

Практична робота 5

Тема: «Транзакція. Журнал транзакцій»

Атомарність

Створюється програмний код, який на початку робить перевірку наявності бази даних (у прикладі це БД *sqlmagic*), якщо вона існує, то її видаляємо.

```
USE master;  
IF DB_ID ('sqlmagic') IS NOT NULL  
BEGIN  
ALTER DATABASE sqlmagic  
SET RESTRICTED_USER WITH ROLLBACK IMMEDIATE  
DROP DATABASE sqlmagic  
END;
```

створюється база даних

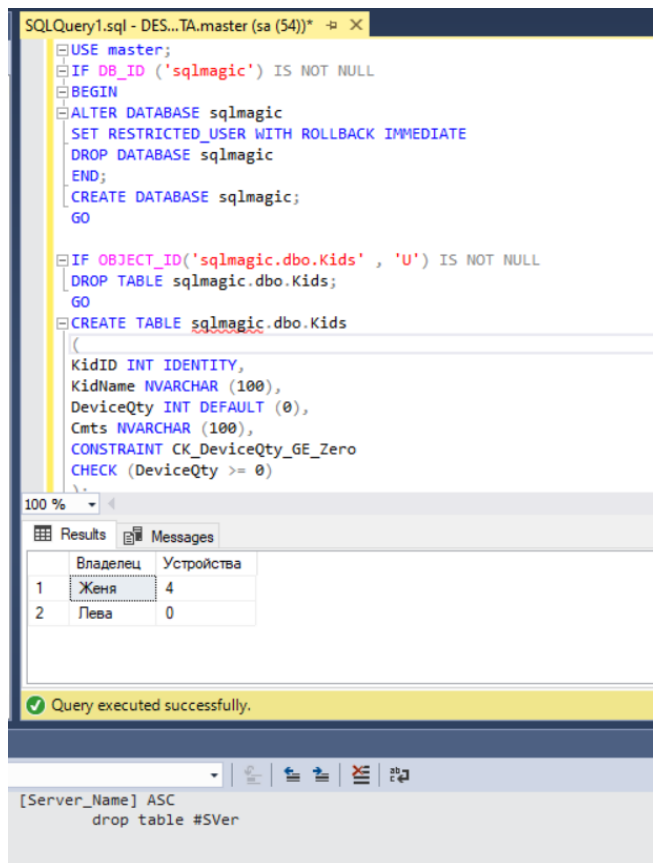
```
CREATE DATABASE sqlmagic;  
GO
```

Створюється таблиця Kids, в якій є унікальний номер дитини, ім'я дитини, поле в якому вказується скільки пристроїв у дитини, та поле для коментаря, яке не використовується в даному прикладі та стоїть обмеження що кількість пристроїв не може бути менше 0 та від'ємне

```
IF OBJECT_ID('sqlmagic.dbo.Kids' , 'U') IS NOT NULL  
DROP TABLE sqlmagic.dbo.Kids;  
GO  
CREATE TABLE sqlmagic.dbo.Kids  
(  
KidID INT IDENTITY,  
KidName NVARCHAR (100),  
DeviceQty INT DEFAULT (0),  
Cmts NVARCHAR (100),  
CONSTRAINT CK_DeviceQty_GE_Zero  
CHECK (DeviceQty >= 0)  
);  
GO
```

Додаються дві дитини з іменами Женя та Льова та кількість пристроїв у них, перевіряється за допомогою інструкції select

```
INSERT sqlmagic.dbo.Kids (KidName, DeviceQty)  
VALUES  
(N'Женя' , 4),  
(N'Лева' , 0);  
  
SELECT  
KidName AS "Владелец",  
DeviceQty AS "Устройства"  
FROM sqlmagic.dbo.Kids ORDER BY KidName;
```



Після того, як ми запускаємо код, ми бачимо що дійсно у Жені 4 пристрої, а у Льови 0.

Далі переходимо до наступного скрипту.

Спочатку перевіряємо стан до початку транзакції. Бачимо що до транзакції у Жені 4 пристрої, а у Льови 0. Також бачимо номер процесу 52.

```

SET NOCOUNT ON;
--Состояние до начала транзакции
SELECT
'До транзакции' AS "Состояние",
KidName AS "Владелец",
DeviceQty AS "Устройства",
@@SPID AS "Процесс",
@@TRANCOUNT AS "Количество транзакций"
FROM sqlmagic.dbo.Kids ORDER BY KidName;
--Начало транзакции
BEGIN TRANSACTION;
SELECT
  @@TRANCOUNT AS "Количество транзакций"; --починаємо транзакцію

UPDATE sqlmagic.dbo.Kids
SET DeviceQty -= 2
WHERE KidName = N'Женя'; --відбувається зменшення пристроїв у Жені, кількість, яка
була, зменшується на 2

SELECT
'Транзакция' AS "Состояние",
KidName AS "Владелец",
DeviceQty AS "Устройства",
@@SPID AS "Процесс",

```

```
@@TRANCOUNT AS "Количество транзакций"
FROM sqlmagic.dbo.Kids ORDER BY KidName; --ми можемо переконатись в тому, що
зменшення відбулось, у Жені стало 2 пристрої, а у Льови 0, кількість транзакції
1
```

```
UPDATE sqlmagic.dbo.Kids
SET DeviceQty += 2
WHERE KidName = N'Лева'; --відбувається дія по збільшенню кількості пристроїв у
Льови
```

```
COMMIT TRANSACTION;
```

```
--Состояние после транзакции
SELECT
'После транзакции' AS "Состояние",
KidName AS "Владелец",
DeviceQty AS "Устройства",
@@SPID AS "Процесс",
@@TRANCOUNT AS "Количество транзакций"
FROM sqlmagic.dbo.Kids ORDER BY KidName;
```

Після виконання усього коду, ми бачимо що до початку транзакції у Жені 4 пристрої, у Льови – 0, та номер процесу 52, кількість транзакцій – 0, далі бачимо що кількість транзакцій змінилось та стало 1, а пристроїв у Жені стало 2, у Льови кількість пристроїв не змінилась – 0, після закінчення транзакції все завершується тим, що у Жені кількість пристроїв – 2 та у Льови теж 2, але кількість транзакцій 0. Все виконалось повністю та ми бачимо результат даної роботи.

