Міністерство освіти і науки України

Відокремлений структурний підрозділ
«Ковельський промислово-економічний фаховий коледж
Луцького національного технічного університету»

СУБД Microsoft SQL Server

МЕТОДИЧНИЙ ПОСІБНИК

для здобувачів освітньо-кваліфікаційного рівня
фаховий молодший бакалавр
спеціальності 122 «Комп'ютерні науки»
денної форми навчання

Ковель 2023

УДК 004.01

Затверджено до видання методичною радою ВСП «КПЕФК ЛНТУ»
протокол №_____ від « ___» _____ 2023 року
Голова методичної ради ВСП «КПЕФК ЛНТУ» _____ Ігор ІЛЮШИК

Розглянуто і схвалено на засіданні циклової комісії 122 «Компютерні науки», протокол № 5 від « 25» січня 2023 року.
Голова циклової методичної комісії _____ Олександр ПРИСАДА

Електронна копія друкованого видання передана до книжкового фонду бібліотеки ВСП «КПЕФК ЛНТУ»
Завідувачка бібліотеки_____Неля ТЕЛЮЧИК

Укладач:_____ Олександр ПРИСАДА, викладач ВСП «КПЕФК ЛНТУ»

Рецензент _____Степан БАБАРИКА, к.т.н., викладач ВСП «КПЕФК ЛНТУ»

Відповідальний за випуск: _____Леся ПРОКОПЧУК, методист ВСП «КПЕФК ЛНТУ»

Методичний посібник СУБД Microsoft SQL Server для здобувачів освітньо-кваліфікаційного рівня фаховий молодший бакалавр спеціальності 122 «Комп'ютерні науки» денної форми навчання /уклад. О.В. Присада – Ковель: КПЕК Луцького НТУ, 2023. – 64 с.

Відповідно до нормативної програми дисципліни «Організація баз даних та знань» приведено навчальний матеріал для вивчення проектування та створення баз даних з використанням структурованої мови запитів SQL. Відповідно до нормативної програми дисципліни «Організація баз даних та знань» викладено методичні вказівки та рекомендації щодо проектування та створення баз даних з використанням структурованої мови запитів SQL, а також для виконання практичних робіт. Засвоєнню теоретичного матеріалу сприятиме розроблений термінологічний словник ключових понять, навчальні завдання та завдання для перевірки знань. Подана широка тематика лабораторних робіт для самостійного виконання

Методичний посібник може бути використаний здобувачами освіти при вивченні освітніх компонент "Організація баз даних і баз знань", "Офісне програмне забезпечення", "Диплоному проєктуванню", а також для написання кваліфікаційної роботи.

Зміст

Лекція 1: Види інформаційних систем. Основні поняття інформаційних систем. Історія Microsoft SQL Server +2008

Мета:

1. Вивчити основні види інформаційних систем.
2. Визначити переваги і недоліки технології Файл-Сервер і технології Клієнт-Сервер
3. Основні поняття інформаційних систем.

Види інформаційних систем

Інформаційні системи – це комплекс засобів, призначених для зберігання, упорядкування та аналізу великих обсягів інформації.

Інформаційні системи бувають електронними і неелектронними.

До неелектронних інформаційних систем відносяться:

- Каталог в бібліотеці.
- Реєстратура в лікарні.
- Бібліотека.

До електронних інформаційних систем відносяться:

- База даних відділу кадрів підприємства.
- Записник у мобільному телефоні.
- Мережа Інтернет.

Існує три види інформаційних систем:

1. База даних – система для зберігання великих обсягів структурованої інформації (інформації, яка вводиться за шаблоном) певного типу. До баз даних відносяться наступні інформаційні системи:

- каталог бібліотеки;
- реєстратура лікарні;
- записна книжка мобільного телефону;
- база даних відділу кадрів.

2. База знань – система для зберігання великого обсягу неструктурованої інформації різних типів. До баз знань відносяться наступні інформаційні системи:

- бібліотека;
- мережа Інтернет.

3. Інформаційно-аналітична система - система, призначена як для зберігання, так і для аналізу збереженої інформації

- Excel.
- STATISTICA.
- SPSS.
- 1С бухгалтерія.

- 1С підприємство.

Всі електронні інформаційні системи діляться на два класи за способом зберігання інформації:

1. Не мережеві інформаційні системи, що працюють за технологією файл-сервер. Дані системи працюють на окремо взятому комп'ютері, без використання комп'ютерної мережі (Excel, STATISTICA, SPSS).
2. Мережеві інформаційні системи, що працюють за технологією клієнт-сервер. Дані системи працюють на комп'ютері, підключеному до комп'ютерної мережі (Інтернет).

Основна відмінність технології клієнт-сервер від технології файл-сервер полягає в способі зберігання інформації, суть технології файл-сервер полягає в наступному - інтерфейс інформаційної системи і дані, з якими вона працює зберігається на одному комп'ютері (локально).

***Зауваження*:**

1. Клієнтами мережі є комп'ютери користувачів, підключені до мережі. Клієнти отримують доступ до сервера через мережу. Іноді клієнти мережі називають клієнтськими комп'ютерами.
2. Сервер мережі – комп'ютер, який управляє мережею. Всі ресурси сервера доступні клієнтам мережі, тобто будь-які зміни даних на сервері відразу видно всім клієнтам мережі.

В інформаційних системах, побудованих за технологією клієнт-сервер, інформація зберігається на сервері, а інтерфейс інформаційної системи зберігається на клієнтських комп'ютерах, через нього користувачі інформаційної системи отримують доступ до даних.

Переваги та недоліки технології Файл-Сервер:	Переваги та недоліки технології Клієнт-Сервер:
❖ Простота розробки. ❖ Незалежність комп'ютеравідо мережі. ❖ Високий захист від несанкціонованого доступу. ❖ Не оперативне оновленя даних на декількох комп'ютерах. ❖ Висока вартість комп'ютерів в такій системі. ❖ Складність зміни структури даних.	❖ Проста синхронізація даних. ❖ Низька вартість апаратного забезпечення (потужним повинен бути тільки сервер). ❖ Оперативна заміна структури даних. ❖ Низький захист від несанкціонованого доступу. ❖ Залежність від комп'ютерної мережі. ❖ Висока вартість.

Основні поняття інформаційних систем

Будь-яка інформаційна система або база даних (з погляду їх створення) у мовах програмування складаються з трьох компонентів:

1. **Файл даних** – файл, що знаходиться на локальному комп'ютері або на сервері, який містить усередині себе структуру даних. До структури даних відносяться таблиці, запити і фільтри, а також збережені процедури, призначені для користувача функції, діаграми і тригери.
2. **Об'єкт зв'язку** – об'єкт мови програмування, що здійснює зв'язок між файлом даних і інтерфейсом інформаційної системи.
3. **Інтерфейс інформаційної системи** – комплекс засобів, що здійснює взаємодію системи з кінцевими користувачами. Він може знаходитися як на клієнтському комп'ютері, так і на сервері.

Розробка ІС за технологією клієнт-сервер складається з декількох етапів:

1. На сервер в комп'ютерній мережі встановлюються серверна СУБД (наприклад, Microsoft SQL Server, MySQL, Oracle), встановлюється серверна частина СУБД. Якщо реалізується web-інтерфейс, то на сервер ставиться програма web-сервер (наприклад, Apache).
2. Якщо реалізуються клієнтські програми, то на всі клієнтські частини мережі ставиться клієнтська частина (даний крок не обов'язковий і виконується тільки в тому випадку, якщо користувачі інформаційної системи мають можливість керувати сервером).
3. Налаштовується серверна частина СУБД, клієнтські частини СУБД і web-сервер.
4. Визначається структура даних (зв'язки між таблицями і типи даних полів), також визначаються первинні і вторинні таблиці в запитах.
5. На сервері створюються таблиці і запити, що виконуються на стороні сервера. Перед створенням запитів, таблиці заповнюються початковими даними. Також створюються збережені процедури, призначені для користувача функції, діаграми і тригери.
6. У разі використання клієнтського застосування, за допомогою мови програмування створюються об'єкти зв'язку, вони підключаються до таблиць, запитів і збережених процедур. Також на них створюються запити і збережені процедури, що виконуються на стороні сервера.
7. Створюються форми.
8. Створюються звіти.
9. Система заповнюється реальними даними.

