

Мелюхин Павел Васильевич

Отчет по выполнению тестового задания

Предварительная подготовка

Приложение реализовано в виде консольного приложения на языке C# (.NET 8) в среде разработки Visual Studio 2022.

Реализован класс модели Employee и два сервисных класса: ArgsService – для обработки аргументов командной строки и запуска соответствующей задачи; EmployeesService – для взаимодействия с БД EmployeeDB.

Для создания базы данных использована СУБД MS SQL Server v.15.0. Две строки подключения (к базам данных maser и EmployeeDB) хранятся в файле конфигурации appsettings.json.

Асинхронный метод EmployeesService.EnsureCreateDatabaseAsync() содержит код, обеспечивающий создание БД EmployeeDB. Этот метод запускается каждый раз перед выполнением выбранной пользователем задачи. Его суть состоит в отправке на сервер (localdb)\\MSSQLLocalDB запроса

```
IF NOT EXISTS (SELECT name FROM sys.databases WHERE name = 'EmployeeDB') CREATE DATABASE EmployeeDB
```

Задача 1

Асинхронный метод EmployeesService.EnsureCreateEmployeeTableAsync() содержит код, обеспечивающий создание в БД EmployeeDB таблицы Employees со столбцами Fullname, Birthdate и Gender.

Он отправляет в БД EmployeeDB запрос

```
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'Employees') AND type in (N'U'))
```

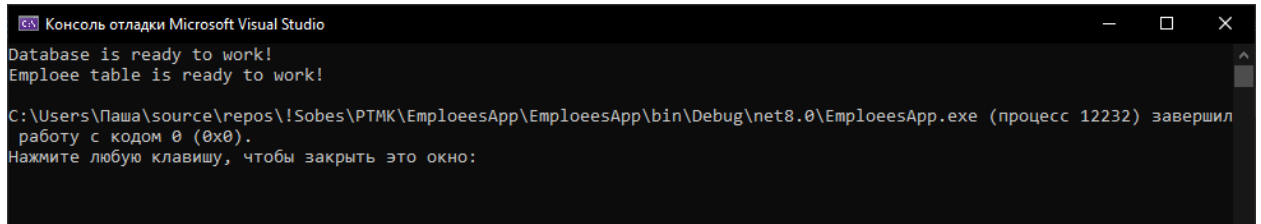
```
CREATE TABLE Employees (Fullname NVARCHAR(100) NOT NULL, Birthdate DATE NOT NULL, Gender BIT NOT NULL);
```

Для столбца Gender был выбран тип данных BIT, соответствующий в C# типу данных bool. Возможно это спорное решение, т.к. в классификаторе ОКИН (Фасет 1) для этого используются целые числа.

На мой взгляд подобная таблица в БД должна сопровождаться уникальным идентификатором, чтобы применять его в качестве первичного ключа. Однако в задании об этом ничего не сказано и я не стал добавлять ничего от себя.

Ниже приведены результаты запуска программы в режиме 1. Консоль отображает сообщения о готовности к работе БД и таблицы Employee, которые можно наблюдать в SQL Server Management Studio.

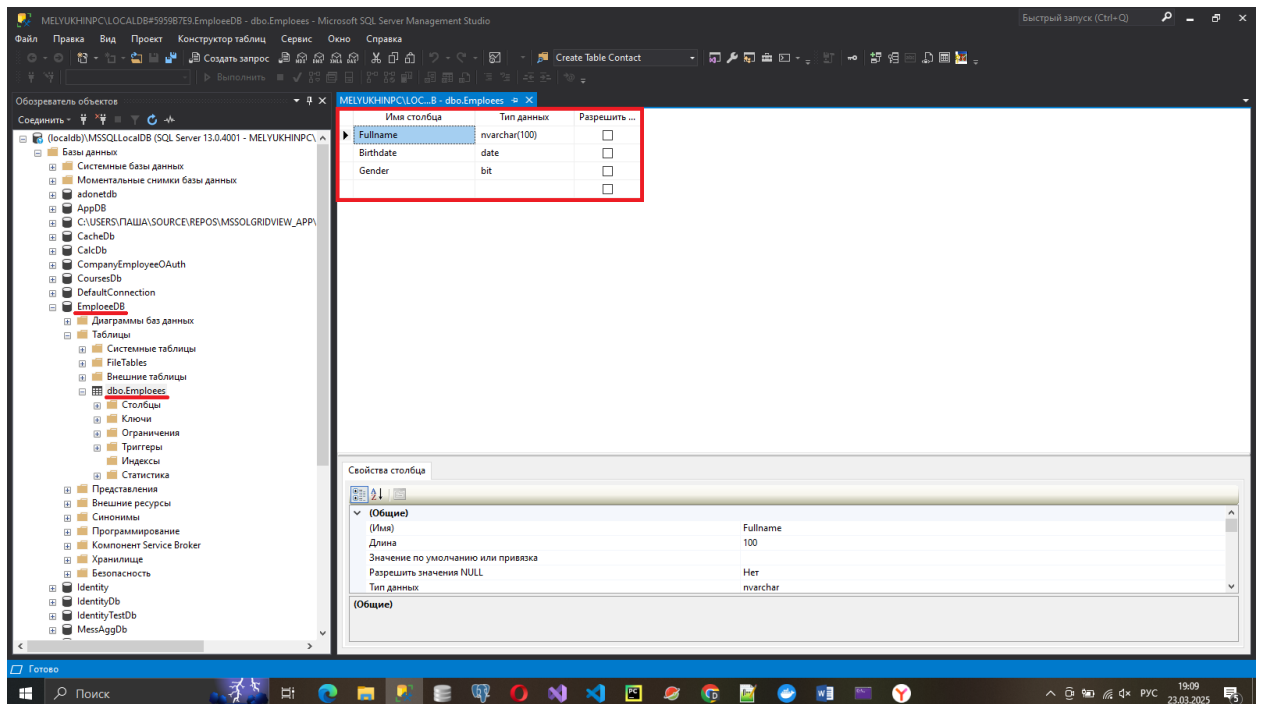
КОНСОЛЬ



```
Консоль отладки Microsoft Visual Studio
Database is ready to work!
Employee table is ready to work!

C:\Users\Паша\source\repos\!Sobes\PTMK\EmployeesApp\EmployeesApp\bin\Debug\net8.0\EmployeesApp.exe (процесс 12232) завершил
работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно:
```

