'ИСиТ';

--- t1 ---

SELECT

@@SPID 'SPID',

'update AUDITORIUM' 'результат',

AUDITORIUM\_NAME,

AUDITORIUM\_TYPE,

AUDITORIUM\_CAPACITY

FROM dbo.AUDITORIUM

WHERE AUDITORIUM.AUDITORIUM\_TYPE LIKE 'ЛК%';

commit;

--- t2 ---

--- B ---

set transaction isolation level READ COMMITTED

begin transaction

SELECT @@SPID 'SPID';

INSERT into dbo.SUBJECT\_T values('test\_subj6', 'test\_subj\_tran', 'ИСиТ');

UPDATE AUDITORIUM SET AUDITORIUM.AUDITORIUM\_CAPACITY = 22

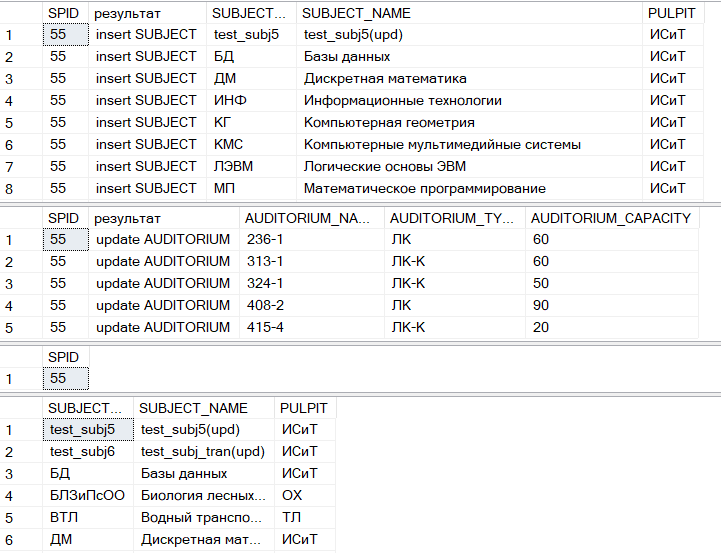
FROM dbo.AUDITORIUM

WHERE AUDITORIUM.AUDITORIUM = '301-1';

--- t1 ---

--- t2 ---

rollback;



1. Разработать два сценария A и B на примере базы данных X\_UNIVER. Сценарии A и В представляют собой явные транзакции с уровнем изолированности READ COMMITED. Сценарий A должен демонстрировать, что уровень READ COMMITED не допускает неподтвержденного чтения, но при этом возможно неповторяющееся и фантомное чтение.

--- A ---

set transaction isolation level READ COMMITTED

begin transaction

select count(\*) [count] from AUDITORIUM where AUDITORIUM\_CAPACITY >= 30;

--- t1 ---

--- t2 ---

select

@@SPID 'SID',

'update AUDITORIUM' 'результат',

AUDITORIUM\_NAME,

AUDITORIUM\_TYPE,AUDITORIUM\_CAPACITY

from AUDITORIUM

where AUDITORIUM\_NAME = '301-1';

commit;

--- B ---

begin transaction

--- t1 ---

update AUDITORIUM set AUDITORIUM\_CAPACITY = 30 where AUDITORIUM\_NAME = '301-1';

commit;

--- t2 ---



1. Разработать два сценария A и B на примере базы данных X\_UNIVER. Сценарий A представляет собой явную транзакцию с уровнем изолированности REPEATABLE READ. Сценарий B – явную транзакцию с уровнем изолированности READ COMMITED. Сценарий A должен демонстрировать, что уровень REAPETABLE READ не допускает неподтвержденного чтения и неповторяющегося чтения, но при этом возможно фантомное чтение.

--- A ---

set transaction isolation level REPEATABLE READ

begin transaction

select SUBJECT\_NAME from SUBJECT\_T where SUBJECT\_T = 'test\_subj6';

--- t1 ---

--- t2 ---

select case

when SUBJECT\_T = 'test\_subj6' then 'insert SUBJECT' else '-'

end 'результат',

SUBJECT\_NAME

FROM SUBJECT\_T

WHERE PULPIT = 'ИСиТ';

commit;

--- B ---

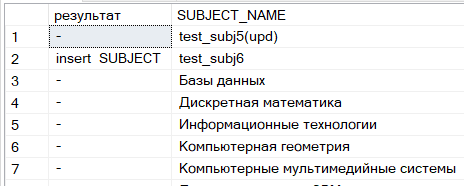
begin transaction

--- t1 ---

INSERT into dbo.SUBJECT\_T values('test\_subj6', 'test\_subj6', 'ИСиТ');

commit;