***Зауваження*:** Під час створення і заповнення таблиць інформаційної системи необхідно слідувати 3 правилам:

1. У таблицях не повинно бути повторюваних груп записів. Це досягається введенням індексних полів, тобто сортуванням записів.

2. У таблиці не повинно бути полів з однаковими іменами. Це досягається розбивкою однієї таблиці на декілька, з подальшим зв'язуванням їх запитом.
3. Не повинно бути правил під час заповнення таблиць, це досягається хаотичністю заповнення таблиць бази даних.

Інформаційна система, яка задовольняє цим умовам, називається нормалізованою інформаційною системою або базою даних.

Історія Microsoft SQL Server 2008, його версії і системні вимоги

Засновником серії SQL Server і його основою є мова запитів SQL. Дана мова була створена компанією IBM на початку 1970 р. минулого століття. Спочатку вона називалася SEQUEL (Structured English Query Language) В основу мови SQL, використовуваного в SQL Server, лягл різновид мови T-SQL (Transact - SQL).

На початку 80-х років фірма IBM і її підрядники Microsoft і Sybase створюють першу версію мережевий СУБД, яка називалася SQL Server версія 1.0, для операційної системи IBM OS / 2. Після цього під цю операційну систему було випущено ще 3 версії SQL Server. У середині 80-х років компанії Microsoft і Sybase відокремлюються від фірми IBM, і Microsoft починає роботу над своєю операційною системою Windows, і разом з компанією Sybase починає розвиток SQL Server.

У середині 90-х років (зокрема в 1995 р) Microsoft створила операційну систему Windows NT і разом з компанією Sybase випускає першу версію SQL Server для Windows версії 4.1.

Після цього компанія Sybase розриває свої відносини з Microsoft і Microsoft створює Microsoft SQL Server 6.0. Дана версія була призначена для роботи в операційній системі Windows NT, 95 і 98. У 1999 р. виходить версія Microsoft SQL Server 7.0, яка стала однією з найпопулярніших серверних СУБД у світі. У 2000 р. виходить 8-а версія Micrsoft SQL Server 2000. У 2005 році виходить нова версія сервера, заснована на новій технології .NET, а в 2008 році виходить її поліпшена версія Microsoft SQL Server +2008.

Лекція 2: ***Основні компоненти Microsoft SQL Server 2008. Створення файла даних. Управління базами даних за допомогою команд мови T-SQL***

Мета:

1. Вивчити систему основних компонентів Microsoft SQL Server +2008.
2. Зрозуміти процес створення файла даних.
3. Освоїти управління базами даних за допомогою команд мови T-SQL.

Основні компоненти Microsoft SQL Server +2008

Всі компоненти Microsoft SQL Server +2008 запускаються з меню Пуск\Програми\Microsoft SQL Server 2008. У Microsoft SQL Server +2008 входять наступні компоненти:

1. Deployment Wizard – майстер з висновку інформації зберігається на сервері.
2. SQL Server Installation Center – центр установки SQL Server 2008.
3. Reporting Services Configuration Manager – менеджер служби налаштування звітів.
4. SQL Server Configuration Manager – менеджер настройки сервера.
5. SQL Server Error and Usage Reporting – служба протоколювання роботи сервера і служба звітів про помилки.
6. Microsoft Samples Overview – посилання на сайт корпорації Microsoft, де можна переглянути приклади роботи з сервером.
7. SQL Server Books Online – повна довідкова система по Microsoft SQL Server 2008. Вона містить довідки, як з програмування, так і з адміністрування сервера.
8. SQL Server Tutorials – підручники по роботі з сервером.
9. Data Profile Viewer – перегляд профілів по роботі з даними.
10. Execute Package Utility – інструменти зі стиснення даних.
11. Database Engine Tuning Advisor – майстер установки ядра бази даних.
12. SQL Server Profiler – настройка профілів по роботі з даними.
13. Import and Export Data – імпорт і експорт даних.
14. SQL Server Business Intelligence Development Studio – інтегроване середовище розробки Business Intelligence Development Studio.
15. SQL Server Management Studio – графічна оболонка для управління сервером і розробки баз даних.

Створення файла даних

Нову БД можна створити, використовуючи стандартні команди мови T-SQL. Для створення нової БД необхідно зробити активну БД «Master». Це можна зробити або вибором її зі списку БД на панелі інструментів, або набором команди USE Master на вкладці нового запиту.

***Зауваження*:** Всі команди мови T-SQL набираються на вкладці нового запиту (SQLQuery). Для того, щоб створити новий запит на панелі інструментів, необхідно натиснути кнопку:

Для виконання команд мови T-SQL на панелі інструментів необхідно натиснути кнопку:

або на вкладці нового запиту набрати команду **GO**.

***Зауваження*:** У Microsoft SQL Server БД складається з двох частин:

- Файл даних – файл, що має розширення mdf і де знаходяться всі таблиці і запити.
- Файл журналу транзакцій – файл, що має розширення ldf, містить журнал, де фіксуються всі дії з БД. Даний файл призначений для відновлення БД у разі її виходу з ладу.

Для створення нового файла даних використовується команда **CREATE DATABASE**, яка має наступний синтаксис:

```
CREATE DATABASE <Имя БД>
ON (Name=<Логическое имя>,
FileName=<Имя файла>
[Size=<Нач.размер>,]
[Maxsize=<Макс.размер>,]
[FileGrowth=<Шаг>])
[LOG ON
(Name=<Логическое имя>,
FileName=<Имя файла>
[Size=<Нач.размер>,]
[Maxsize=<Макс.размер >,]
[FileGrowth=<Шаг>])
```

Тут:

- **Имя БД** – ім'я створюваної БД.
- **Логическое имя** – визначає логічне ім'я файла даних БД, по якому відбувається звернення до файла даних.
- **Имя файла** – визначає повний шлях до файла даних.
- **Нач.размер** – початковий розмір файла даних в Мб.
- **Макс.размер** – максимальний розмір файла даних в Мб.
- **Шаг** – крок збільшення файла даних, або в Мб, або в%.

Параметри в розділі **LOG ON** аналогічні параметрам в розділі **CREATE DATABASE**. Однак вони визначають параметри журналу транзакцій.

Приклад: Створити БД «Студент», розташовану у файлі «D: \ Students.mdf», який має початковий розмір файла даних 1 мб. Максимальний розмір файла даних 100 мб. і крок збільшення файла даних рівний 1 мб. Файл журналу транзакцій даної БД має ім'я

«Студент» і розташований у файлі «D: \ Students.ldf». Даний файл має початковий розмір рівний 1 мб. Максимальний розмір рівний 100 мб. і крок збільшення рівний 1 мб.

```
CREATE DATABASE Students
ON (Name = Students,
FileName = 'D: \ Students.mdf',
Size = 1Mb,
Maxsize = 100Mb,
FileGrowth = 1Mb)LOG ON
(Name = StudentsLog,
FileName = 'D: \ Students.ldf',
Size = 1Mb,
Maxsize = 100Mb,
FileGrowth = 1Mb)
```

Управління базами даних за допомогою команд мови T-SQL

У мові запитів T-SQL з БД можливі наступні дії:

1. Відображення відомостей про БД: **EXEC sp_helpdb <Имя БД>.**
2. Зміна параметрів БД: **EXEC sp_dboption <Имя БД>, <Параметр>, <Значение>.**
3. Додавання нових файлів, видалення файлів і перейменування файлів, що входять до БД:

```
ALTER DATABASE <Имя БД>
ADD FILE (<Параметри>) |
REMOVE FILE <Логическое имя файла> |
MODIFY FILE (<Параметри>)
```

де, розділ **ADD FILE** – додає файл, **REMOVE FILE** – видаляє, а розділ **MODIFY FILE** – змінює параметри файла.