SSMS



Задача 2

При запуске с консоли задачи 2 класс `ArgsService` запускает свой метод `GetEmployee()`, который проверяет аргументы командной строки, необходимые для создания объекта `Employee`, создает и возвращает его. При некорректном вводе приложение выдает сообщение о соответствующей ошибке.

Затем запускается асинхронный метод `EmployeesService.SaveEmployeeAsync(Employee)`. Он создает соединение с БД и вызывает у переданного объекта `Employee` его метод `SaveToDbAsync(SqlConnection)` передавая ему объект `SqlConnection`.

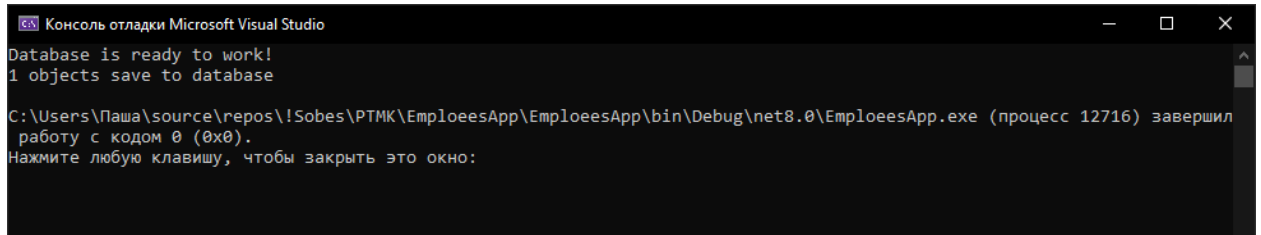
`SaveToDbAsync(SqlConnection)` содержит код, сохраняющий текущий объект `Employee` в базу данных при помощи запроса

```
INSERT INTO Employees (Fullname, Birthdate, Gender) VALUES (@fullname, @birthdate, @gender)
```

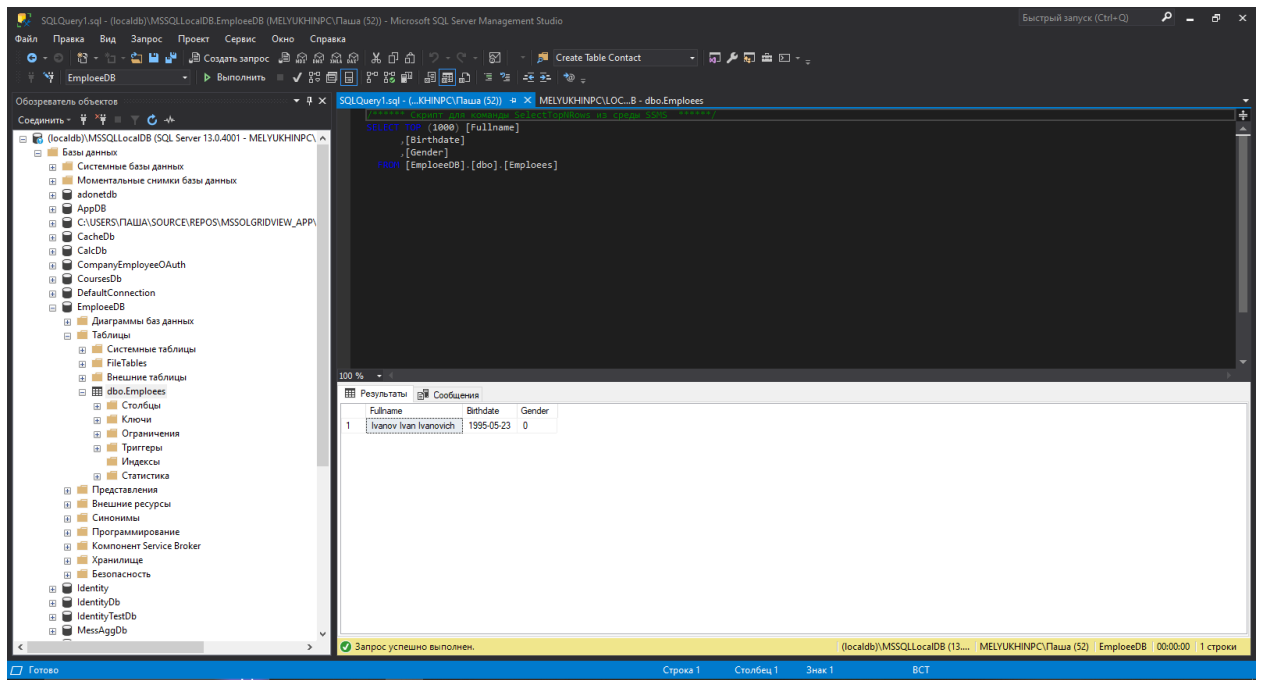
Объект класса `SqlCommand` перед отправкой запроса принимает соответствующие параметры, значения которых передаются из свойств текущего объекта `Employee`.