The screenshot shows a SQL Server Enterprise Manager window with a query execution window open. The query is as follows:

```
SET NOCOUNT ON;
--Состояние до начала транзакции
SELECT
'До транзакции' AS "Состояние",
KidName AS "Владелец",
DeviceQty AS "Устройства",
@@SPID AS "Процесс",
@@TRANCOUNT AS "Количество транзакций"
FROM sqlmagic.dbo.Kids ORDER BY KidName;
--Начало транзакции
```

The results window shows the following data:

Состояние	Владелец	Устройства	Процесс	Количество транзакций
До транзакции	Женя	4	52	0
До транзакции	Лева	0	52	0

The messages window shows the following message:

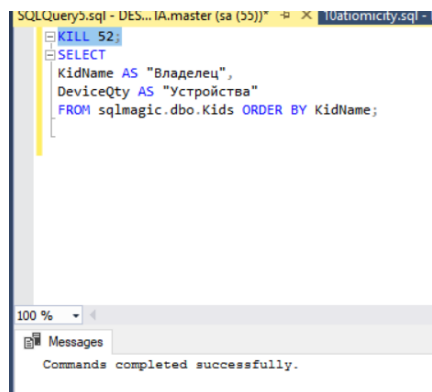
```
1 1
```

The status bar at the bottom indicates "Query executed successfully."

```
KILL 52;
SELECT
KidName AS "Владелец",
```

```
DeviceQty AS "Устройства"  
FROM sqlmagic.dbo.Kids ORDER BY KidName;
```

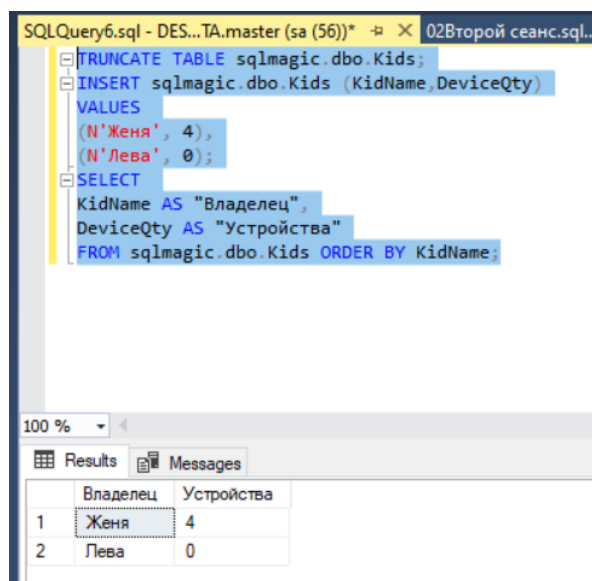
Ми примусово прибираємо процес під номер 52, за допомогою інструкції Kill, воно виконується, та бачимо що знову у Жені 4 пристрої, а у Льови – 0, тобто коли відбулась примусова зупинка процесу, в якому відбувається транзакція, таке могло статися в результаті якоїсь помилки, все повернулось на свої місця, наче нічого і не було. Атомарність працює, тобто робота єдина, якщо робота не дійшла до кінця, то все повертається в початковий стан.



Погодження даних

```
TRUNCATE TABLE sqlmagic.dbo.Kids;  
INSERT sqlmagic.dbo.Kids (KidName, DeviceQty)  
VALUES  
(N'Женя', 4),  
(N'Лева', 0);  
SELECT  
KidName AS "Владелец",  
DeviceQty AS "Устройства"  
FROM sqlmagic.dbo.Kids ORDER BY KidName;
```

За допомогою даного коду ми повертаємо все в початковий вигляд, тобто у Жені 4 пристрої, а у Льови – 0.



```

SET NOCOUNT ON;
--Состояние до начала транзакции
SELECT
KidName AS "Владелец",
DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids ORDER BY KidName;

```

SQLQuery7.sql - DES...TA.master (sa (57))* -> X 10atiomicity.

```

--Начало транзакции
UPDATE sqlmagic.dbo.Kids
SET DeviceQty =
CASE KidName
WHEN N'Женя' THEN DeviceQty -5
WHEN N'Лева' THEN DeviceQty +5
END
WHERE KidName IN (N'Женя', N'Лева');

```

100 %

Results Messages

	Владелец	Устройства
1	Женя	4
2	Лева	0

До початку транзакції ми бачимо що у Жені 4 пристрої, а у Льови – 0.

```

--Начало транзакции
UPDATE sqlmagic.dbo.Kids
SET DeviceQty =
CASE KidName
WHEN N'Женя' THEN DeviceQty -5
WHEN N'Лева' THEN DeviceQty +5
END
WHERE KidName IN (N'Женя', N'Лева');

```

Виконується дія пов'язана зі спробою відібрати у Жені 5 пристроїв, та додати їх Льові також стоїть фільтр, який за допомогою where відбирає тільки Женю та Льову. За допомогою кейсу CASE, якщо ім'я дитини Женя, то забрати в нього 5 пристроїв, а якщо Льова то додати йому 5 пристроїв.

SQLQuery7.sql - DES...TA.master (sa (57))* X 10atiomicity.sql - D...ATA.master (sa (52)) 01Revert.sql - DESK...ATA.master (sa (56)) 00sqlmagic.sql - DE...TA.master (sa (54))

```

--Начало транзакции
UPDATE sqlmagic.dbo.Kids
SET DeviceQty =
CASE KidName
WHEN N'Женя' THEN DeviceQty -5
WHEN N'Лева' THEN DeviceQty +5
END
WHERE KidName IN (N'Женя', N'Лева');
--Состояние после транзакции
SELECT
KidName AS "Владелец",
DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids ORDER BY KidName;

```

100 %

Messages

Msg 547, Level 16, State 0, Line 9
The UPDATE statement conflicted with the CHECK constraint "CK_DeviceQty_GE_Zero". The conflict occurred in database "sqlmagic", table "dbo.Kids", column 'DeviceQty'.
The statement has been terminated.

Відбувається помилка перевірки, тому що кількість пристроїв повинна бути більше або дорівнювати 0, а в даному випадку, якщо в нас всього 4 пристрої, а ми заберемо 5, то буде від'ємне число.

```

--Состояние после транзакции
SELECT
KidName AS "Владелец",
DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids ORDER BY KidName;

```

Отже, після перевірки транзакції, ми бачимо що у Жені 4 пристрої, а у Льови – 0.

```
SQLQuery7.sql - DES...TA.master (sa (57))* X 10atomicity.sql - [
SET DeviceQty =
CASE KidName
WHEN N'Женя' THEN DeviceQty -5
WHEN N'Лева' THEN DeviceQty +5
END
WHERE KidName IN (N'Женя', N'Лева');

--Состояние после транзакции
SELECT
KidName AS "Владелец",
DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids ORDER BY KidName;
```

	Владелец	Устройства
1	Женя	4
2	Лева	0

Відсутність ізолюваності транзакції

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET NOCOUNT ON;
-- Состояние до начала транзакции
SELECT
KidName AS "Владелец",
DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids
ORDER BY KidName;
```

Перевіряємо стан до початку транзакції, яка кількість пристроїв у хлопців, та бачимо що до початку транзакції у Жені 4 пристрої, а у Льови 0 пристроїв.

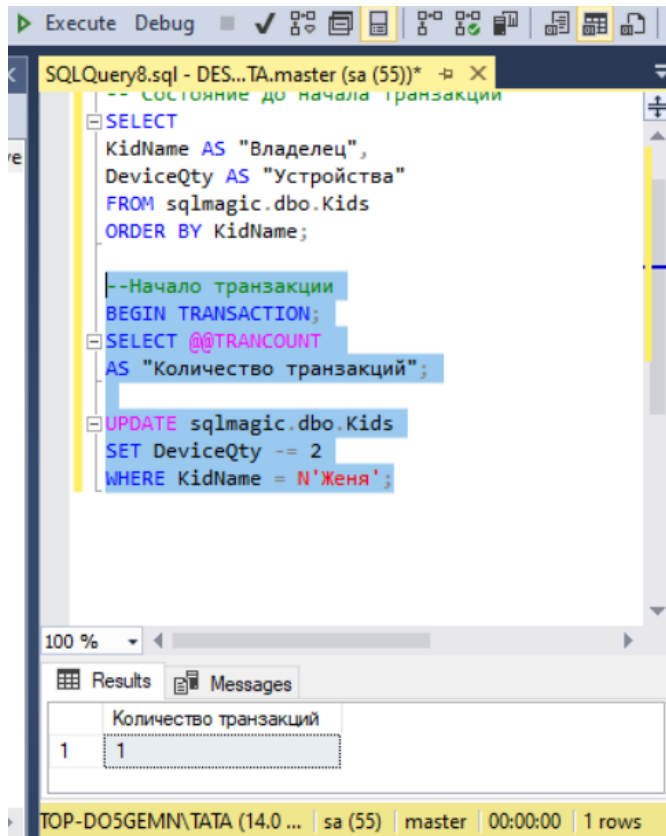
```
SQLQuery8.sql - DES...TA.master (sa (55))* X 21Consistency.sql - [
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET NOCOUNT ON;
-- Состояние до начала транзакции
SELECT
KidName AS "Владелец",
DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids
ORDER BY KidName;
```

	Владелец	Устройства
1	Женя	4
2	Лева	0