4. Стиснення всій БД: **DBCC SHRINKDATABASE <Имя БД>.**
5. Стиснення конкретного файла БД: **DBCC SHRINKFILE < Логическое имя файла >.**
6. Перейменування БД: **EXEC SP_RENAMEDB <Имя БД>, <Новое имя БД>.**
7. Видалення БД: **DROP DATABASE <Имя БД>.**

***Зауваження*:** Вищеперераховані команди використовують наступні параметри:

- **<Имя БД>** – ім'я БД з якою проводиться дія.
- **<Параметр>** – змінний параметр.
- **<Значение>** – нове значення змінюваного параметра.
- **<Параметри>** – параметри файла БД, аналогічні установкам в команді **CREATE DATABASE.**
- **< Логическое имя файла >** – логічне ім'я файла, що входить в БД.
- **<Новое имя БД>** – нове ім'я БД.

Тест 1 до Лекції 1, Лекції 2

№ З/п	*Питання*	*Варіанти відповідей*
1.	**Вкажіть інформаційні системи, що не відносяться до електронних:**	1. Реєстратура в лікарні. 2. Комп'ютерна база даних відділу кадрів підприємства. 3. Паперовий каталог у бібліотеці.
2.	**До якого виду інформаційних систем відноситься реєстратура лікарні?**	1. База знань. 2. Інформаційно-аналітична система. 3. База даних.
3.	**Вкажіть переваги електронних інформаційних систем, що працюють за технологією файл-сервер:**	1. Простота розробки. 2. Незалежність від комп'ютера мережі. 3. Низька вартість апаратного забезпечення. 4. Високий рівень захисту від несанкціонованого доступу. 5. Оперативну зміну структури даних. 6. Проста синхронізація даних.
4.	**Вкажіть недоліки електронних інформаційних систем, що працюють за технологією клієнт-сервер:**	1. Складність зміни структури даних. 2. Низький рівень захисту від несанкціонованого доступу. 3. Не оперативне оновлення даних на декількох комп'ютерах. 4. Висока вартість інформаційної системи. 5. Висока вартість комп'ютерів для роботи в такій системі. 6. Залежність від комп'ютерної мережі.
5.	**Інтерфейс інформаційної системи – це ...**	1. Комплекс засобів, що здійснює взаємодію системи з кінцевими користувачами. 2. Об'єкт мови програмування, що здійснює зв'язок між файлом даних та інтерфейсом інформаційної системи. 3. Об'єкт, що знаходиться на локальному комп'ютері або на сервері, який містить усередині себе структуру даних.

6.	**Які умови необхідно виконати, щоб інформаційна система називалася нормалізованою?**	1. В таблицях не повинно бути повторюваних груп записів. 2. Не повинно бути правил під час заповнення таблиць. 3. В таблицях не повинно бути полів з різними іменами. 4. В таблицях не повинно бути полів з однаковими іменами.
7.	**Вкажіть команду мови запитів T-SQL, яка видаляє файл з бази даних:**	1. Remove file. 2. Create database. 3. Dbcc shrinkfile. 4. Add file.
8.	**Вкажіть команду мови запитів T-SQL, яка стискає базу даних:**	1. Modify file. 2. Dbcc shrinkdatabase. 3. Add file.
9.	**Вкажіть команду мови запитів T-SQL, яка видаляє базу даних:**	1. Exec cp_renamedb. 2. Drop database. 3. Exec sp_dboption.
10.	**Яке розширення має файл журналу транзакцій бази даних Microsoft SQL Server?**	1. ldf. 2. Exe. 3. Mdf.
11.	**До якого виду інформаційних систем відноситься програма Excel?**	1. База знань. 2. База даних. 3. Інформаційно-аналітична система.
12.	**Клієнтами мережі є ...**	1. Комп'ютери, які керують мережею. 2. Комп'ютери, які підключені до мережі бази даних. 3. Комп'ютери користувачів, що працюють локально.

Бланк відповідей

1	2	3	4	5	6	7	8	9	10	11	12

Самостійна робота 2: Створення файла даних і журналу транзакцій

Мета: навчитися створювати файли даних і журнал транзакцій.

Створення будь-якої БД починається зі створення файла даних. Розглянемо цей процес в «Microsoft SQL Server 2008» на прикладі створення простої БД з обліку успішності студентів.

Для початку необхідно запустити середовище розробки «SQL Server Management Studio». Для цього в меню «Пуск» вибираємо пункт **«Программи\Microsoft SQL Server +2008\SQL Server Management Studio»** (рис. 2.1).

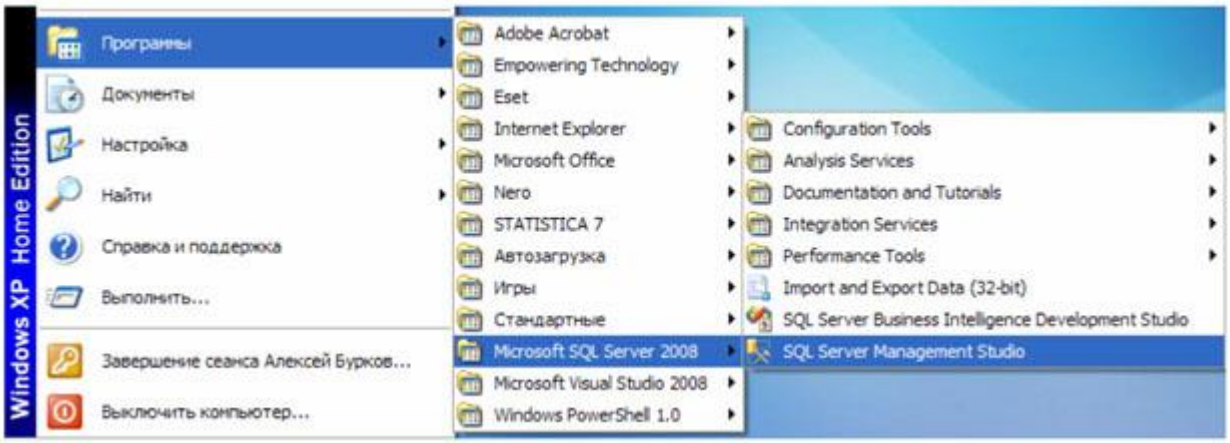


Рис. 2.1

Після запуску середовища розробки з'являється вікно підключення до сервера **«Соединение с сервером»** (рис. 2.2).

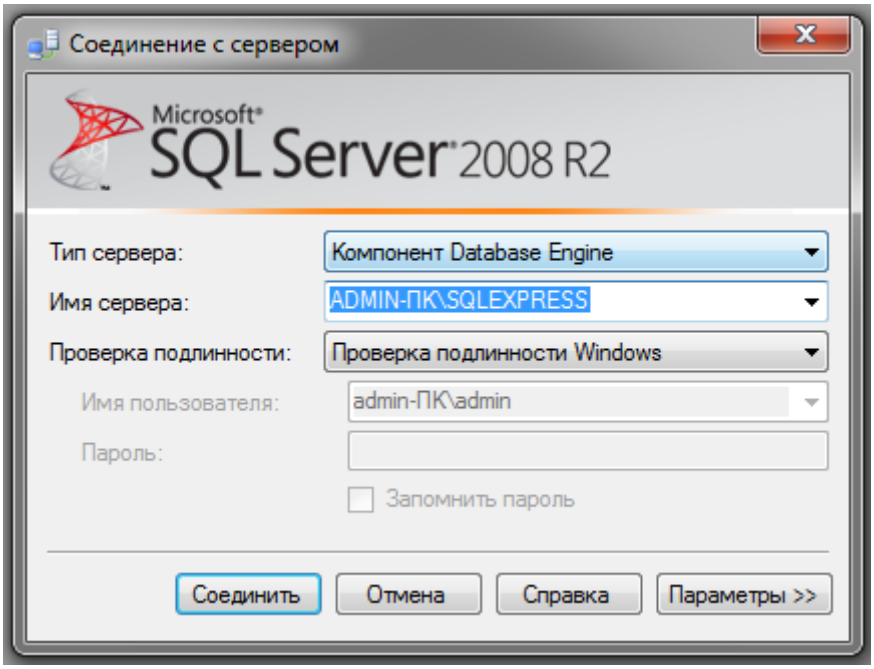


Рис. 2.2

У цьому вікні необхідно натиснути кнопку **«Соединить».**

***Зауваження*:** Якщо при установці «Microsoft SQL Server 2008» був заданий логін і пароль підключення до сервера, то перед натисканням кнопки **«Соединить»,** у випадаючому списку **«Проверка подлинности»** потрібно вибрати «**Проверка подлинности SQL Server»**, а потім необхідно ввести задані при установці логін і пароль.