Ниже приведены результаты запуска программы с параметрами 2 "Ivanov Ivan Ivanovich" 1995-05-23 Male. Консоль отображает сообщения о готовности к работе БД и количество добавленных объектов. Добавленную строку можно наблюдать в SSMS.

КОНСОЛЬ



SSMS



Задача 3

Вывод всех строк с уникальным значением ФИО + дата рождения, теоретически предполагает, что значение пола в таких строках может быть разным. Из задания не понятно, что делать если есть строки с одинаковым сочетанием ФИО + дата рождения, но разные по полу. Т.к. в выводе должен присутствовать пол сотрудника, применение ключевого слова **DISTINCT** в данном случае недостаточно.

Такую задачу можно решить с помощью оконной функции **ROW_NUMBER()**, которая выбирает партии с одинаковым сочетанием ФИО + дата рождения и выбирает первую строку для каждой такой партии. В результате в выходные данные попадет пол того сотрудника, который находится первым в партии.

Полученный набор данных также сортируется по ФИО.

Асинхронный метод `EmployeesService.ShowDistinctEmployeeListAsync()` содержит код, который отправляет в БД запрос

```
SELECT sub.Fullname, sub.Birthdate, sub.Gender FROM  
(SELECT Fullname, Birthdate, Gender,  
ROW_NUMBER() OVER (PARTITION BY Fullname, Birthdate ORDER BY Fullname) AS row_num  
FROM Employees) AS sub
```

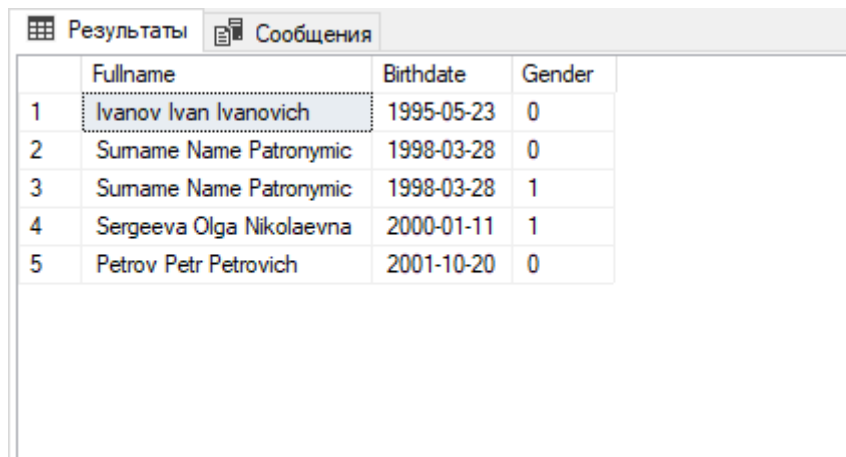
```
WHERE row_num = 1 ORDER BY sub.Fullname;
```

Полученные данные выводятся построчно в консоль в виде таблицы с помощью объекта класса SqlDataReader.

В классе модели Employee переопределен метод ToString() для вывода свойств Fullname, Birthdate, Gender и возраста сотрудника. Возраст вычисляется и возвращается приватным методом Employee.GetAge().

Перед запуском программы в режиме 3 в БД предварительно были добавлены строки, в которых есть два объекта с одинаковым сочетанием ФИО + дата рождения, но разного пола.

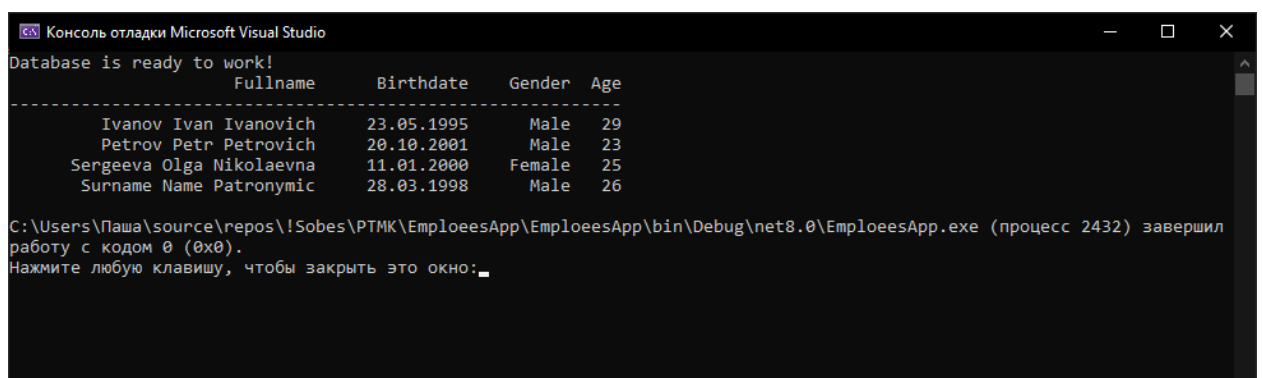
SSMS



	Fullname	Birthdate	Gender
1	Ivanov Ivan Ivanovich	1995-05-23	0
2	Surname Name Patronymic	1998-03-28	0
3	Surname Name Patronymic	1998-03-28	1
4	Sergeeva Olga Nikolaevna	2000-01-11	1
5	Petrov Petr Petrovich	2001-10-20	0

Ниже приведены результаты запуска программы в режиме 3. Сочетания значений Fullname + Birthdate уникальны, дополнительно приводятся пол и возраст.