--- t2 ---



1. Разработать два сценария A и B на примере базы данных X\_UNIVER. Сценарий A представляет собой явную транзакцию с уровнем изолированности SERIALIZABLE. Сценарий B – явную транзакцию с уровнем изолированности READ COMMITED. Сценарий A должен демонстрировать отсутствие фантомного, неподтвержденного и неповторяющегося чтения.

set transaction isolation level SERIALIZABLE;

--- A ---

BEGIN transaction

DELETE SUBJECT\_T WHERE SUBJECT\_T LIKE 'test\_subj%';

INSERT INTO SUBJECT\_T VALUES('test\_subj7', 'test\_subj7', 'ИСиТ');

UPDATE dbo.SUBJECT\_T

SET SUBJECT\_T.SUBJECT\_NAME = SUBJECT\_T.SUBJECT\_NAME + '(upd)'

FROM dbo.SUBJECT\_T

WHERE SUBJECT\_T.SUBJECT\_NAME LIKE 'test\_subj%';

--- t1 ---

SELECT SUBJECT\_T from SUBJECT\_T where SUBJECT\_T LIKE 'test\_subj%';

--- t2 ---

commit;

--- B ---

BEGIN transaction

delete SUBJECT\_T where SUBJECT\_T LIKE 'test\_subj%';

INSERT INTO SUBJECT\_T VALUES('test\_subj7', 'test\_subj7', 'ИСиТ');

UPDATE dbo.SUBJECT\_T

SET SUBJECT\_T.SUBJECT\_NAME = SUBJECT\_T.SUBJECT\_NAME + '(upd)'

FROM dbo.SUBJECT\_T

WHERE SUBJECT\_T.SUBJECT\_NAME LIKE 'test\_subj%';

SELECT SUBJECT\_T from SUBJECT\_T where SUBJECT\_T LIKE 'test\_subj%';

--- t1 ---

commit;

SELECT SUBJECT\_T from SUBJECT\_T where SUBJECT\_T LIKE 'test\_subj%';

--- t2 ---



1. Разработать сценарий, демонстрирующий свойства *вложенных* транзакций, на примере базы данных X\_UNIVER.

use UNIVER;

SET nocount on;

BEGIN transaction

INSERT INTO SUBJECT\_T VALUES('test\_subj8', 'test\_subj8', 'ИСиТ');

BEGIN transaction

UPDATE dbo.SUBJECT\_T

SET SUBJECT\_T.SUBJECT\_NAME = SUBJECT\_T.SUBJECT\_NAME + '(upd)'

FROM dbo.SUBJECT\_T

WHERE SUBJECT\_T.SUBJECT\_NAME LIKE 'test\_subj%';

commit;

if @@TRANCOUNT > 0

rollback;

SELECT \*

FROM SUBJECT\_T

WHERE SUBJECT\_T.SUBJECT\_NAME LIKE 'test\_subj%';



9. Разработать скрипты с использованием транзакций для базы данных **X\_MyBASE**.

use Dobriyan\_MyBase;

-- #1

DECLARE @rows int;

SET IMPLICIT\_TRANSACTIONS ON;

SET nocount on;

INSERT INTO Specialities VALUES(5, 'test1'),(6, 'test2'),(7, 'test3'),(8, 'test4');

SET @rows = (SELECT COUNT(\*) FROM Specialities);

print 'Количество строк в таблице Specialities: ' + cast(@rows as varchar(5));

if @rows < 5

commit;

else

rollback;

SET IMPLICIT\_TRANSACTIONS OFF;

DELETE Specialities WHERE Specialities.Speciality LIKE 'test%';

if exists (select \* from SYS.OBJECTS where OBJECT\_ID = object\_id(N'DBO.Specialities'))

print 'таблица Specialities есть';

else

print 'таблицы Specialities нет';

-- #2

use Dobriyan\_MyBase;

SET nocount on;

BEGIN try

BEGIN transaction

INSERT INTO Specialities VALUES(5, 'test1'),(6, 'test2'),(7, 'test3'),(8, 'test4');

DELETE Specialities WHERE Specialities.Speciality LIKE 'test%';

RAISERROR('Test error', 1, 1);

commit transaction;

END try

BEGIN catch

print 'Ошибка: ' + cast(error\_number() as varchar(5))

print 'Line: ' + cast(error\_line() as varchar(5))

print 'Message: ' + error\_message();

if @@TRANCOUNT > 0

rollback transaction;

END catch;

-- #3

use Dobriyan\_MyBase;

set nocount on;

DECLARE @checkpoint varchar(10);