Після натискання кнопки **«Соединить»** з'явиться вікно середовища розробки **«SQL Server Management Studio»** (рис. 2.3).

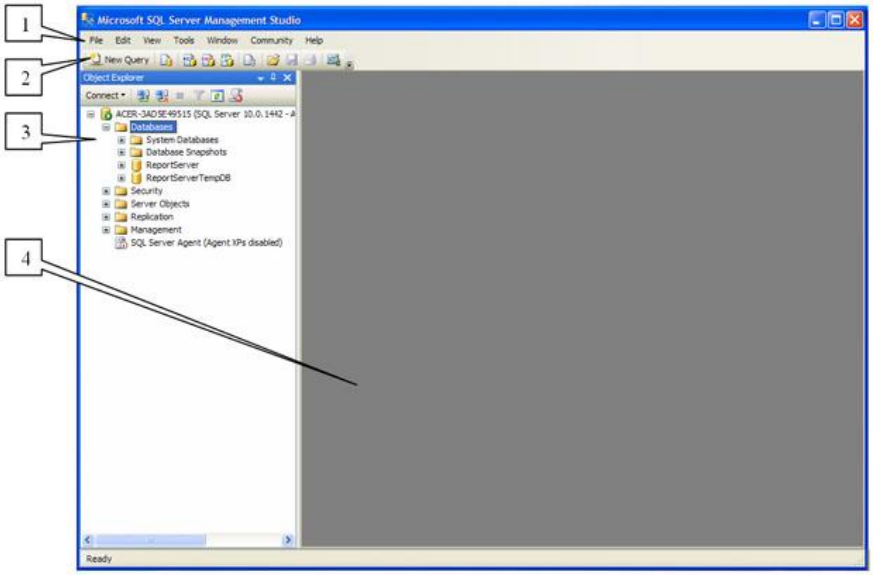


Рис. 2.3

Дане вікно має наступну структуру:

1. **Віконне меню** – містить повний набір команд для управління сервером і виконання різних операцій.
2. **Панель інструментів** – містить кнопки для виконання найбільш часто вироблених операцій. Зовнішній вигляд даної панелі залежить від виконуваної операції.
3. **Панель «Object Explorer»** – оглядач об'єктів. Оглядач об'єктів – це панель з деревовидною структурою, що відображає всі об'єкти сервера, а також дозволяє проводити різні операції, як з самим сервером, так і з БД. Оглядач об'єктів є основним інструментом для розробки БД.
4. **Робоча область.** У робочій області проводяться всі дії з БД, а також відображається її вміст.

***Зауваження*:** У браузері об'єктів самі об'єкти знаходяться в папках. Щоб відкрити папку необхідно клацнути по знаку «+» зліва від зображення папки.

Тепер перейдемо безпосередньо до створення файла даних. Для цього в оглядачі об'єктів клацніть ПКМ на папці **«Базы данных»**(рис. 2.3) і в меню виберіть пункт **«Создать базу данных»**. З'явиться вікно налаштувань параметрів файла даних нової БД «**Создание базы данных»** (рис. 2.4) . У лівій частині вікна налаштувань є список **«Выбор страницы»**. Цей список дозволяє перемикатися між групами налаштувань.

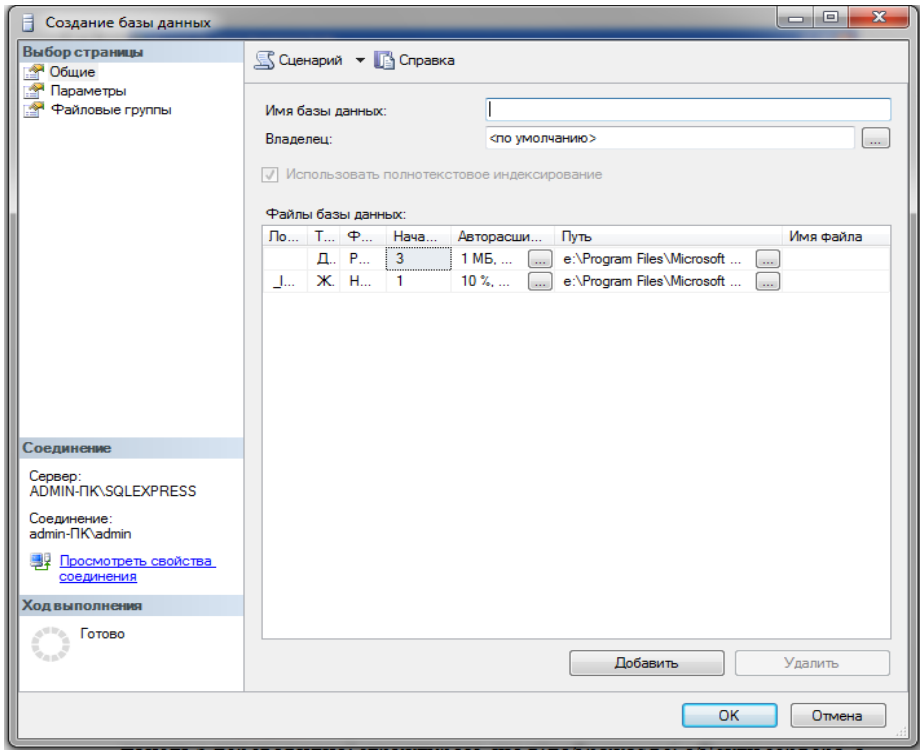


Рис. 2.4

Для початку налаштуємо основні налаштування «**Общие»**. Для вибору основних налаштувань потрібно просто клацнути мишею по пункту «**Общие»** у списку **«Выбор страницы»**. У правій частині вікна «**Создать базу данных»** з'являться основні налаштування (рис. 2.4).

Розглянемо їх більш докладно. У верхній частині вікна розташовано два параметри: **«Имя базы данных»** і **«Владелец»**. Задайте параметр **«Имя базы данных»** рівним «**Студент»**. Параметр **«Владелец»** залиште без змін.

Під вищенаведеними параметрами у вигляді таблиці розташовуються налаштування файла даних і журналу транзакцій. Таблиця має такі стовпці:

- **Логическое имя** – логічне ім'я файла даних і журналу транзакцій. За цим іменем відбуватиметься звернення до вищенаведених файлів в БД. Можна помітити, що файл даних має те ж ім'я, що і БД, а ім'я файла журналу транзакцій складено з імені БД і суфікса «_log».
- **Тип файла** – тип файлу. Цей параметр показує, чи є файл файлом даних або журналом транзакцій.
- **Файловая группа** – група файлів, показує до якої групи файлів відноситься файл. Групи файлів налаштовуються в групі налаштувань «Filegroups».
- **Начальный размер (MB)** – початковий розмір файлу даних і журналу транзакцій в мегабайтах.
- **Авторасширение** – авторозширення розміру файла. Як тільки файл заповнюється інформацією, його розмір автоматично збільшується на величину, зазначену в параметрі **«Авторасширение»**. Збільшення можна задавати як в мегабайтах, так і у

відсотках. Тут же можна задати максимальний розмір файлів. Для зміни цього параметра треба натиснути кнопку «...». У нашому випадку (рис. 2.4) розмір файлів не обмежений. Файл даних збільшується на 1 мегабайт, а файл журналу транзакцій на 10%.