консоль



```
Консоль отладки Microsoft Visual Studio
Database is ready to work!
Fullname      Birthdate    Gender  Age
-----
Ivanov Ivan Ivanovich  23.05.1995   Male   29
Petrov Petr Petrovich  20.10.2001   Male   23
Sergeeva Olga Nikolaevna  11.01.2000  Female  25
Surname Name Patronymic  28.03.1998   Male   26

C:\Users\Паша\source\repos\!Sobes\PTMK\EmployeesApp\EmployeesApp\bin\Debug\net8.0\EmployeesApp.exe (процесс 2432) завершил
работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно: _
```

Задача 4

Для заполнения БД набором строк в соответствии с заданными правилами реализован метод EmployeesService.FillDbRandomAsync(int, bool), принимающая параметры: int size – количество добавляемых строк и bool isMaleStartsWithF – указывающий, должны ли добавляемые объекты обладать мужским полом и фамилией, начинающейся с 'F'.

При формировании данных для каждого объекта Employee применяются соответствующие методы EmployeesService.GetRandomEmployee() и EmployeesService.GetRandomMaleStartsWithF() использующие объект класса Random.

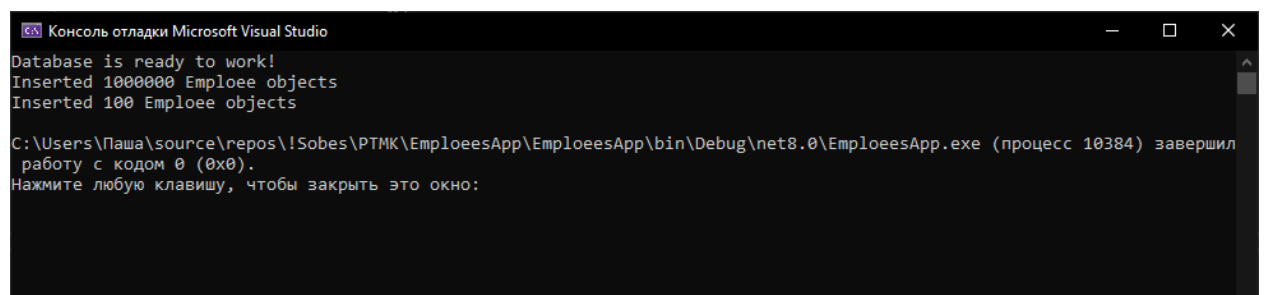
Сформированный список объектов Employee передается в статический метод Employee.SaveRangeToDb(SqlConnection, List<Employee>) вместе с заранее созданным объектом SqlConnection. В этом методе для отправки списка Employee в БД используется объект класса SqlDataAdapter.

Т.к. характеристики моего компьютера не очень высоки, при попытке добавить в БД сразу 1_000_000 объектов Employee программа завершала выполнение, не дождав ответа от БД. Поэтому я все-таки добавил в программный код разбиение всего массива отправляемых данных на порции по 10_000 строк.

Процессы заполнения БД 1_000_000 и 100 строк (с фамилиями на 'F') выполняются последовательно при запуске программы с параметром 4. Каждый процесс заканчивается выводом на консоль количества добавленных объектов.

Ниже приведены результаты запуска программы в режиме 4. В консоль выводятся сообщения о добавлении в БД сначала 1_000_000, а затем ещё 100 объектов.

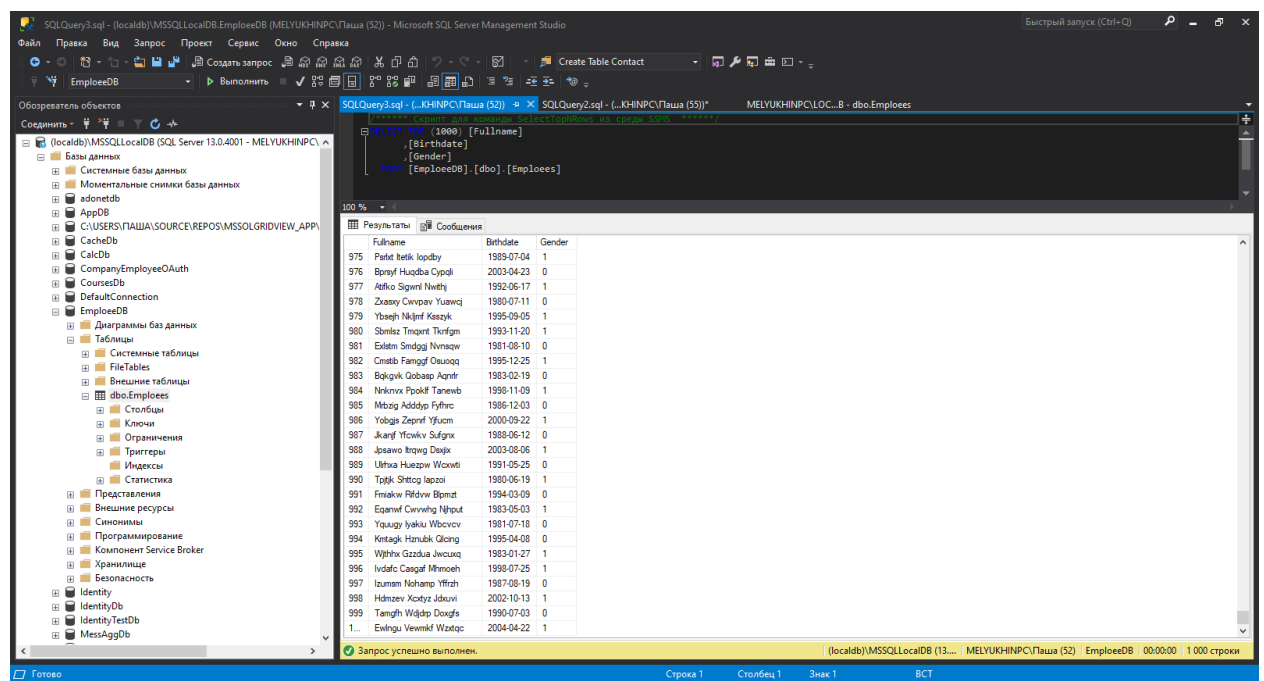
КОНСОЛЬ



```
Консоль отладки Microsoft Visual Studio
Database is ready to work!
Inserted 1000000 Employee objects
Inserted 100 Employee objects

C:\Users\Паша\source\repos\!Sobes\PTMK\EmployeesApp\EmployeesApp\bin\Debug\net8.0\EmployeesApp.exe (процесс 10384) завершил
работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно:
```

