Министерство науки и высшего образования Российской Федерации

1. Санкт-Петербургский Политехнический Университет Петра Великого
2. —
3. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 3**

1. **«Физическая организация хранения данных»**
2. по дисциплине «Системы управления базами данных»
3. Выполнил
4. студент гр. 4851003/70801 Гасанов Э.А.

<*подпись*>

* + - 1. Преподаватель

1. доцент, к.т.н. Полтавцева М.А.

<*подпись*>

1. Санкт-Петербург
2. 2021

**Цель работы**

Получить навыки оценки размера физического хранилища данных и работы с ним.

# Задачи

1. Определите размер файлов пользовательских и системных баз данных (учтите, что в ряде систем, например, Oracle, одному экземпляру сервера советует единственное понятие базы данных). Определите размер экстента.
2. Создайте таблицу хранения лога действий пользователя в системе, созданной ранее. Определите структуру таблицы и ключи.
3. Оцените рост хранилища данных. Для этого:
   1. Определите для пустой таблицы, сколько места она занимает. Для этого:
      1. Воспользуйтесь запросом к системным таблицам сервера или предустановленным процедурам.
      2. Оцените размер физического хранения файла на диске
   2. Добавьте в таблицу некоторое (от 10) количество записей.
   3. Повторите оценку пространства.
      1. Если оно не изменилось, удвойте количество данных, пока разница не станет заметна.
      2. Определите размер одной записи.
   4. Повторите проверку с дальнейшим увеличением числа записей (с приростом не менее пяти - шести раз).
4. Определите реальный средний размер одной записи. Для этого:
   1. По методике выше определите точную границу (в записях), на которой происходит выделение нового блока данных.
   2. Установите размер записи на основании заполненного дискового пространства.
   3. Повторите для следующего выделения записи для уточнения результат.
5. Определите, какой объем понадобиться для хранения N записей. Обоснуйте результат.
6. Проверьте полученный результат, увеличив заполненное число записей в несколько раз (не менее пяти – шести проверок).
7. Проведите секционирование таблицы и его оценку. Для этого:
   1. Очистите таблицу. Автоматически заполните ее значениями до размера в 1 000 000 записей (или менее/более в зависимости от оборудования).
   2. Если обработка записей происходит очень быстро (или очень медленно) их количество можно уменьшить (или, соответственно, увеличить) до получения таблицы с большой (несколько минут, десятков минут) задержкой на операции простой выборки данных.
   3. Организуйте секционирование созданной таблицы средствами СУБД (если они доступны) и вручную.
   4. Организуйте (при возможности) размещение секций на разных дисках.
8. Оцените скорость выполнения трех запросов к секционированным данным:
   1. Для варианта размещения на одном диске (не менее 10 тестов)

**Ход работы**

Для выполнения данной лабораторной работы была использована база данных, где существует таблица T\_Log. В таблице содержатся имена сотрудников учебного центра, их предмет, и время выполнения этих действий.

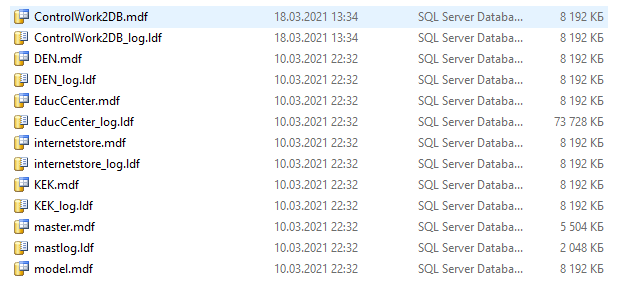


Рисунок 1 - Содержимое каталога с файлами БД.

Расширение .mdf расшифровывается как Master Database File. Файлы с этим расширениям хранят основную информацию о БД.

Расширение .ldf расшифровывается как Log Database File. Файлы с таким расширениям хранят информацию о транзакциях и прочих изменениях.

Файлы с расширением .ndf являются вторичными файлами данных, не являются обязательными, и используются для различных целей.

MS SQL используюет Extent равный 64Кб и состоящий из 8 страниц по 8Кб каждая.

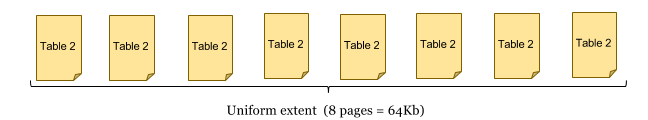


Рисунок 2 – экстент MS-SQL.