- **Путь** – шлях до папки, де зберігаються файли. Для зміни цього параметра також треба натиснути кнопку «...».
- **Имя файла** – імена файлів. За замовчуванням імена файлів аналогічні логічним іменах. Однак файл даних має розширення «mdf», а файл журналу транзакцій – розширення «ldf».

***Зауваження*:** Для додавання нових файлів даних або журналів транзакцій використовується кнопка **«Добавить»,** а для видалення кнопка «**Удалить».**

Тепер перейдемо до інших другорядним налаштувань файла даних. Для доступу до цих налаштувань необхідно натиснути мишею по пункту «**Параметры»** у списку «**Выбор страницы»**. З'явиться наступне вікно (рис. 2.5).

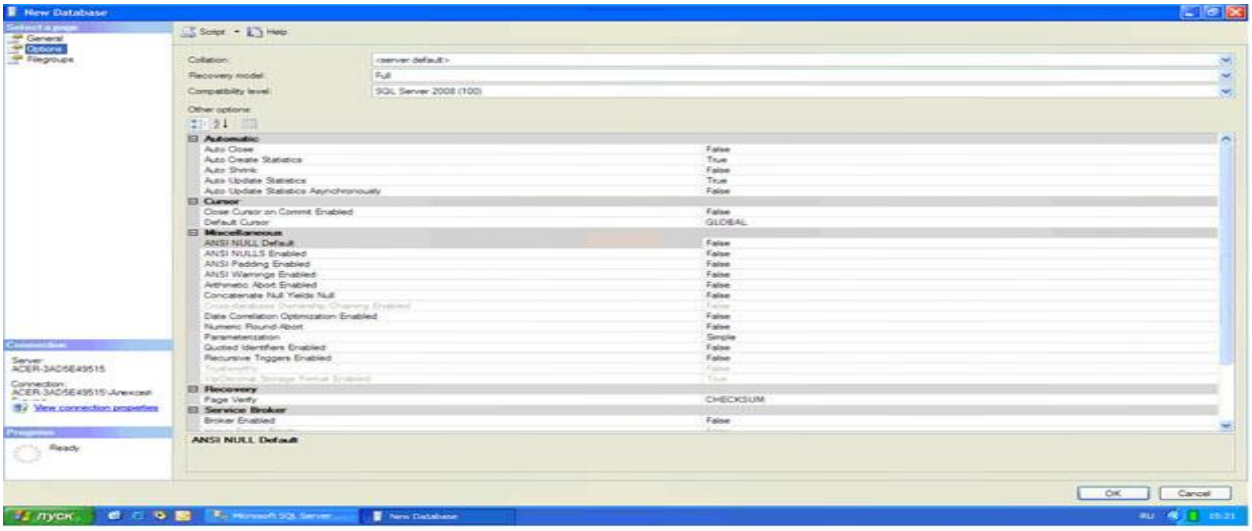


Рис. 2.5

У правій частині вікна ми бачимо наступні налаштування:

- **Параметры сортировки** – цей параметр відповідає за обробку текстових рядків, їх порівняння, текстовий пошук і т.д. Рекомендується залишити його як **«<по умолчанию сервера>«**. При цьому даний параметр буде дорівнювати значенню, заданому на вкладці **«Параметры сортировки»**, під час установки сервера.
- **Модель восстановления** – модель відновлення. Даний параметр відповідає за інформацію, призначену для відновлення БД, що зберігається у файлі транзакцій. Чим повніше модель відновлення, тим більше вірогідність відновлення даних при збої системи або помилках користувачів, але і більше розмір файла журналу транзакцій. При наявності місця на диску, рекомендується залишити цей параметр в значенні **«Простая»**.

- **Уровень совместимости** – рівень сумісності, визначає сумісність файла даних з більш ранніми версіями сервера. Якщо планується перенесення даних на іншу, більш ранню версію сервера, то її необхідно вказати в цьому параметрі.
- **Другие параметры** – другорядні параметри. Дані параметри є необов'язковими для зміни.

У нашому випадку всі параметри в розділі **«Параметры»**, рекомендується залишити, як на рис. 2.5.

Нарешті розглянемо останню групу налаштувань **«Файловые группы»**. Дана група налаштувань відповідає за групи файлів. Для її відображення у списку **«Создание базы данных»** необхідно натиснути ЛКМ по пункту **«Файловые группы»**. Відобразяться налаштування груп файлів (рис. 2.6).

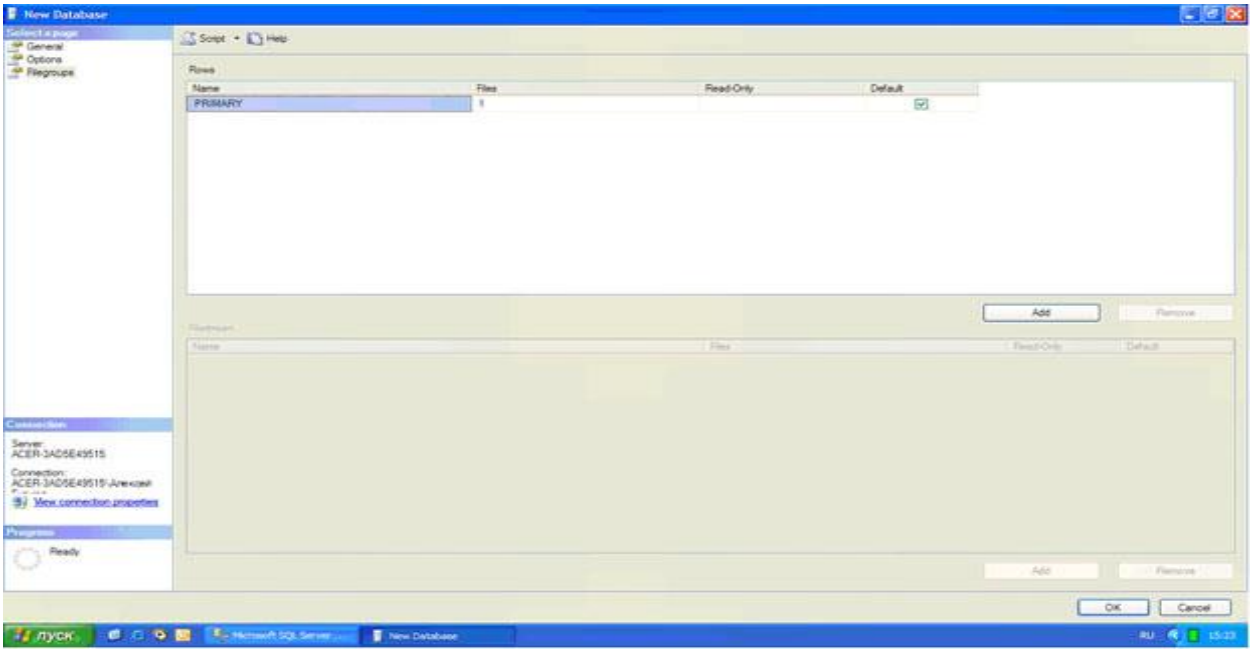


Рис. 2.6

Групи файлів представлені у таблиці **«Строки»** в правій частині вікна (рис. 2.6). Дана таблиця має такі стовпці:

- **Имя** – ім'я групи файлів.
- **Файлы** – кількість файлів входять у групу.
- **Только для чтения** – файли в групі будуть тільки для читання. Тобто, їх можна тільки переглядати, але не можна змінювати.
- **По умолчанию** – група за замовчуванням. Всі нові файли даних будуть входити в цю групу.

***Зауваження*:** Як і у випадку з файлами даних, для додавання нових груп використовується кнопка **«Добавить»**, а для видалення кнопка **«Удалить»**.