SSMS



SQL Query 3.sql - (localdb)\MSSQLLocalDB.EmployeeDB (MELYUKHINPC\Паша (52)) - Microsoft SQL Server Management Studio

Обозреватель объектов

Соединить

SQL Query 3.sql - (localdb)\MSSQLLocalDB (SQL Server 13.0.4001 - MELYUKHINPC\Паша (52))

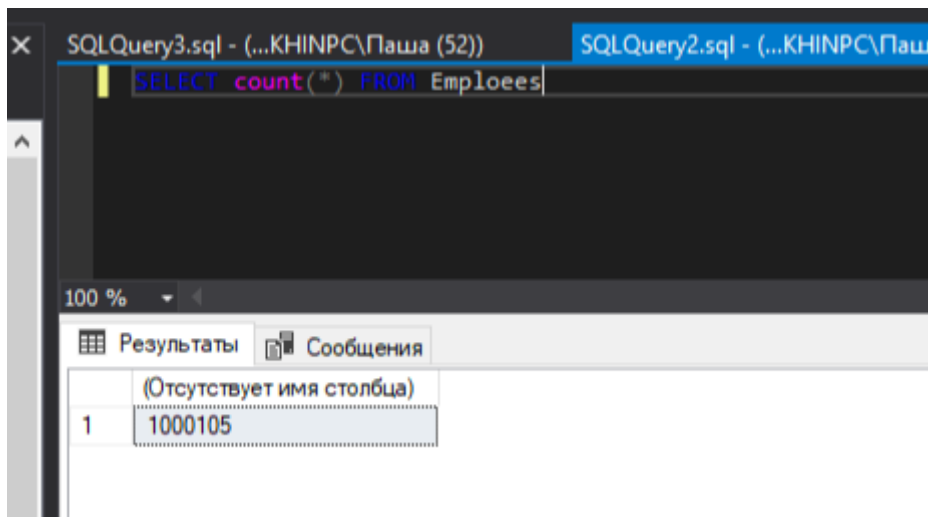
SQL Query 2.sql - (localdb)\MSSQLLocalDB (SQL Server 13.0.4001 - MELYUKHINPC\Паша (52))

MELYUKHINPC\LOC...B - dbo.Employees

SELECT TOP (1000) [Fullname]
[Birthdate]
[Gender]
FROM [EmployeeDB].[dbo].[Employees]

	Fullname	Birthdate	Gender
975	Perket Ietisk Iopdoby	1989-07-04	1
976	Bpnyf Huodba Cypoli	2003-04-23	0
977	Athko Sigwnt Nwatty	1992-06-17	1
978	Zuany Cuvpav Ylawnj	1980-07-11	0
979	Yonafj Wajnt Kasejk	1995-08-05	1
980	Sholez Tmawt Tawfjm	1993-11-20	1
981	Exlntk Smdgaj Nvnwaz	1981-08-10	0
982	Cmnlto Fangaf Oluooq	1995-12-25	1
983	Bokgvik Gobaap Agnrf	1983-02-19	0
984	Nekmxv Ppokf Tanewb	1998-11-09	1
985	Mbzag Addyop Fyfwc	1986-12-03	0
986	Yobag Zepnf Yfucm	2000-09-22	1
987	Jkarf Yfwkv Sufgix	1988-06-12	0
988	Jpsawo Itqmg Dwaix	2003-08-06	1
989	Ufhwa Huezrw Wxwtd	1991-05-25	0
990	Tpjk Shtog Iapoz	1990-06-19	1
991	Fmkwv Pfdvzv Bpmzt	1994-03-09	0
992	Egwnf Cuvwng Yhpwt	1983-05-03	1
993	Yauagy Iyaku Wbocv	1981-07-18	0
994	Ketagk Hznubk Olong	1995-04-08	0
995	Wjfhv Gzduka Jwcuq	1983-01-27	1
996	Ivdafc Caagaf Mhmoeh	1998-07-25	1
997	Iuzumv Nohamp Yffzh	1987-08-19	0
998	Hdnzerv Xotyz Jdeuvi	2002-10-13	1
999	Tamghv Wqdpw Dwaifs	1990-07-03	0
1...	Ewlngu Vewmkf Wzqtac	2004-04-22	1

Запрос успешно выполнен. (localdb)\MSSQLLocalDB (13.... MELYUKHINPC\Паша (52) EmployeeDB 00:00:00 1 000 строки