Пустая таблица не занимает место на диске:

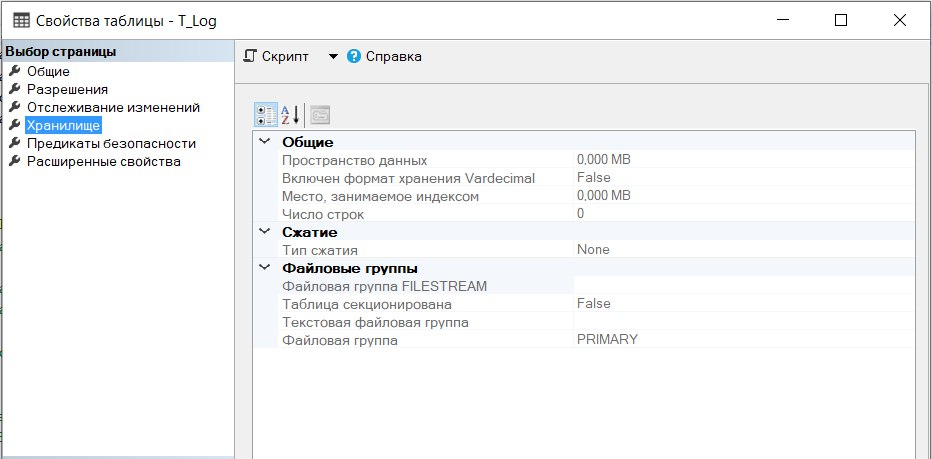


Рисунок 3- Размер пустой таблицы.

Далее размер таблицы узнавался с помощью команды sp\_spaceused T\_Log. Данные для пустой таблицы, и таблице из 10 записей одинакового размера представлены на рисунке 4.



Рисунок 4 - Размер пустой таблицы и таблицы с 10 записями.

Как видно из рисунка, СУБД резервирует место экстентами, но выделяет страницами.

Была найдена граница страницы: при добавлении 84-ой записи 1 страницы уже не хватало и происходило выделение новой страницы(16Кб вместо 8 Кб). Исходя из этого, посчитаем **размер 1 записи** как байт.

Следующая граница достигается при добавлении 167-ой записи. Размер 1 записи при такой оценке не изменится, так как число записей и число страниц изменилось ровно в 2 раза. Последующие удвоения приносили такие же результаты. Шаг по 83 байтов.

Исходя из полученных результатов, можно определить объём памяти, необходимый для хранения N записей. При наличии в таблице хотя бы 1 записи, система выделяет 72Кб, из которых 8Кб используется для индекса, а остальное для хранения записей. Однако при резервировании второй страницы для данных, для индекса также резервируется вторая страница, и всё последующее время он занимает 16Кб.

Таким образом из первого экстента для хранения данных используется лишь 56Кб, то есть

(72Кб-(8Кб+8Кб)=56 Кб)

Последующие выделения экстентов происходят по 64Кб и все 64Кб используются для хранения записей.

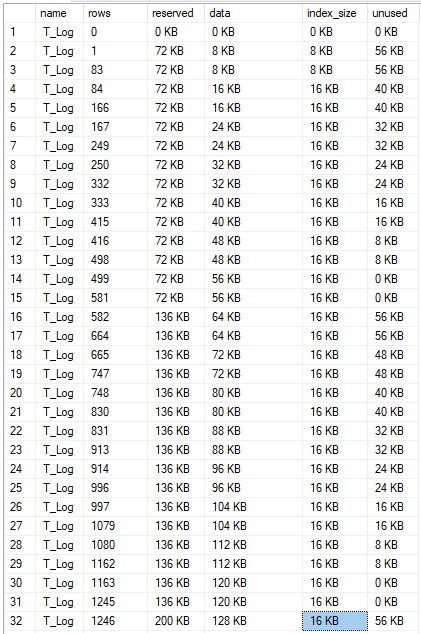


Рисунок 5 - Зависимость зарезервированного пространства от числа записей.

Таким образом, объём памяти в байтах, необходимый для хранения N записей можно вычислить по следующей формуле:

Где 581 — это максимальное число записей, умещающееся в первый экстент. После этого момента выделяется новый экстент.

Теперь попробуем в разы увеличить размер таблицы, и проверить закономерность. Результат представлен на рисунке 6.

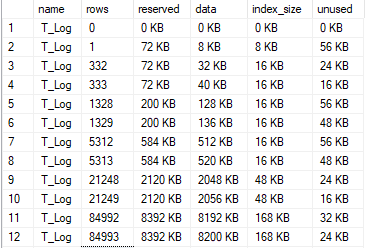


Рисунок 6- Размеры таблицы при значительном увеличении числа записей.