У розглянутій БД немає необхідності додавати нові групи файлів. Тому залишимо групу налаштувань **«Файловые группы»** без змін.

На цьому ми закінчуємо настройку властивостей наших файлів. Для прийняття всіх налаштувань і створення файла даних і журналу транзакцій нашої БД у вікні **«Создание базы данных»** натиснемо кнопку **«Ok»**.

Відбудеться повернення у вікно середовища розробки **«SQL Server Management Studio»**. На панелі оглядача об'єктів у програмі **«Базы данных»** з'явиться нова БД **«Студент»** (рис. 2.7).

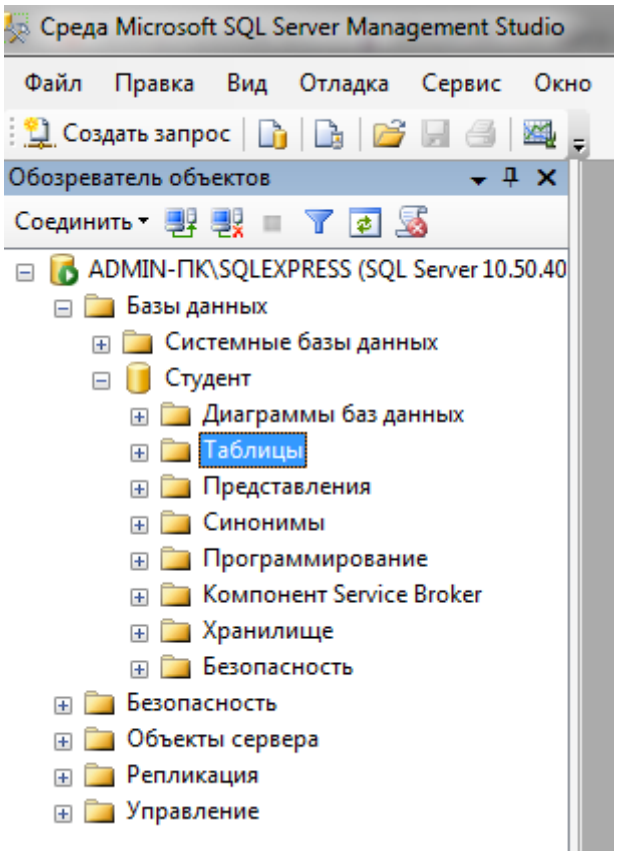


Рис. 2.7

***Зауваження*:** Для перейменування БД необхідно в оглядачі об'єктів клацнути по ній ПКМ і в меню вибрати пункт **«Переименовать»**. Для видалення в цей же меню вибираємо пункт **«Удалить»**, для оновлення – пункт **«Обновить»**, а для зміни властивостей описаних вище – пункт **«Свойства».**

Лекція 3: Таблиці. Типи даних і властивості полів. Створення і заповнення таблиць.

Мета:

1. Вивчити таблиці і типи даних полів.
2. Освоїти створення таблиць та основні операції з таблицями.

Таблиці. Типи даних полів

Вся інформація в базі даних зберігається в таблицях. Таблиці це звичайні таблиці для зберігання даних. Таблиці складаються з записів.

Запис це рядок в таблиці. Вся інформація обробляється по записах. Кожен запис складається з полів. Поле це стовпець таблиці. Кожне поле має три характеристики:

1. Ім'я поля – використовується для звернення до поля.
2. Значення поля – визначає інформацію, збережену в поле.
3. Тип даних поля – визначає який вид інформації можна зберігати в поле.

У SQL сервер використовуються наступні типи даних:

- **Бітові типи даних** які містять послідовності нулів та одиниць: Binary (n) і Varbinary (n), де n довжина. Довжина вмісту полів типу Binary завжди дорівнює n, різниця заповнюється пробілами. Varbinary розмір поля дорівнює n або меншого.
- **Цілочисельні типи даних** – типи даних для зберігання цілих чисел (в дужках вказано діапазон значень типу даних, приблизно): Tinyint (0-255), Smallint (± 215), Int (± 231), Bigint (± 263).
- **Типи даних для зберігання дробових чисел**: Real сьомій знаків після коми, Float (m) може зберігати числа з m знаків, максимальне m = 38, Decimal (m, n) дробові числа з m знаків до коми і n після.
- **Спеціальні типи даних**: Bit – логічний тип данних.является заміною логічному типу Boolean в Visual Basic, Text – тип для зберігання великих обсягів тексту, одне поле може зберігати до 2 Гб тексту, Image – тип даних для зберігання до 2Гб малюнків, RowGUID - унікальний ідентифікатор рядки таблиці, SQL_Variant - аналогічний типу Variant в Visual Basic.
- **Типи даних дати і часу**: Datetime (1 січня 1753 - 31 грудня 9999). SmallDatetime (від 1.01.1900 до 06.06.2079).
- **Грошові типи даних для зберігання фінансової інформації**: Money (від -922337203685 477,5808 до +922337203685 477,5807), Smallmoney (від -214 748,3648 до 214 748,3647).
- **Автоматично оновлювані типи даних** – аналоги лічильників, але в даній ролі вони не використовуються: RowVersion – унікальний ідентифікатор рядка. TimeStamp – закодоване дата і час створення рядка.

Створення таблиць.

Для створення таблиць в SQL Server в першу чергу необхідно зробити активною ту БД, в якій створюється таблиця. Для цього в новому запиті можна набрати команду: **USE <Имя БД>,** або на панелі інструментів необхідно вибрати у випадаючому списку робочу БД. Після вибору БД можна створювати таблиці.

Таблиці створюються командою:

CREATE TABLE <Имя таблицы>(<Имя поля1> <Тип1> [IDENTITY NULL|NOTNULL],<Имя поля2> <Тип2>, …)

Тут:

- **< Имя таблицы >** – ім'я створюваної таблиці.
- **< Имя поля >** – імена полів таблиці.
- **<Тип>** – типи полів.
- **<IDENTITY NULL | NOT NULL>** – поле лічильник.

***Зауваження*:** Якщо ім'я поля містить пробіл, то воно береться в квадратні дужки.

Приклад: Створити таблицю «Студент», що містить поля: **Код студента** (первинне поле зв'язку, лічильник), **ФИО, Адрес, Код специальности** (вторинне поле зв'язку):

```
CREATE TABLE Студент
([Код студента] Bigint Identity,
ФИО Varchar(20),
Адрес Varchar(100),
[Код специальности] Bigint)
```

***Зауваження*:** Якщо необхідно створити обчислювальне поле, то в команді Create Table в обчислюваному полі замість типу даних потрібно вказати вираз.

Приклад: розрахувати середній бал студента за трьома його оцінками.

```
CREATE TABLE Оценки
(ФИО Varchar(20),
Оценка1 int,
Оценка2 int,
Оценка3 int,
[средний балл] = (Оценка1+ Оценка2+ Оценка3)/3
```

***Зауваження*:** Отримання інформації про таблиці здійснюється застосуванням команди: **EXEC SP_HELP <Имя таблицы>.** Видалення таблиці здійснюється командою: **DROP TABLE <Имя таблицы>.**

Заповнення таблиць

У SQL Server 2008 заповнення таблиць проводиться за допомогою наступної команди:

```
INSERT INTO <Имя таблицы> [(<Список полей>)]
VALUES (<Значения полей>)
```

де < Имя таблицы > – таблиця, куди вводимо дані, **(<Значения полей >**) – список полів, куди вводимо дані, якщо не вказуємо, то мається на увазі заповнення всіх полів, у списку полів поля вказуються через кому, (<Значення полів>) – значення полів через кому.

В якості значень можна вказати константу Default, тобто буде поставлено значення за замовчуванням, або можна підставити оператор **Select**. Тут він використовується як інструмент обчислення формул.