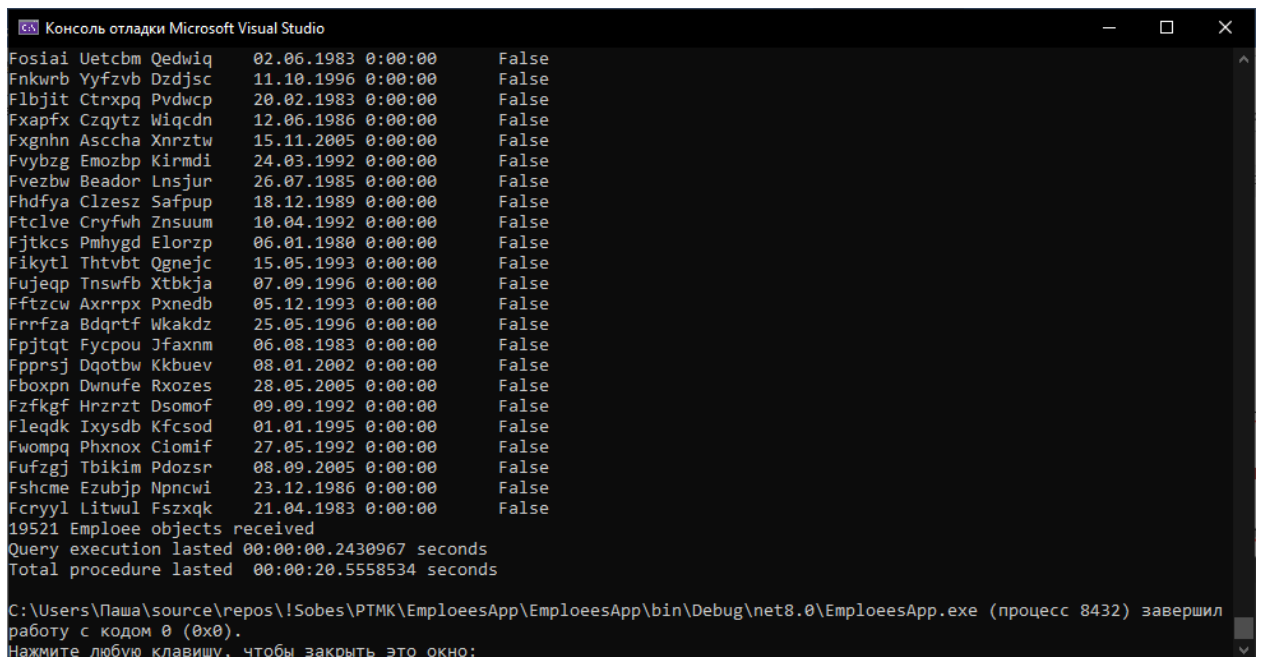


Задача 5

Для выборки из таблицы по заданному критерию (пол мужской, фамилия начинается с 'F') реализован метод `EmployeesService.ShowListOfMaleStartsWithF()`, который отправляет в БД запрос

```
SELECT Fullname, Birthdate, Gender FROM Employees WHERE Fullname LIKE 'F%' AND Gender = 0;
```

Сначала метод был реализован с помощью объекта класса `SqlDataAdapter`. В результате его запуска работа с базой данных заняла 0,243 секунды, а в целом работа приложения вместе с выводом данных на консоль 20,555 секунды.