BEGIN try

BEGIN transaction

INSERT INTO Specialities VALUES(5, 'test1'),(6, 'test2'),(7, 'test3'),(8, 'test4');

SET @checkpoint = 'checkpoint1'; SAVE transaction @checkpoint;

DELETE Specialities WHERE Specialities.Speciality LIKE 'test%';

SET @checkpoint = 'checkpoint2'; SAVE transaction @checkpoint;

SELECT \*

FROM dbo.Specialities;

commit transaction;

END try

BEGIN catch

print 'Ошибка: ' + cast(error\_number() as varchar(5))

print 'Line: ' + cast(error\_line() as varchar(5))

print 'Message: ' + error\_message();

if @@TRANCOUNT > 0

BEGIN

print 'Контрольная точка: ' + @checkpoint;

rollback transaction @checkpoint;

commit transaction;

END;

END catch;

-- #4

use Dobriyan\_MyBase;

SET nocount on;

--- A ---

SET transaction isolation level READ UNCOMMITTED;

BEGIN transaction

SELECT @@SPID 'SPID', 'insert Speciality' 'результат', \*

FROM dbo.Specialities;

--- t1 ---

SELECT

@@SPID 'SPID',

'update AUDITORIUM' 'результат',

id, Speciality

FROM dbo.Specialities;

commit;

--- t2 ---

--- B ---

begin transaction

SELECT @@SPID 'SPID';

INSERT INTO Specialities VALUES(5, 'test1'),(6, 'test2'),(7, 'test3'),(8, 'test4');

UPDATE Specialities SET Specialities.id = 22

FROM dbo.Specialities

WHERE Specialities.Speciality = 'test2';

--- t1 ---

rollback;

--- t2 ---

-- #5

use Dobriyan\_MyBase;

SET nocount on;

--- A ---

set transaction isolation level READ COMMITTED

begin transaction

select count(\*) [count] from Courses where Courses.Peoples\_Count >= 20;

--- t1 ---

--- t2 ---

select

@@SPID 'SID',

'update Courses' 'результат', \*

from Courses

where Courses.Course\_Subject = 'ОИТ';

commit;

--- B ---

begin transaction

--- t1 ---

update Courses set Courses.Peoples\_Count = 20 where Courses.Course\_Subject = 'ОИТ';

commit;

--- t2 ---

-- #6

use Dobriyan\_MyBase;

SET nocount on;

--delete dbo.Groups where Students\_Count = 33;

--- A ---

set transaction isolation level REPEATABLE READ

begin transaction

select \* from Groups where Groups.Speciality = 2;

--- t1 ---

--- t2 ---

select case

when Groups.Students\_Count = 33 then 'insert Group' else '-'

end 'результат', \*

FROM Groups

WHERE Groups.Speciality = 2;

commit;

--- B ---

begin transaction

--- t1 ---

INSERT into dbo.Groups values(10, 33, 2, 'дневная'),(11, 33, 2, 'дневная');

commit;

--- t2 ---

-- #7

use Dobriyan\_MyBase;

SET nocount on;

set transaction isolation level SERIALIZABLE;

--- A ---

BEGIN transaction

DELETE Specialities WHERE Specialities.Speciality LIKE 'test%';

INSERT INTO Specialities VALUES(5, 'test1'),(6, 'test2'),(7, 'test3'),(8, 'test4');

UPDATE Specialities SET Specialities.id = 22

FROM dbo.Specialities

WHERE Specialities.Speciality = 'test2';

--- t1 ---

SELECT \* from Specialities where Speciality LIKE 'test%';

--- t2 ---

commit;

--- B ---

BEGIN transaction

DELETE Specialities WHERE Specialities.Speciality LIKE 'test%';

INSERT INTO Specialities VALUES(5, 'test1'),(6, 'test2'),(7, 'test3'),(8, 'test4');

UPDATE Specialities SET Specialities.id = 22

FROM dbo.Specialities

WHERE Specialities.Speciality = 'test2';

SELECT \* from Specialities where Speciality LIKE 'test%';

--- t1 ---

commit;

SELECT \* from Specialities where Speciality LIKE 'test%';

--- t2 ---

-- #8

use Dobriyan\_MyBase;

SET nocount on;

BEGIN transaction

INSERT INTO Specialities VALUES(5, 'test1'),(6, 'test2'),(7, 'test3'),(8, 'test4');

BEGIN transaction

UPDATE Specialities SET Specialities.id = 22

FROM dbo.Specialities

WHERE Specialities.Speciality = 'test2';

commit;

if @@TRANCOUNT > 0

rollback;

SELECT \* from Specialities where Speciality LIKE 'test%';

DELETE Specialities WHERE Specialities.Speciality LIKE 'test%';