Можно заметить, что в определённый момент index\_size становится больше 16Кб, и продолжает увеличиваться, что делает приведённую ранее формулу для расчёта объёма таблицы непригодной для больших значений N. Однако, оценка размера 1 записи всё ещё остаётся верной. Можно сделать вывод, что место на странице, в которое не поместилась ни одна запись, теряется.

Заполнение таблицы в цикле с помощью INSERT занимает порядка 3 минут времени. В связи с этим дальнейшие операции будут происходить над таблицей указанного размера.

Было организовано секционирование таблицы средствами СУБД MS SQL. Секционирование было настроено с помощью «мастера создания секций». В качестве столбца секционирования был выбран столбец дат Date\_Time. Таблица заполнена таким образом, что каждая запись имеет уникальную дату, и даты идет по возрастанию с интервалом в 1 секунду.

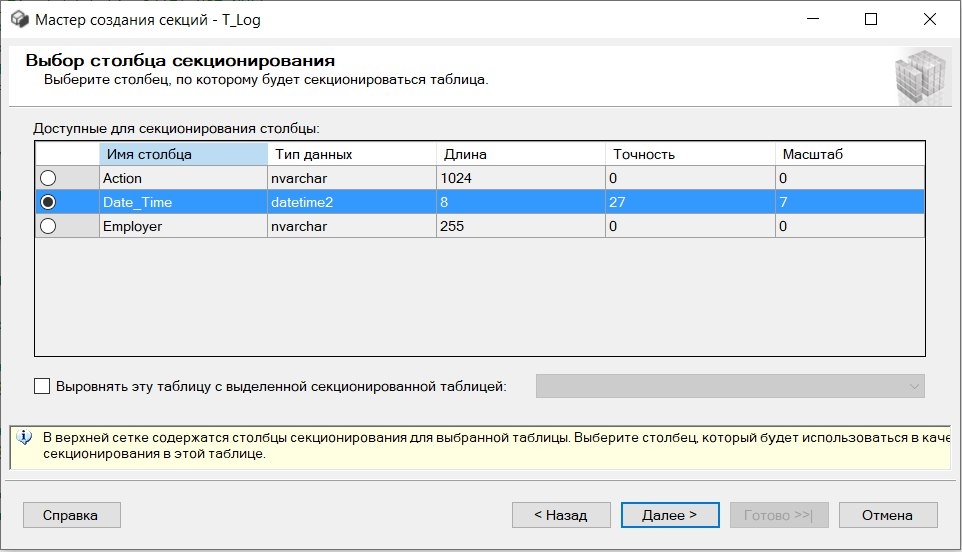


Рисунок 7 – Оснастка мастера создания секций.

В качестве функции секционирования было использование разделение таблицы на 2 равных части с значением даты на границе «2023-03-08 18:53:20.0000000». Был сгенерирован скрипт, выполняющий секционирование. В таблице 1 представлены результаты выполнения запросов к разным частям таблицы.

|  |  |  |
| --- | --- | --- |
| SQL-код запроса | Среднее время запроса за 10 попыток | Позиция выбираемых данных |
| select Date\_Time from T\_Log where Date\_Time = '2023-03-08 15:52:00.000001' | 35 ms | Первая секция |
| select Date\_Time from T\_Log where Date\_Time = '2023-03-08 19:30:04.000001' | 29 ms | Граница секций (обе секции) |
| select Date\_Time from T\_Log where Date\_Time = '2023-03-08 20:53:30.000001' | 30 ms | Вторая секция |
| select Date\_Time from T\_Log where Date\_Time = '2023-03-08 19:31:09.000001' | 37 ms | Без секционирования |

Таблица 1. Запросы к секционированной таблице (средствами СУБД)

Затем было осуществлено ручное секционирование. Для этого было создано 2 таблицы с заголовками, идентичными T\_Log. Была создана процедура для вставки данных в таблицы, которая вставляет данные в одну из двух таблиц, руководствуясь тем же правилом, что и при встроенном секционировании. Тогда после вставки данные имеют следующий вид:

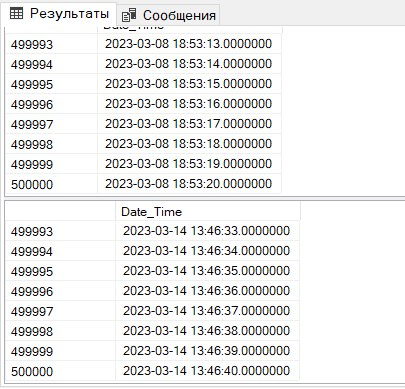


Рисунок 8 – Заполненные T\_Log1 и T\_Log2.