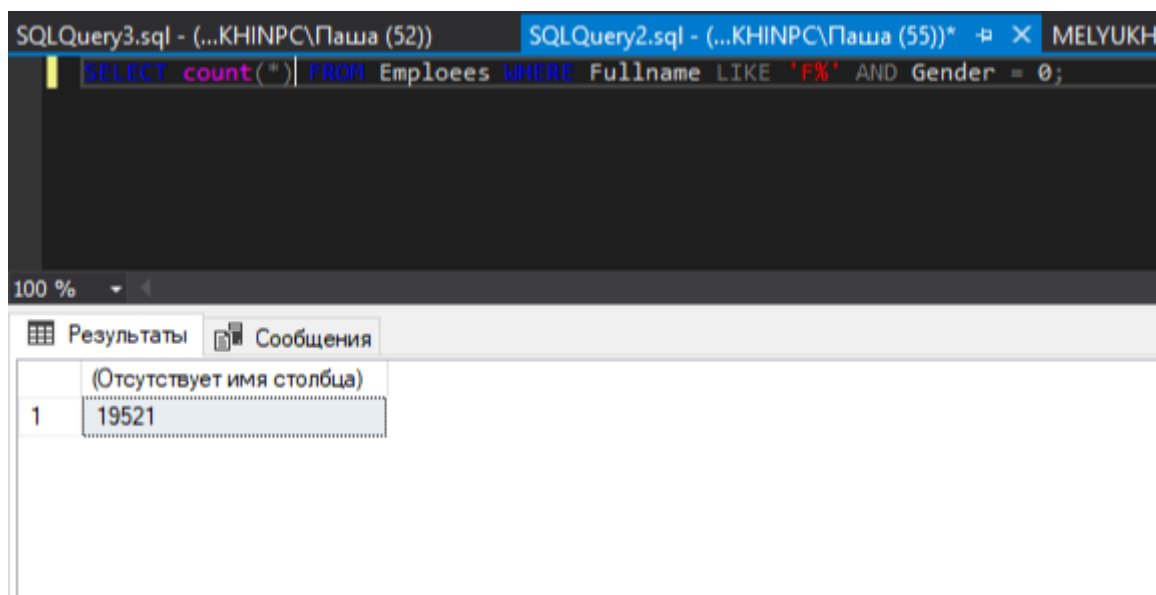


Затем метод был переписан с применением объекта класса SqlDataReader. При его запуске работа с базой данных заняла 0,045 секунды, а в целом – 17,491 секунды.

```
Консоль отладки Microsoft Visual Studio
Fosiai Uetcbm Qedwiq 02.06.1983 Male 41
Fnkurb Yyfvzb Dzdjsc 11.10.1996 Male 28
Flbjit Ctrxpq Pvdwcp 20.02.1983 Male 42
Fxpafx Czqytz Wlqcdn 12.06.1986 Male 38
Fxgnhn Asccha Xnrztw 15.11.2005 Male 19
Fvybzg Emozbp Kirmdi 24.03.1992 Male 32
Fvezbw Beador Lnsjur 26.07.1985 Male 39
Fhdfya Clzesz Safpup 18.12.1989 Male 35
Ftclve Cryfwh Znsuum 10.04.1992 Male 32
Fjtkcs Pmhygd Elorzp 06.01.1980 Male 45
Fikytl Thvtbt Qgnejc 15.05.1993 Male 31
Fujeqp Tnsafb Xtbkja 07.09.1996 Male 28
Fftzcv Axrrpx Pxnedb 05.12.1993 Male 31
Frrfza Bdqrtf Wkakdz 25.05.1996 Male 28
Fpjttt Fycpou Jfaxnm 06.08.1983 Male 41
Fpprsj Dqotbw Kkbuev 08.01.2002 Male 23
Fboxpn Dwnufe Rxozes 28.05.2005 Male 19
Fzfkgy Hrzrzt Dsomof 09.09.1992 Male 32
Fleqdk Ixysdb Kfcsod 01.01.1995 Male 30
Fwompq Phxnno Ciomif 27.05.1992 Male 32
Fufzgj Tvikim Pdozsr 08.09.2005 Male 19
Fshcme Ezubjp Npncwi 23.12.1986 Male 38
Fcryyl Litwul Fszxqk 21.04.1983 Male 41
19521 Employee objects received
Query execution lasted 00:00:00.0450490 seconds
Total procedure lasted 00:00:17.4910670 seconds

C:\Users\Паша\source\repos\!Sobes\PTMK\EmployeesApp\EmployeesApp\bin\Debug\net8.0\EmployeesApp.exe (процесс 9748) завершил
работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно: _
```

В SSMS можно убедиться, что БД действительно содержит 19521 объект, подходящий под условие (Fullname LIKE 'F%' AND Gender = 0).



Задача 6

Для оптимизации работы с задачей 5 к таблице Employee был добавлен уникальный кластеризованный индекс на основе полей Fullname и Gender.

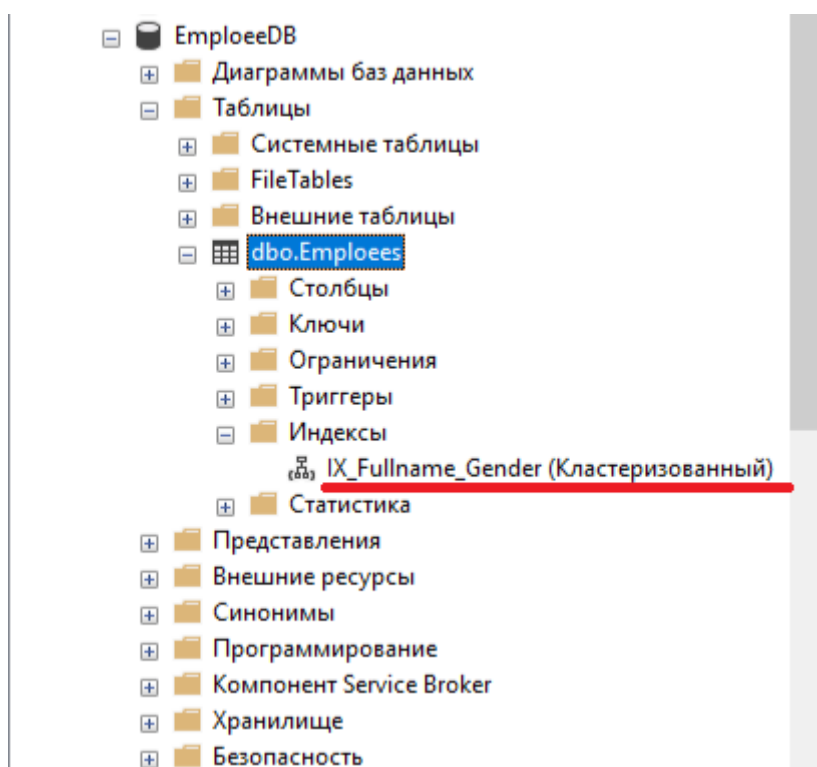
При создании индекса в БД создается древовидная структура данных на основе значений одного или нескольких столбцов целевой таблицы. Поиск строк в таблице по этим столбцам, благодаря такой структуре, улучшается до $O(\log n)$. Если индекс некластеризованный, то в его «листьях» хранятся ссылки на соответствующие строки таблицы, а если он кластеризованный, то сами строки. Т.е. кластеризованный индекс, по сути, и есть сама таблица, упорядоченная в виде такого дерева. Поэтому для одной таблицы не может существовать более одного кластеризованного индекса.

Если бы в данной таблице уже существовал бы какой-нибудь кластеризованный индекс, например, на основе первичного ключа, то его предварительно нужно было бы удалить. Затем первичный ключ можно вернуть, но тогда на его основе уже будет создан некластеризованный индекс (хранящий ссылки, а не сами строки).

Индекс может обладать также свойством уникальности. В этом случае каждый набор значений целевых столбцов будет для БД уникальным, что может еще больше ускорить процесс поиска в таблицах с большим количеством строк. При этом, в таблице может существовать несколько строк с одинаковым набором значений в целевых столбцах, просто БД помечает их специальными «метками».

Для создания уникального кластеризованного индекса в БД был передан запрос `CREATE UNIQUE CLUSTERED INDEX IX_Fullname_Gender ON Employees(Fullname, Gender);`

Отправка этого запроса была реализована в методе `EmployeesService.OptimizeDbAsync()`. После его запуска в SSMS можно наблюдать созданный индекс.