```
--Начало транзакции
BEGIN TRANSACTION;
SELECT @@TRANSCOUNT
AS "Количество транзакций";

UPDATE sqlmagic.dbo.Kids
SET DeviceQty -= 2
WHERE KidName = N'Женя';
```

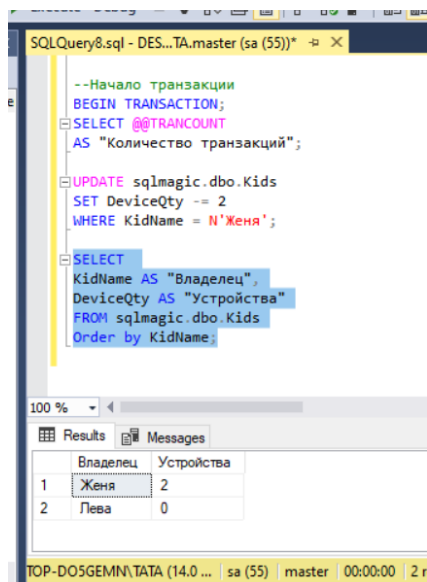
Починаємо транзакцію та рахуємо кількість транзакцій, бачимо що вона 1.



```

SELECT
  KidName AS "Владелец",
  DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids
Order by KidName;

```



Далі дивимось в якому стані кількість пристроїв у хлопців в середині транзакції. Бачимо що на даному етапі у Жені 2 пристрої, а у Льови – 0.

```

SET NOCOUNT ON;
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
--Читаем неподтвержденные изменения

```

```

SELECT
  KidName AS "Владелец",
  DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids
ORDER BY KidName;

```

```
33Pessimistic.sql -...TATA.master (sa (58))* -p X
SET NOCOUNT ON;
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITED
--Читаем неподтвержденные изменения

SELECT
    KidName AS "Владелец",
    DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids
ORDER BY KidName;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED
-- Читаем только подтвержденные изменения
```

	Владелец	Устройства
1	Женя	2
2	Лева	0

В даному прикладі ми обираємо перший рівень ізольованості транзакцій. Ми допускаємо читання непідтверджених змін. Транзакція ще не закінчилась, але ми вже бачимо неузгодженість. Дані не коректні.

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
-- Читаем только подтвержденные изменения
```

```
SELECT
    KidName AS "Владелец",
    DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids
ORDER BY KidName
```

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
-- Читаем только подтвержденные изменения

SELECT
    KidName AS "Владелец",
    DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids
ORDER BY KidName
```

	Владелец	Устройства
1	Женя	2
2	Лева	2

Якщо нас не влаштовує читання непідтверджених даних, то ми встановлюємо рівень ізольованості та читаємо тільки підтверджені зміни, але нам доведеться заплатити за це часом, вона буде виконуватись до тих пір, поки не буде завершена та транзакція, яка блокування встановила.

```
SELECT
    KidName AS "Владелец",
    DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids
Order by KidName;
```

```
UPDATE sqlmagic.dbo.Kids
SET DeviceQty += 2
where KidName = N'Лева'
COMMIT TRANSACTION
```



```

31Pessimistic.sql - ...TATA.master (sa (55))
AS "Количество транзакций";
UPDATE sqlmagic.dbo.Kids
SET DeviceQty += 2
WHERE KidName = N'Женя';
SELECT
  KidName AS "Владелец",
  DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids
ORDER BY KidName;
UPDATE sqlmagic.dbo.Kids
SET DeviceQty += 2
WHERE KidName = N'Лева'

```

	Владелец	Устройства
1	Женя	2
2	Лева	0

```

--Состояние после транзакции
SELECT
  KidName AS "Владелец",
  DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids
ORDER BY KidName;

```

```

31Pessimistic.sql - ...TA.sqlmagic (sa (57))
01Revert.sql - DESK...A.sqlm
ORDER BY KidName;
UPDATE sqlmagic.dbo.Kids
SET DeviceQty += 2
WHERE KidName = N'Лева'
COMMIT TRANSACTION
--Состояние после транзакции
SELECT
  KidName AS "Владелец",
  DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids
ORDER BY KidName;

```

	Владелец	Устройства
1	Женя	2
2	Лева	2

Після того, як транзакція завершилась, ми вже можемо побачити що блокування до даних знімається та одразу доступ до даних надається, транзакція бачить правильне, відповідаючи бізнес логіці наших змін, дані, але за це доводиться платити очікуванням а отже зниженням продуктивності.

Надійність

```

SET NOCOUNT ON;
--Состояние до начала транзакции
SELECT
  'До начала транзакции' AS "Состояние",
  KidName AS "Владелец",
  DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids ORDER BY KidName;

```

Дивимось стан до початку транзакції у Жені 4 пристрої, у Льови – 0.

```

44Durability.sql - D:\A.sqlmagic (sa (59)) - 31Pessimistic.sql -
SET NOCOUNT ON;
--Состояние до начала транзакции
SELECT
'До начала транзакции' AS "Состояние",
KidName AS "Владелец",
DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids ORDER BY KidName;
--Начало транзакции

```

100 %

Results Messages

	Состояние	Владелец	Устройства
1	До начала транзакции	Женя	4
2	До начала транзакции	Лева	0

```

--Начало транзакции
BEGIN TRANSACTION;

```

```

UPDATE sqlmagic.dbo.Kids
SET DeviceQty -=2
WHERE KidName = N'Женя';

```

```

UPDATE sqlmagic.dbo.Kids
SET DeviceQty +=2
WHERE KidName = N'Лева';

```

Виконуємо транзакцію. В рамках цієї транзакції ми зменшуємо кількість пристроїв у Жені та збільшуємо у Льови.

```

COMMIT TRANSACTION;

```

Далі ми фіксуємо транзакцію.

```

--Транзакция подтверждена
--Остановка SQL NOWAIT;
PRINT 'До остановки SQL Server'
SHUTDOWN WITH NOWAIT;
PRINT 'Возможно после остановки SQL Server?';

```

А далі ми зупинимо весь екземпляр SQL Server, при цьому ніякі дані зберігатись не будуть та відразу відбудеться відключення всіх сеансів підключення до цього екземпляру.