Было создано представление(содержание выбирается из других таблиц с помощью выполнения запроса), объединяющее обе таблицы.

Наконец, к таблице, секционированной вручную, было выполнено 3 запроса, таких же, как для встроенного секционирования. Результаты приведены в таблице 2.

|  |  |  |
| --- | --- | --- |
| SQL-код запроса | Среднее время запроса за 10 попыток | Позиция выбираемых данных |
| select Date\_Time from T\_Log\_view where Date\_Time = '2023-03-08 15:52:00.000001' | 41 мс | Первая секция |
| select Date\_Time from T\_Log\_view where Date\_Time = '2023-03-08 19:30:04.000001' | 38 мс | Граница секций (обе секции) |
| select Date\_Time from T\_Log\_view where Date\_Time = '2023-03-08 20:53:30.000001' | 54 мс | Вторая секция |

Таблица 2. Запросы к секционированной таблице (ручное секционирование)

При ручном секционировании среднее время запросов оказалось хуже автоматического секционирования.

**Вывод**

Были изучены основы организации физического хранения в СУБД на примере MS SQL. В результате выполнения данной лабораторной работы были улучшены навыки работы с MS SQL, в частности умения эффективно заполнять таблицы большим количеством данных. Были получены знания и навыки секционирования таблиц.

# Список литературы

1. Базы данных. Лабораторный практикум. Полтавцева М.А.: учебное пособие / М.А. Полтавцева. – СПб.: Изд-во Политехн. Ун-та, 2016 г.
2. Конспект лекций по СУБД.

# Приложение 1

Скрипт для создания таблицы лога:

CREATE TABLE [dbo].[T\_Log](

[Date\_Time] [datetime2](7) NOT NULL,

[Employer] [nvarchar](255) NOT NULL,

[Action] [nvarchar](1024) NOT NULL,

)

GO

# Приложение 2

Скрипт для автоматической генерации и вставки уникальных записей в таблицу:

-- переменная - начальная дата

Declare @date [datetime2](7)

Set @date = '2023-03-03 00:00:00'

-- переменная счётчик для управления числом записей

Declare @cnt int

Set @cnt = 0

While @cnt < 1000000

Begin

-- инкремент даты на 1 секунду

Set @date = DATEADD(second,1,@date)

-- инкремент счётчика

Set @cnt = @cnt + 1

-- вставка новой уникальноц записи

Insert Into T\_Log values (@date,'Андреев Алексей Альбертович', 'Чтение')

End

# Приложение 3

Скрипт для секционирования с помощью встроенных средств MS SQL:

USE [POTATO]

GO

BEGIN TRANSACTION

CREATE PARTITION FUNCTION [test1](datetime2(7)) AS RANGE LEFT FOR VALUES (N'2023-03-08T18:53:20')

CREATE PARTITION SCHEME [test1\_schema] AS PARTITION [test1] TO ([PRIMARY], [SE

COND])

ALTER TABLE [dbo].[T\_Log] DROP CONSTRAINT [PK\_\_T\_Log\_\_4A953BE508C461EA] WITH ( ONLINE = OFF )

SET ANSI\_PADDING ON

ALTER TABLE [dbo].[T\_Log] ADD PRIMARY KEY CLUSTERED

(

[Date\_Time] ASC,

[Employer] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [test1\_schema]([Date\_Time])

COMMIT TRANSACTION

# Приложение 4

Скрипты для секционирования вручную:

1. Создание двух таблиц

CREATE TABLE [dbo].[T\_Log1](

[Date\_Time] [datetime2](7) NOT NULL,

[Employer] [nvarchar](255) NOT NULL,

[Action] [nvarchar](1024) NOT NULL,

)

GO

CREATE TABLE [dbo].[T\_Log2](

[Date\_Time] [datetime2](7) NOT NULL,

[Employer] [nvarchar](255) NOT NULL,

[Action] [nvarchar](1024) NOT NULL,

)

GO

1. Создание функции для корректной вставки:

CREATE PROC my\_insert @DT datetime2(7), @Emp nvarchar(255), @Act nvarchar(1024)

AS

if(@DT < '2023-04-30 20:53:19.000001')

INSERT INTO T\_Log1 (Date\_Time, Employer, Action) VALUES (@DT, @Emp, @Act);

else

INSERT INTO T\_Log2 (Date\_Time, Employer, Action) VALUES (@DT, @Emp, @Act);

1. Создание объединяющего представления для выборки:

CREATE VIEW T\_Log\_view

AS

select \* from T\_Log1 UNION select \* from T\_Log2