Приклад: Додавання запису має наступні значення полів ПІБ = Иванов, Адрес = Київ, Код специальности = 5 в таблицю «Студент».

```
INSERT INTO Студент (ФИО, Адрес, [Код специальности])
VALUES (‘Иванов А.А.’, Киев, 5)
```

Видалення окремих стовпців і окремих рядків з таблиці

З таблиці можна видалити всі стовпці або окремі записи. Це здійснюється командою:

```
DELETE FROM <Имя таблицы>
[WHERE <Условие>]
```

де **< Условие >** – умови, якими задовільняють видалені записи. Якщо умови не вказані, то видаляються всі рядки таблиці. Якщо умови вказані, то видаляються записи, поля яких відповідають умові.

Приклад: Видалити записи з таблиці «Студенти», у яких поле Адреса = Київ.

```
DELETE FROM Студенты
WHERE Адрес = ‘Киев’
```

Зміна даних в таблиці

Для цього використовується наступна команда:

```
UPDATE <Имя таблицы>
SET
<Имя поля1> = <Выражение1>,
[<Имя поля2> = <Выражение2>,]
…
[WHERE <Условие>]
```

Тут **< Имя поля1>, < Имя поля2>** – імена змінних полів, **< Выражение1>, <Выражение1>** – або конкретні значення, або **NULL**, або оператори **SELECT**. Тут **SELECT** застосовується як функція.

<Условие> – умова, якій повинні відповідати записи, поля яких змінюємо.

Приклад: У таблиці «Студенти» у студента Іванова А.А. поміняти адресу Київ на Йошкар-Ола, а код спеціальності замість 5 поставити 3.

```
UPDATE Студент
SET
Адрес = 'Йошкар-Ола',
[Код специальности] = 3
WHERE ФИО = 'Иванов А.А.'
```

***Зауваження*:** В якості вираження можна використовувати математичні формули.

Наприклад: **SET [Средний балл]= (Оценка1+ Оценка2+ Оценка3)/3)**обчислює поле **«Средний балл»** як середнє полів **«Оценка1», «Оценка2» і «Оценка3».** При цьому поля **«Оценка1», «Оценка2» і «Оценка3»** мають уже існувати і тип даних поля **«Средний балл»** повинен бути з плаваючою комою (наприклад, Real).

***Зауваження*:** Якщо необхідно з таблиці видалити всі записи, але зберегти її структуру, для цього використовують команду **TRUNCATE TABLE <Имя таблицы>** при цьому всі дані будуть видалені, але сама таблиця залишиться.

Самостійна робота 3: Створення та заповнення таблиць

Мета: навчитися створювати й заповнювати таблиці.

Перейдемо тепер до створення таблиць. Усі таблиці нашої БД знаходяться в підпапці «**Таблицы»** папки **«Студент»** у вікні оглядача об'єктів (рис. 3.1).

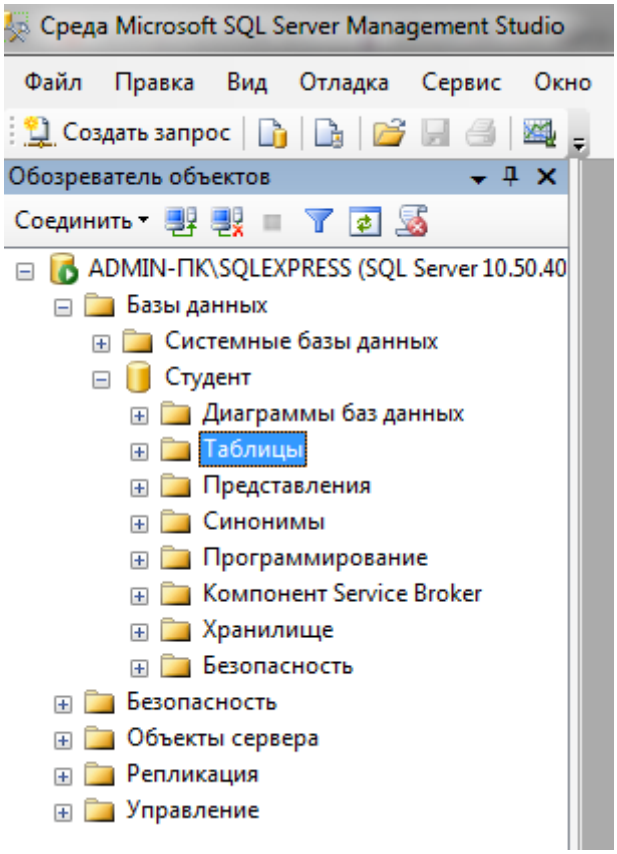


Рис. 3.1

Створюємо таблицю **«Спеціальності»**. Для цього клацніть ПКМ по папці **«Таблицы»** і в меню виберіть пункт «**Создать таблицу»**. З'явиться вікно створення нової таблиці (рис. 3.2).

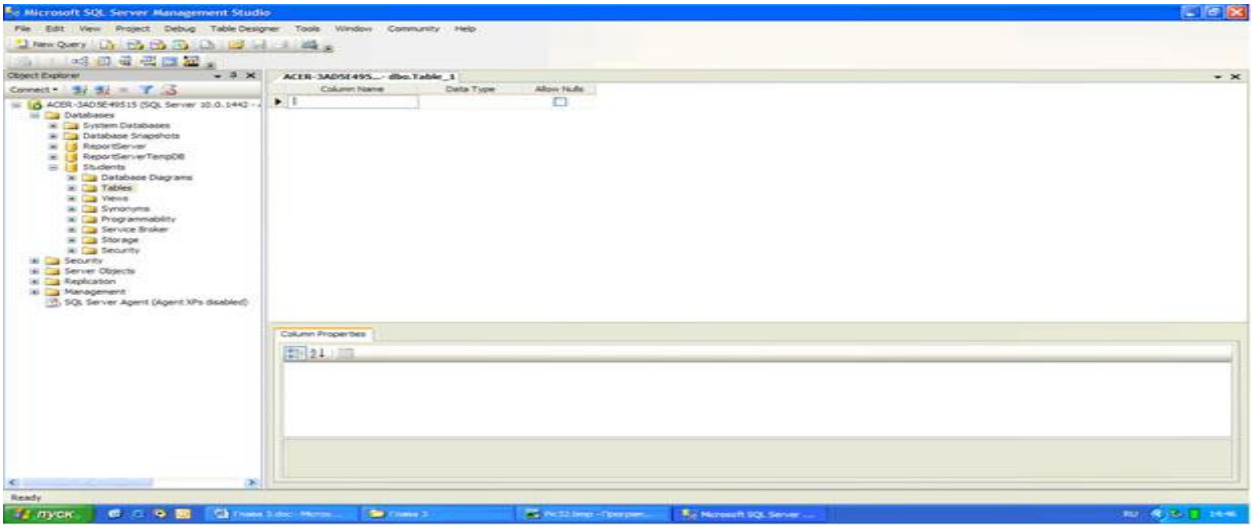


Рис. 3.2

У правій частині вікна розташована таблиця визначення полів нової таблиці. Дана таблиця має такі стовпці:

- **Имя столбца** – ім'я поля. Ім'я поля повинно завжди починатися з літери і не повинно містити різних спеціальних символів і знаків пунктуації. Якщо ім'я поля містить пробіли, то воно автоматично виділяється в квадратні дужки.
- **Тип данных** – тип даних поля.
- **Разрешить значение** – допуск значення **Пусто**. Якщо ця опція поля включена, то в разі незаповнення поля в нього буде автоматично підставлено значення **Пусто**. Тобто, поле необов'язкове для заповнення.

***Зауваження*:** Під таблицею визначення полів розташовується таблиця властивостей виділеного поля «**Свойства столбца»**. У даній таблиці налаштовуються властивості виділеного поля. Деякі з них будуть розглянуті нижче.

Перейдемо до створення полів і налаштування їх властивостей. У таблиці визначення полів задайте значення стовпців «**Имя столбца»**, «**Тип данных»** і **«Разрешить значение»**, як показано на малюнку нижче (рис. 3.3).