```

22 UPDATE sqlmagic.dbo.Kids
23 SET DeviceQty += 2
24 WHERE KidName = N'Лева';
25 COMMIT TRANSACTION;

```

100 %

Results Messages

До остановки SQL Server

Server shut down by NOWAIT request from login ROMA\mag.
SQL Server is terminating this process.

Отже, ми бачимо повідомлення 'До остановки SQL Server', а повідомлення, яке ми намагались вивести після зупинки, в нас його звичайно не вивелось. Але замість цього ми бачимо повідомлення що сервер був зупинений та сеанс підключення був розірваний.

```

SELECT
'После транзакции' AS "Состояние",

```

```
KidName AS "Владелец",
DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids ORDER BY KidName;
```

44Durability.sql - D...A.sqlmagic (sa (59))* 31Pessimistic.sql

```
--Транзакция подтверждена
--Остановка SQL NOWAIT;
PRINT 'До остановки SQL Server'
SHUTDOWN WITH NOWAIT;
PRINT 'Возможно после остановки SQL Server?'
```

```
SELECT
'После транзакции' AS "Состояние",
KidName AS "Владелец",
DeviceQty AS "Устройства"
FROM sqlmagic.dbo.Kids ORDER BY KidName;
```

100 %

Results Messages

	Состояние	Владелец	Устройства
1	После транзакции	Женя	2
2	После транзакции	Лева	2

Бачимо що стан пристроїв у Жені та Льови – 2. Тобто, та транзакція, яка була підтверджена, успішно зберіглась та зупинка SQL Server відразу після виконання транзакції, не завадила збереженню змін.

Як влаштований журнал транзакцій логічно

```
USE [master];
GO
IF DB_ID('SimpleModel') IS NOT NULL
BEGIN
ALTER DATABASE [SimpleModel] --Перевіряємо чи немає бази даних SimpleModel
SET RESTRICTED_USER WITH ROLLBACK IMMEDIATE;
DROP DATABASE [SimpleModel]; --ЯКЩО така є, то ми її видаляємо
END
```

```
CREATE DATABASE [SimpleModel] --після того як ми базу даних видалили або її не було
спочатку, то ми створюємо дану базу даних
ON PRIMARY
```

```
(
NAME = N'SimpleModel',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\DATA\SimpleModel.mdf',
SIZE = 4288KB, MAXSIZE = UNLIMITED,
FILEGROWTH = 1024KB
)
LOG ON
(
NAME = N'SimpleModel_log',
FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL14.TATA\MSSQL\Log\SimpleModel_log.ldf',
SIZE = 1MB, MAXSIZE = 2048GB, FILEGROWTH = 10%
);
GO
ALTER DATABASE [SimpleModel]
SET RECOVERY SIMPLE
```

```
00СозданиеБд.sql - ...TA.master (sa (52)) * X
USE [master];
GO
IF DB_ID('SimpleModel') IS NOT NULL
BEGIN
ALTER DATABASE [SimpleModel]
SET RESTRICTED_USER WITH ROLLBACK IMMEDIATE;
DROP DATABASE [SimpleModel];
END
CREATE DATABASE [SimpleModel]
ON PRIMARY
(
NAME = N'SimpleModel',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\DATA\SimpleModel.mdf',
SIZE = 4288KB, MAXSIZE = UNLIMITED,
FILEGROWTH = 1024KB
)
LOG ON
(
NAME = N'SimpleModel_log',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\Log\SimpleModel_log.ldf',
SIZE = 1MB, MAXSIZE = 2048GB, FILEGROWTH = 10%
);
GO
ALTER DATABASE [SimpleModel]
100 %
Messages
Commands completed successfully.
```

База даних створена, з використанням моделі Simple, за допомогою інструкції ALTER DATABASE переводимо її в просту модель відновлення, тобто Simple.

```
USE SimpleModel;
SELECT
*
FROM sys.fn_dblog(NULL, NULL);
```

Дана функція дозволяє нам подивитись журнал транзакцій, а саме активну частину журналу транзакцій, тобто ту частину, яка зараз використовується. В функції два параметри, які ми вказуємо, в даному випадку ми вказуємо що хочемо побачити всі записи журналу транзакцій, які доступні.

```
02fn_dblog.sql - D...mpleModel (sa (56)) X 00СозданиеБд.sql - ...TA.master (sa (52)) *
USE [SimpleModel];
SELECT
FROM sys.fn_dblog(NULL, NULL);
DBCC LOG('SimpleModel', -1) WITH TABLERESULTS,
NO_INFOMSGS;
100 %
Results Messages
Current LSN Operation Context Transaction ID LogBlockGeneration Tag Bits Log Record Fixed Length Log Record Length Previous LSN Flag Bits
1 00000024-00000100-0002 LOP_BEGIN_CKPT LCK_NULL 0000-00000000 0 0x0000 96 96 00000024-00000101-0002 0x0000
2 00000024-00000100-0001 LOP_ACT_CKPT LCK_BOOT_PAGE_CKPT 0000-00000000 0 0x0000 24 28 00000000-00000000-0000 0x0000
3 00000024-00000100-0001 LOP_END_CKPT LCK_NULL 0000-00000000 0 0x0000 136 136 00000024-00000100-0002 0x0000
4 00000024-00000100-0001 LOP_BEGIN_XACT LCK_NULL 0000-00000386 0 0x0000 76 120 00000000-00000000-0000 0x0002
5 00000024-00000100-0002 LOP_MODIFY_ROW LCK_BOOT_PAGE 0000-00000386 0 0x0000 62 84 00000024-00000100-0001 0x0002
6 00000024-00000100-0001 LOP_PREP_XACT LCK_NULL 0000-00000386 0 0x0000 64 68 00000024-00000100-0002 0x0002
7 00000024-00000100-0001 LOP_COMMIT_XACT LCK_NULL 0000-00000386 0 0x0000 88 84 00000024-00000100-0001 0x0002
```

Отримуємо набір записів журналу транзакцій. Бачимо, що це дійсно послідовний набір записів, які відрізняються один від одного унікальним ідентифікатором та він монотонно зростає з кожним записом. У кожного запису є визначення операції, яку цей запис виконує.

```
DBCC LOG('SimpleModel', -1) WITH TABLERESULTS,
NO_INFOMSGS;
```

Можна подивитись ще одну можливість переглянути журнал транзакцій це інструкція DBCC LOG. Тут також вводимо два параметри. Перший – назва бази даних, другий – це рівень деталізації.

DBCC LOG('SimpleModel', -1) WITH TABLERESULTS,
NO_INFOMSGS;

Current LSN	Operation	Context	Transaction ID	LogBlockGeneration	Tag Bits	Log Record Fixed Length	Log Record Length	Previous LSN	Flag
00000024:000001bc:0027	LOP_BEGIN_CKPT	LCX_NULL	0000:00000000	0	0x0000	96	96	00000024:000001a1:0002	0x00
00000024:000001cc:0001	LOP_XACT_CKPT	LCX_BOOT_PAGE_CKPT	0000:00000000	0	0x0000	24	28	00000000:00000000:0000	0x00
00000024:000001cd:0001	LOP_END_CKPT	LCX_NULL	0000:00000000	0	0x0000	136	136	00000024:000001bc:0027	0x00
00000024:000001ce:0001	LOP_BEGIN_XACT	LCX_NULL	0000:00000386	0	0x0000	76	120	00000000:00000000:0000	0x00
00000024:000001ce:0002	LOP_MODIFY_ROW	LCX_BOOT_PAGE	0000:00000386	0	0x0000	62	84	00000024:000001ce:0001	0x00
00000024:000001cf:0001	LOP_PREP_XACT	LCX_NULL	0000:00000386	0	0x0000	64	68	00000024:000001ce:0002	0x00
00000024:000001d0:0001	LOP_COMMIT_XACT	LCX_NULL	0000:00000386	0	0x0000	80	84	00000024:000001ce:0001	0x00
00000024:000001d1:0001	LOP_EXPUNGE_R...	LCX_CLUSTERED	0000:00000000	0	0x0000	62	64	00000000:00000000:0000	0x00

Query executed successfully. DESKTOP-D05GEMN\TATA (14.0 ... sa (56) SimpleModel 00:00:00 32 rows

Журнал транзакцій на фізичному рівні