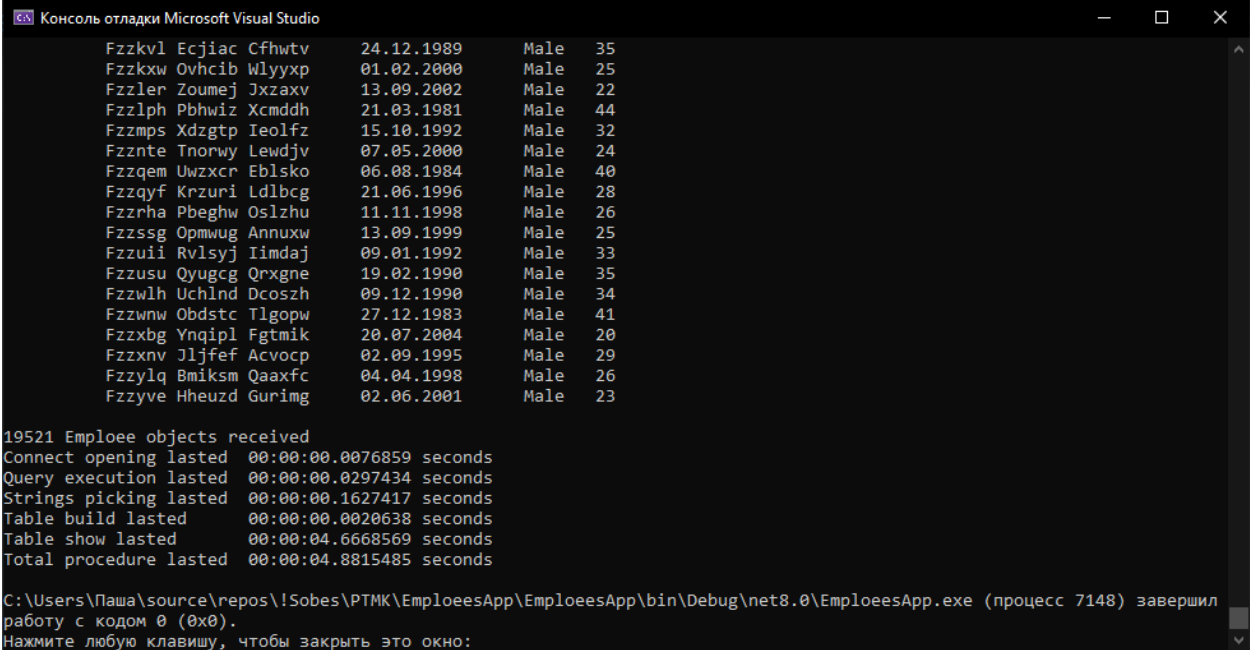
Были опробованы другие варианты индекса, например, покрывающий индекс по всем 3-м полям целевого запроса, но это не привело к заметному сокращению времени выполнения.

Дополнительно оптимизирована работа со строками. Вместо последовательного создания и отправки на консоль каждой отдельной строки был применен объект класса `StringBuilder`. В результате в консоль выводится уже вся таблица целиком.

Объекты класса `String` являются иммутабельными. Поэтому при объединении нескольких строк в одну происходит создание нового объекта `String`. Если такое объединение происходит слишком часто, это негативным образом сказывается на скорости выполнения приложений.

Класс `StringBuilder`, являющийся примером паттерна «Строитель», добавляет все переданные ему объекты в буфер. И только после вызова метода `ToString()` он формирует строку из объектов, хранящихся в буфере. При работе с большими строковыми объектами такой алгоритм может заметно ускорить работу приложения, если есть достаточно памяти для организации буфера.

Ниже приведены результаты запуска программы в режиме 5 после её оптимизации. Выполнение запроса заняло 0,029 секунды, а в целом работа программы – 4,881 секунды.



```
Консоль отладки Microsoft Visual Studio

Fzzkvl Ecjiac Cfhwtv 24.12.1989 Male 35
Fzzkxw Ovhcib Wlyyxp 01.02.2000 Male 25
Fzzler Zoumej Jxzaxv 13.09.2002 Male 22
Fzzlph PbhwiZ Xcmddh 21.03.1981 Male 44
Fzzmps Xdzgtp Ieolfz 15.10.1992 Male 32
Fzznte Tnorwy Lewdjv 07.05.2000 Male 24
Fzzqem UwzxcR EbIsko 06.08.1984 Male 40
Fzzqyf Krzuri Ldlbcg 21.06.1996 Male 28
Fzzrha Pbeghw Oslzhu 11.11.1998 Male 26
Fzzssg Opmwug Annuxw 13.09.1999 Male 25
Fzzuii Rvlsyj IimdaJ 09.01.1992 Male 33
Fzzusu Qyugcg Qrxgne 19.02.1990 Male 35
Fzzwlh Uchlnd Dcoszh 09.12.1990 Male 34
Fzzwnw Obdstc Tlgopw 27.12.1983 Male 41
Fzzxbg Ynqipl Fgtmik 20.07.2004 Male 20
Fzzxnv Jljfef Acvocp 02.09.1995 Male 29
Fzzylq Bmiksm Qaaxfc 04.04.1998 Male 26
Fzzzye Hheuzd Gurimg 02.06.2001 Male 23

19521 Employee objects received
Connect opening lasted 00:00:00.0076859 seconds
Query execution lasted 00:00:00.0297434 seconds
Strings picking lasted 00:00:00.1627417 seconds
Table build lasted 00:00:00.0020638 seconds
Table show lasted 00:00:04.6668569 seconds
Total procedure lasted 00:00:04.8815485 seconds

C:\Users\Паша\source\repos\!Sobes\PTMK\EmployeesApp\EmployeesApp\bin\Debug\net8.0\EmployeesApp.exe (процесс 7148) завершил
работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно:
```

Из результатов видно, что основные затраты времени уходят на вывод уже сформированной таблицы в консоль.

В принципе для оптимизации приложений можно использовать асинхронный запуск параллельных задач. Однако в данном случае, когда «тормозит» сам вывод в консоль, это не принесет заметных результатов.