```
--Если log <= 64MB то создается 4VLF
--Не запускать
USE [master];
GO
IF DB_ID('SmallLog') IS NOT NULL
BEGIN
ALTER DATABASE [SmallLog]
SET RESTRICTED_USER WITH ROLLBACK IMMEDIATE;
DROP DATABASE [SmallLog];
END
CREATE DATABASE [SmallLog]
ON PRIMARY
(
NAME = N'SmallLog',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\DATA\SmallLog.mdf',
SIZE = 4288KB, MAXSIZE = UNLIMITED,
FILEGROWTH = 1024KB
)
LOG ON
(
NAME = N'SmallLog_log',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\Log\SmallLog_log.ldf',
SIZE = 64MB, MAXSIZE = 2048GB, FILEGROWTH = 10%
);
GO
```

При цьому, при створенні бази даних, ми вказуємо розмір файлу журналу транзакцій в 64MB або менше. При такому розмірі, а саме 64MB у нас має вийти 4 файли.

```
DBCC LOGININFO('SmallLog') WITH TABLERESULTS,
NO_INFOMSGS;
```

RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	0	16711680	8192	36	2	64	0
2	0	16711680	16719872	0	0	0	0
3	0	16711680	33431552	0	0	0	0
4	0	16965632	50143232	0	0	0	0

За допомогою даної інструкції ми можемо подивитись інформацію. Після виконання даної інструкції ми бачимо номер фізичного файлу всередині бази

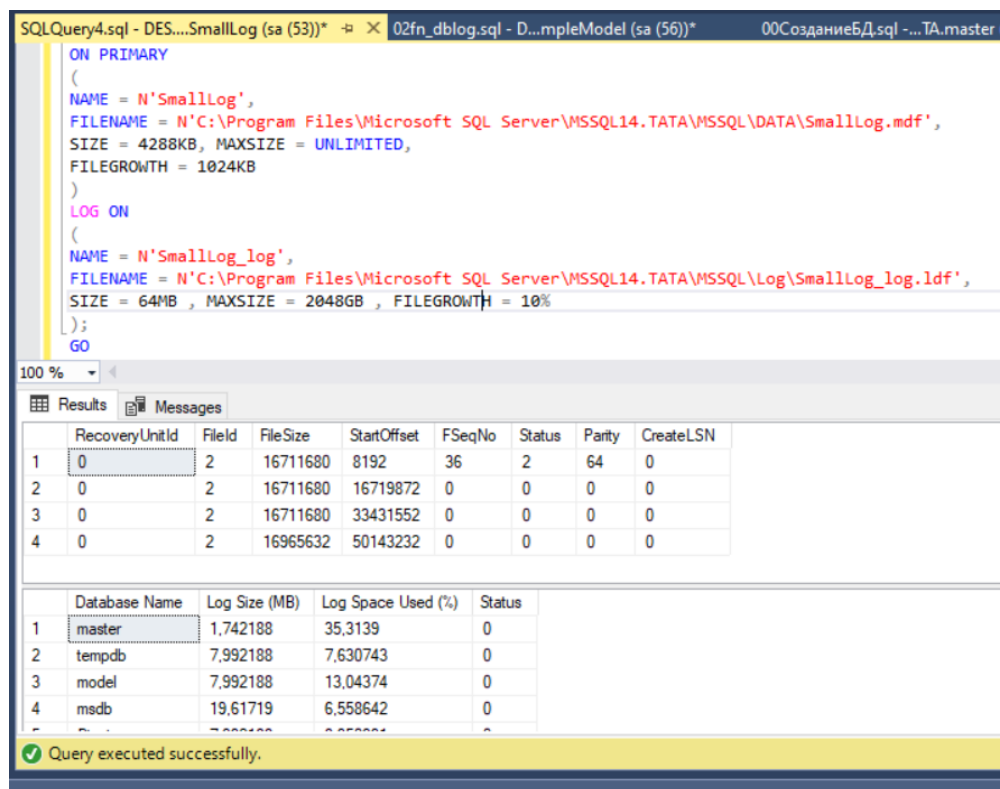
даних. Розмір файлу, здвиг всередині файлу для віртуального логу, параметр статус, він означає що цей віртуальний файл лог зараз зберігає дані, він є активним, його не можна використовувати повторно. 2– активний, 0 – не активний. Також є параметр номеру віртуального логу.

```
USE SmallLog
IF OBJECT_ID('SmallLog..TestLog') IS NULL
CREATE TABLE SmallLog..TestLog
(
    X CHAR(8000)
)
```

Крім цього, ми створюємо таблицю для тестів, один стовбець розміром 8000 символів, для того, щоб можна було заповнювати базу даних різними значеннями.

```
DBCC SQLPERF (LOGSPACE);
```

Також використовується ще одна інструкція, яка дозволяє подивитись розмір в журналі транзакцій баз даних та використовуваний простір.



```
SQLQuery4.sql - DES...SmallLog (sa (53))* 02fn_dblog.sql - D...mpleModel (sa (56))* 00СозданиеБД.sql - ...TA.master (sa (57))
```

```
ON PRIMARY
(
    NAME = N'SmallLog',
    FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\DATA\SmallLog.mdf',
    SIZE = 4288KB, MAXSIZE = UNLIMITED,
    FILEGROWTH = 1024KB
)
LOG ON
(
    NAME = N'SmallLog_log',
    FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\Log\SmallLog_log.ldf',
    SIZE = 64MB , MAXSIZE = 2048GB , FILEGROWTH = 10%
);
GO
```

	RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	0	2	16711680	8192	36	2	64	0
2	0	2	16711680	16719872	0	0	0	0
3	0	2	16711680	33431552	0	0	0	0
4	0	2	16965632	50143232	0	0	0	0

	Database Name	Log Size (MB)	Log Space Used (%)	Status
1	master	1,742188	35,3139	0
2	tempdb	7,992188	7,630743	0
3	model	7,992188	13,04374	0
4	msdb	19,61719	6,558642	0

Query executed successfully.

Отримуємо наступну інформацію, ми переконались що створюється 4 віртуальних файлів журналу, далі ми бачимо розмір для нашої бази даних та простір, який використовується всередині цього блоку.

Розглянемо якщо ми вкажемо розмір більше 64MB.

```
--Если > 64MB AND <= 1 GB то создается 8
--Не запускать
USE [master]
GO
IF DB_ID('MiddleLog') IS NOT NULL
BEGIN
    ALTER DATABASE [MiddleLog]
    SET RESTRICTED_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE [MiddleLog];
END
```

```

END
CREATE DATABASE [MiddleLog]
ON PRIMARY
(
NAME = N'MiddleLog',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\DATA\MiddleLog.mdf',
SIZE = 4288KB, MAXSIZE = UNLIMITED,
FILEGROWTH = 1024KB
)
LOG ON
(
NAME = N'MiddleLog_log',
FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL14.TATA\MSSQL\Log\MiddleLog_log.ldf',
SIZE = 65MB , MAXSIZE = 2048GB , FILEGROWTH = 10%
);
GO

```

```

DBCC LOGINFO('MiddleLog') WITH TABLERESULTS,
NO_INFOMSGS;

```

```

USE MiddleLog
IF OBJECT_ID('MiddleLog..TestLog') IS NULL
CREATE TABLE MiddleLog..TestLog
(
X CHAR(8000)
)
DBCC SQLPERF (LOGSPACE);

```

SQLQuery5.sql - DE...MiddleLog (sa (57))* X 05CREATEsmallLog.s...SmallLog (sa (53)) 02fn_dblog.sql - D...mpleModel (sa (56))

```

SIZE = 4288KB, MAXSIZE = UNLIMITED,
FILEGROWTH = 1024KB
)
LOG ON
(
NAME = N'MiddleLog_log',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\Log\MiddleLog_log.ldf',
SIZE = 65MB , MAXSIZE = 2048GB , FILEGROWTH = 10%
);
GO

```

DBCC LOGINFO('MiddleLog') WITH TABLERESULTS,
NO_INFOMSGS;

USE MiddleLog

100 %

Results Messages

	RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	0	2	8454144	8192	36	2	64	0
2	0	2	8454144	8462336	0	0	0	0
3	0	2	8454144	16916480	0	0	0	0
4	0	2	8454144	25370624	0	0	0	0
5	0	2	8454144	33824768	0	0	0	0
6	0	2	8454144	42278912	0	0	0	0
7	0	2	8454144	50733056	0	0	0	0
8	0	2	8970240	59187200	0	0	0	0

Database Name Log Size (MB) Log Space Used (%) Status

Query executed successfully.

В даному випадку при створенні бази даних ми вказуємо розмір журналу транзакцій в 65MB але не більше 1ГБ, якщо в попередньому випадку в нас створювалось 4 віртуальних файлів журналу, то в даному випадку в нас виходить 8 віртуальних файлів журналу.

Ще один приклад, якщо більше 1ГБ

```
--Если > 1 GB то создается 16 VLF
--Не запускать
USE [master]
GO
IF DB_ID('BigLog') IS NOT NULL
BEGIN
ALTER DATABASE [BigLog]
SET RESTRICTED_USER WITH ROLLBACK IMMEDIATE;
DROP DATABASE [BigLog];
END
CREATE DATABASE [BigLog]
ON PRIMARY
(
NAME = N'BigLog',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\DATA\BigLog.mdf',
SIZE = 4288KB, MAXSIZE = UNLIMITED,
FILEGROWTH = 1024KB
)
LOG ON
(
NAME = N'BigLog_log',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\Log\BigLog_log.ldf',
SIZE = 1025MB , MAXSIZE = 2048GB , FILEGROWTH = 10%
);
GO
```

```
DBCC LOGININFO('BigLog') WITH TABLERESULTS,
NO_INFOMSGS;
```

```
USE MiddleLog
IF OBJECT_ID('BigLog..TestLog') IS NULL
CREATE TABLE BigLog..TestLog
(
X CHAR(8000)
)
DBCC SQLPERF (LOGSPACE);
```

SQLQuery6.sql - DE...MiddleLog (sa (55))* -> X 07/createMiddle.sql...MiddleLog (sa (57)) 05CREATEsmallLog.s...SmallLog (sa (53)) 02fr

```
END
CREATE DATABASE [BigLog]
ON PRIMARY
(
NAME = N'BigLog',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\DATA\BigLog.mdf',
SIZE = 4288KB, MAXSIZE = UNLIMITED,
FILEGROWTH = 1024KB
)
LOG ON
(
NAME = N'BigLog_log',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\Log\BigLog_log.ldf',
SIZE = 1025MB , MAXSIZE = 2048GB , FILEGROWTH = 10%
```

100 %

Results Messages

	RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
8	0	2	67108864	469770240	0	0	0	0
9	0	2	67108864	536879104	0	0	0	0
10	0	2	67108864	603987968	0	0	0	0
11	0	2	67108864	671096832	0	0	0	0
12	0	2	67108864	738205696	0	0	0	0
13	0	2	67108864	805314560	0	0	0	0
14	0	2	67108864	872423424	0	0	0	0
15	0	2	67108864	939532288	0	0	0	0
16	0	2	68149248	1006641...	0	0	0	0

Database Name	Log Size (MB)	Log Space Used (%)	Status
---------------	---------------	--------------------	--------

В даному випадку, ми бачимо, що створюється 16 віртуальних файлів журналу, але логічно він 1.

Тепер ми створили базу даних, та в ній буде декілька файлів.

```
--Если log <= 64MB то создается 4VLF
USE [master]
GO
USE master
IF DB_ID('SeveralFiles') IS NOT NULL
BEGIN
ALTER DATABASE [SeveralFiles]
SET RESTRICTED_USER WITH ROLLBACK IMMEDIATE;
DROP DATABASE [SeveralFiles];
END
CREATE DATABASE [SeveralFiles]
ON PRIMARY
(
NAME = N'SeveralFiles',
FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL14.TATA\MSSQL\DATA\SeveralFiles.mdf',
SIZE = 4288KB, MAXSIZE = UNLIMITED,
FILEGROWTH = 1024KB
)
LOG ON
(
NAME = N'SeveralFiles_log1',
FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL14.TATA\MSSQL\Log\SeveralFiles_log1.ldf',
SIZE = 1MB , MAXSIZE = 2048GB , FILEGROWTH = 1MB
),
(
NAME = N'SeveralFiles_log2',
FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL14.TATA\MSSQL\Log\SeveralFiles_log2.ldf',
SIZE = 1MB , MAXSIZE = 2048GB , FILEGROWTH = 1MB
);
GO

DBCC LOGINFO('SeveralFiles') WITH TABLERESULTS,
NO_INFOMSGS;

ALTER DATABASE SeveralFiles
SET RECOVERY SIMPLE
GO
INSERT SeveralFiles..TestLog DEFAULT VALUES
GO 15
DBCC LOGINFO('SeveralFiles') WITH TABLERESULTS,
NO_INFOMSGS;

ALTER DATABASE [SeveralFiles]
REMOVE FILE [SeveralFiles_log1]
GO
ALTER DATABASE [SeveralFiles]
REMOVE FILE [SeveralFiles_log2]
GO
```

11НесколькоФайлов...A.master (sa (52))*

```

(
NAME = N'SeveralFiles_log1',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\Log\SeveralFiles_log1.ldf',
SIZE = 1MB , MAXSIZE = 2048GB , FILEGROWTH = 1MB
),
(
NAME = N'SeveralFiles_log2',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.TATA\MSSQL\Log\SeveralFiles_log2.ldf',
SIZE = 1MB , MAXSIZE = 2048GB , FILEGROWTH = 1MB
);
GO

```

121 %

Results Messages

	RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	0	2	253952	8192	36	2	64	0
2	0	2	253952	262144	0	0	0	0
3	0	2	253952	516096	0	0	0	0
4	0	2	278528	770048	0	0	0	0
5	0	3	253952	8192	0	0	0	0
6	0	3	253952	262144	0	0	0	0
7	0	3	253952	516096	0	0	0	0
8	0	3	278528	770048	0	0	0	0

Бачимо 4 файли з ідентифікатором 2, та 4 з ідентифікатором 3.

За допомогою інструкції ми спробуємо видалити файли з ідентифікатором 3.

11НесколькоФайлов...A.master (sa (52))*

```

ALTER DATABASE [SeveralFiles]
REMOVE FILE [SeveralFiles_log1]
GO
ALTER DATABASE [SeveralFiles]
REMOVE FILE [SeveralFiles_log2]
GO

```

121 %

Messages

The file 'SeveralFiles_log2' has been removed.

121 %

Можемо побачити що SQL Server дозволяє нам видалити файл log2, бо в ньому немає активних віртуальних логів. А файл log1 ми видалити не можемо, бо в ньому є активні віртуальні логи та він в нас являється первинним, його видалити не можна, навіть у випадку, якщо в ньому не буде активних логів.

```

DBCC LOGINFO ('BigLog') WITH TABLERESULTS,
NO_INFOMSGS;

```

```
--16
USE [master]
go
ALTER DATABASE [BigLog] MODIFY FILE (NAME = N'BigLog_log', SIZE = 1026MB)
GO
DBCC LOGINFO ('BigLog') WITH TABLERESULTS,
NO_INFOMSGS;
ALTER DATABASE [BigLog] MODIFY FILE (NAME = 'BigLog_log', SIZE = 2GB)
GO
DBCC LOGINFO ('BigLog') WITH TABLERESULTS,
NO_INFOMSGS;
```

```
ALTER DATABASE [BigLog] MODIFY FILE (NAME = N'BigLog_log', SIZE = 1026MB)
GO
```

```
SQLQuery1.sql - DES...TA.master (sa (53))* -> X
DBCC LOGINFO ('BigLog') WITH TABLERESULTS,
NO_INFOMSGS;
--16
USE [master]
go
ALTER DATABASE [BigLog] MODIFY FILE (NAME = N'BigLog_log', SIZE = 1026MB)
GO
DBCC LOGINFO ('BigLog') WITH TABLERESULTS,
NO_INFOMSGS;
121 %
Messages
Commands completed successfully.
```

Збільшуємо розмір файлу, було 1025МБ, а ми змінюємо на 1026МБ.

```
DBCC LOGINFO ('BigLog') WITH TABLERESULTS,
NO_INFOMSGS;
```

```
SQLQuery1.sql - DES...TA.master (sa (53))* -> X
USE [master]
go
ALTER DATABASE [BigLog] MODIFY FILE (NAME = N'BigLog_log', SIZE = 1026MB)
GO
DBCC LOGINFO ('BigLog') WITH TABLERESULTS,
NO_INFOMSGS;
ALTER DATABASE [BigLog] MODIFY FILE (NAME = 'BigLog_log', SIZE = 2GB)
GO
DBCC LOGINFO ('BigLog') WITH TABLERESULTS,
NO_INFOMSGS;
121 %
Results
Messages
```

	RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateTime
12	0	2	67108864	738205696	0	0	0	0
13	0	2	67108864	805314560	0	0	0	0
14	0	2	67108864	872423424	0	0	0	0
15	0	2	67108864	939532288	0	0	0	0
16	0	2	68149248	1006641...	0	0	0	0
17	0	2	1048576	1074790...	0	0	0	360

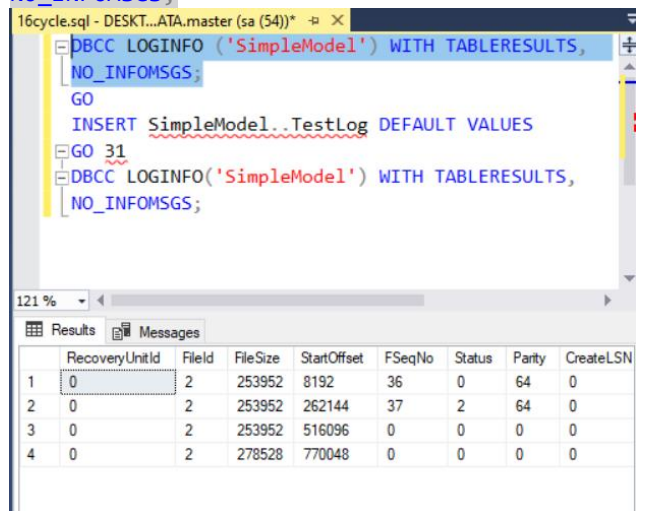
Query e... DESKTOP-DO5GEMN\TATA (14.0 ... sa (53) master 00:00:00 17 rows

Бачимо що в нас додався ще один віртуальний лог 17, розміром 1ГБ.

Таким чином, якщо ми збільшуємо розмір журналу транзакцій на невелике значення, ми отримуємо велику кількість маленьких віртуальних логів.

Повторне використання журналу транзакцій

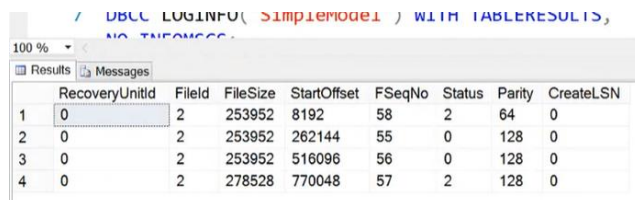
```
DBCC LOGINFO ('SimpleModel') WITH TABLERESULTS,
NO_INFOMSGS;
```



RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
0	2	253952	8192	36	0	64	0
2	0	253952	262144	37	2	64	0
3	0	253952	516096	0	0	0	0
4	0	278528	770048	0	0	0	0

Бачимо що в даній базі даних 4 віртуальні файли, при цьому, бачимо, що тільки у одного з них статус 2, тобто він активний.

```
GO
INSERT SimpleModel..TestLog DEFAULT VALUES
GO 31
DBCC LOGINFO('SimpleModel') WITH TABLERESULTS,
NO_INFOMSGS;
```



RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
0	2	253952	8192	58	2	64	0
2	0	253952	262144	55	0	128	0
3	0	253952	516096	56	0	128	0
4	0	278528	770048	57	2	128	0

Зараз ми бачимо що 1 та 4 віртуальні файли мають статус 2, тобто, вони активні.

Збільшуємо розмір журналу транзакцій

```
USE SimpleModel
DBCC LOGINFO('SimpleModel') WITH TABLERESULTS,
NO_INFOMSGS;
--Раз

BEGIN TRAN
SELECT @@TRANCOUNT

--Два
INSERT SimpleModel..TestLog DEFAULT VALUES
GO 30
DBCC LOGINFO ('SimpleModel') WITH TABLERESULTS,
NO_INFOMSGS;

--Три
WHILE (@@TRANCOUNT > 0) COMMIT
```

```
DBCC LOGINFO('SimpleModel') WITH TABLERESULTS,  
NO_INFOMSGS;
```

```
CHECKPOINT
```

```
DBCC LOGINFO('SimpleModel') WITH TABLERESULTS,  
NO_INFOMSGS;
```

The screenshot shows a SQL query window titled 'SQLQuery3.sql - DES...TA.master (sa (55))'. The query contains the following T-SQL code:

```
USE SimpleModel  
DBCC LOGINFO('SimpleModel') WITH TABLERESULTS,  
NO_INFOMSGS;  
--Раз  
  
BEGIN TRAN  
SELECT @@TRANCOUNT  
  
--Два  
INSERT SimpleModel..TestLog DEFAULT VALUES
```

The 'Results' tab is active, displaying a table with 9 columns: RecoveryUnitId, FileId, FileSize, StartOffset, FSeqNo, Status, Parity, and CreateLSN. The table contains 4 rows of data.

	RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	0	2	253952	8192	36	0	64	0
2	0	2	253952	262144	37	2	64	0
3	0	2	253952	516096	0	0	0	0
4	0	2	278528	770048	0	0	0	0

The screenshot shows the same SQL query window as above. The 'Results' tab is active, displaying a single row with the value 1 in a column labeled '(No column name)'.

(No column name)
1

Починаємо транзакцію

The screenshot shows the same SQL query window. The 'Messages' tab is active, displaying the following text:

```
Beginning execution loop  
Batch execution completed 30 times.
```

Виконується 30 разів.

```
17 INSERT SimpleModel..TestLog DEFAULT VALUES
18 GO 30
```

100 %

Results Messages

	RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	0	2	253952	8192	70	2	128	0
2	0	2	253952	262144	67	0	64	0
3	0	2	253952	516096	68	0	64	0
4	0	2	278528	770048	69	2	64	0

Бачимо що вільне місце на початку файлу журналу транзакцій дозволило нам обійти збільшення розміру.

```
18 GO 30
```

100 %

Results Messages

	RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	0	2	253952	8192	70	2	128	0
2	0	2	253952	262144	71	2	128	0
3	0	2	253952	516096	68	0	64	0
4	0	2	278528	770048	69	2	64	0
5	0	2	262144	1048576	0	0	0	71000000013600053

В нас з'явився 5 віртуальний файл логу, йому поки що не присвоєний номер. Він буде присвоєний при наступній операції.

```
18 GO 30
```

100 %

Results Messages

	RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	0	2	253952	8192	70	2	128	0
2	0	2	253952	262144	71	2	128	0
3	0	2	253952	516096	68	0	64	0
4	0	2	278528	770048	69	2	64	0
5	0	2	262144	1048576	72	2	64	71000000013600053
6	0	2	262144	1310720	0	0	0	72000000012800039

Тут вже можна побачити, що номер присвоєний – 72.

```

SQLQuery3.sql - DES...TA.master (sa (55)) *
NO_INFOMSGS;

--Три
WHILE (@@TRANCOUNT > 0) COMMIT
DBCC LOGININFO('SimpleModel') WITH TABLERE
NO_INFOMSGS;

CHECKPOINT
DBCC LOGININFO ('SimpleModel') WITH TABLER
NO_INFOMSGS;

```

121 %

Messages

Commands completed successfully.

```

25
26
27 CHECKPOINT
28 DBCC LOGININFO('SimpleModel') WITH TABLERESULTS,
NO_INFOMSGS;

```

100 %

Results Messages

	RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
5	0	2	262144	1048576	72	0	64	7100000001360005
6	0	2	262144	1310720	73	0	64	7200000001280003
7	0	2	262144	1572864	74	0	64	730000000100720001
8	0	2	262144	1835008	75	2	64	73000000003840005
9	0	2	262144	2097152	0	0	0	74000000004000000

Велика частина файлів будуть неактивні, збільшується кількість вільних віртуальних файлів.

Зменшуємо розмір журналу транзакцій

```

DBCC LOGININFO ('SimpleModel') WITH TABLERESULTS,
NO_INFOMSGS;
GO
INSERT SimpleModel..TestLog DEFAULT VALUES
GO 30
DBCC LOGININFO('SimpleModel') WITH TABLERESULTS,
NO_INFOMSGS;

USE SimpleModel
DBCC SHRINKFILE ('SimpleModel_log', 0);
go
DBCC LOGININFO('SimpleModel') WITH TABLERESULTS,
NO_INFOMSGS;

```

```

25 USE SimpleModel
26 DBCC SHRINKFILE ('SimpleModel_log', 0);
27
28

```

00 %

Results Messages

	RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	0	2	253952	8192	70	0	128	0
2	0	2	253952	262144	71	0	128	0
3	0	2	253952	516096	68	0	64	0
4	0	2	278528	770048	69	0	64	0
5	0	2	262144	1048576	72	0	64	7100000001360005

Бачимо, що в кінці файлів є вільне місце. Ми вписуємо два параметри, назва файлу та розмір файлу для зменшення.

USE SmallLog
SELECT

100 %

Results Messages

	DbId	FileId	CurrentSize	MinimumSize	UsedPages	EstimatedPages
1	7	2	128	128	128	128

	RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	0	2	253952	8192	70	0	128	0
2	0	2	253952	262144	71	0	128	0
3	0	2	253952	516096	76	2	128	0
4	0	2	278528	770048	69	0	64	0

Зараз у нас файл журналу транзакцій зменшився, кількість віртуальних логів зменшилась, зараз в нас 4 віртуальних файли.

LSN – послідовний номер в журналі транзакцій

```
USE SmallLog
SELECT
[Current LSN],
SmallLog.dbo.ufn_LSNPart("Current LSN", 1) AS VLF,
SmallLog.dbo.ufn_LSNPart("Current LSN", 2) AS Offset,
SmallLog.dbo.ufn_LSNPart("Current LSN", 3) AS Slot,
*
FROM sys.fn_dblog(NULL, NULL);
DBCC LOGINFO('SmallLog') WITH TABLERESULTS,
NO_INFOMSGS;
```

```
8 USE SmallLog
9 SELECT
10 [Current LSN]
11 ,SmallLog.dbo.ufn_LSNPart("Current LSN", 1) AS VLF
12 ,SmallLog.dbo.ufn_LSNPart("Current LSN", 2) AS Offset
```

100 %

Results Messages

	Current LSN	VLF	Offset	Slot	Current LSN	Operation
1	0000002a:00000018:0001	42	24	1	0000002a:00000018:0001	LOP_BEGIN_X
2	0000002a:00000018:0002	42	24	2	0000002a:00000018:0002	LOP_MODIFY
3	0000002a:00000020:0001	42	32	1	0000002a:00000020:0001	LOP_MODIFY
4	0000002a:00000028:0001	42	40	1	0000002a:00000028:0001	LOP_MODIFY
5	0000002a:00000028:0002	42	40	2	0000002a:00000028:0002	LOP_MODIFY
6	0000002a:00000030:0001	42	48	1	0000002a:00000030:0001	LOP_DELETE

LSN в першому рядку відповідає віртуальному файлу – 42, здвигу – 24, та першому слоту.