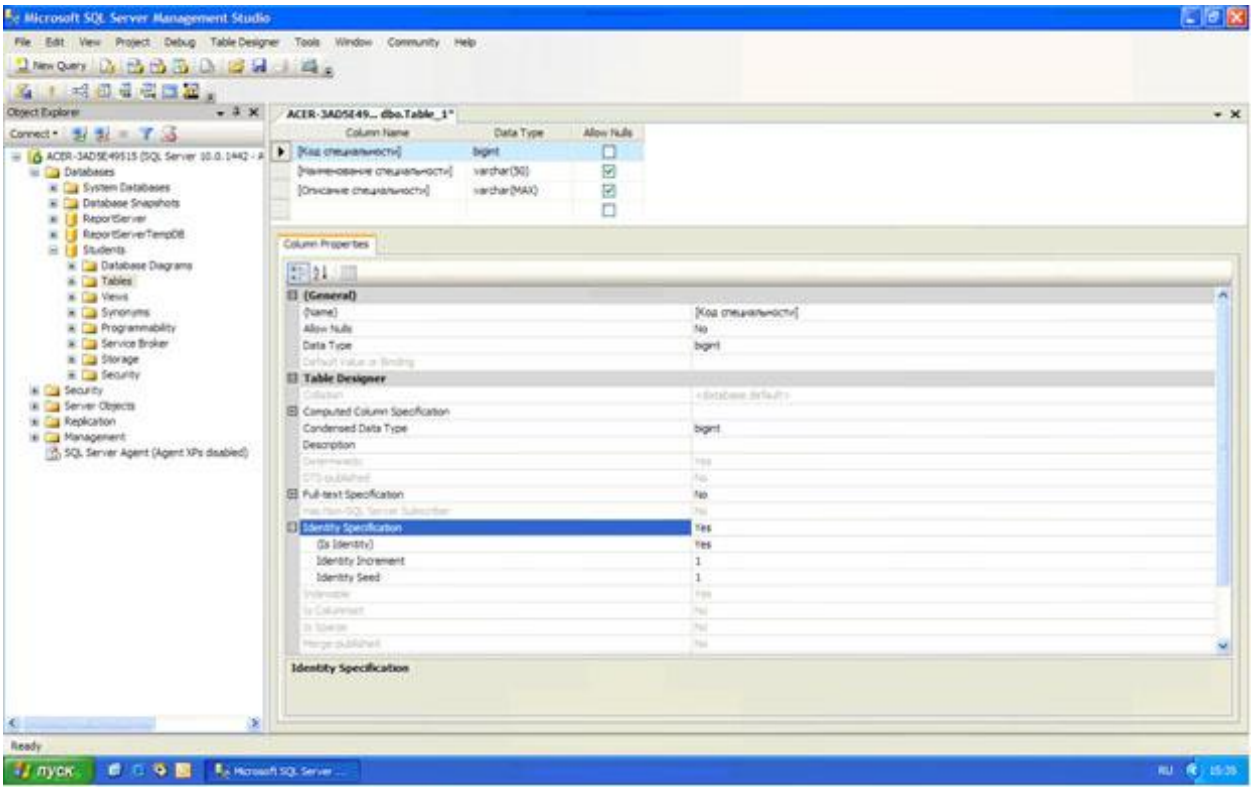


Рис. 3.3

Наша таблиця **«Специальность»** має три поля:

- **Код специальности** – числове поле для зв'язку з таблицею студенти.
- **Наименование специальности** – текстове поле, призначене для зберігання рядків, які мають довжину не більше 50 символів.
- **Описание специальности** – текстове поле, призначене для зберігання рядків, які мають необмежену довжину.

***Зауваження*:** Так як, поле **«Код специальности»** буде первинним полем зв'язку в запиті, що зв'язує таблиці **«Студенти»** і **«Специальность»**. То ми повинні зробити його числовим значенням. Тобто дане поле повинно автоматично заповнюватися числовими значеннями. Більше того, воно має бути ключовим.

Зробимо поле «**Код специальности»** лічильником. Для цього виділіть поле, просто клацнувши по ньому мишкою в таблиці визначення полів. У таблиці властивостей поля відобразяться властивості поля **«Код специальности»**. Розгорніть групу властивостей «**Спецификация идентификатора»**. Властивість **«(Идентификатор)»** встановіть в значення «**Да»**. Задайте властивості «**Начальное значение идентификатора»** і «**Шаг приращения идентификатора».** Ці настройки показують, що значення поля «**Код специальности»** у першому записі в таблиці буде рівним 1, у другому - 2, у третьому 3 і т.д.

Тепер зробимо поле «**Код специальности»** ключовим полем. Виділіть поле, а потім на панелі інструментів натисніть кнопку із зображенням ключа .

У таблиці визначення полів, поряд з полем «**Код специальности»** з'явитися зображення ключа, яке говорить про те, що поле ключове.

На цьому настройку таблиці «**Специальности»** можна вважати завершеною. Закрийте вікно створення нової таблиці, натиснувши кнопку закриття у верхньому правому куті вікна, над таблицею визначення полів. З'являється вікно з запитом про збереження таблиці (рис. 3.4).

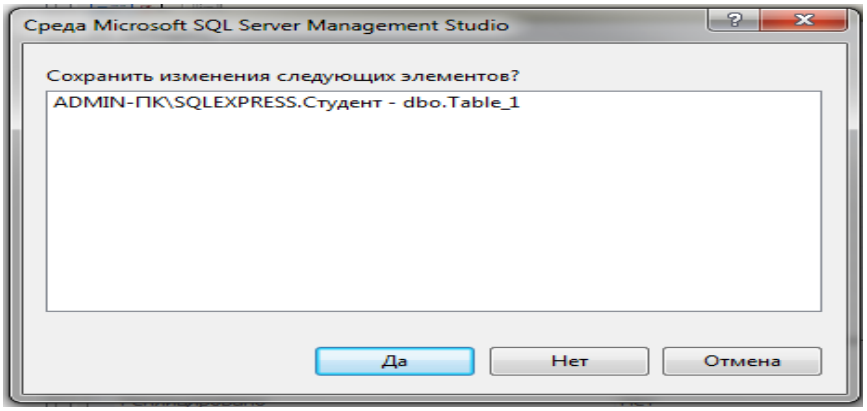


Рис. 3.4

У цьому вікні необхідно натиснути **«Да»**. З'явитися вікно **«Выбор имени»**, призначене для визначення імені нової таблиці (рис. 3.5).

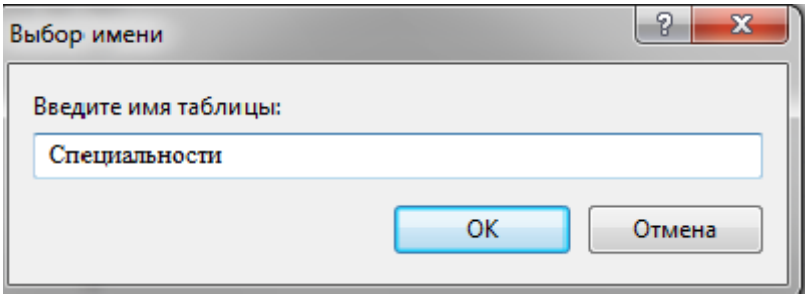


Рис. 3.5

У цьому вікні задайте ім'я нової таблиці як «**Специальности»** і натисніть кнопку **«Ok»**. Таблиця **«Специальности»** відобразиться в оглядачі об'єктів у програмі **«Таблицы» БД «Студент»**.

***Зауваження*:** У браузері об'єктів таблиця **«Специальности»** відображається як **«dbo.Специальности»**. Префікс **«dbo»** означає, що таблиця є об'єктом БД (Data Base Object). Надалі під час роботи з об'єктами БД префікс «**dbo»** можна опускати.

Тепер перейдемо до створення таблиці **«Предметы»**. Як і у випадку з таблицею «**Специальности»** клацніть ПКМ по папці **«Таблицы»** і в меню виберіть пункт **«Создать таблицу»**. Створіть поля представлені на малюнку нижче (рис. 3.6).

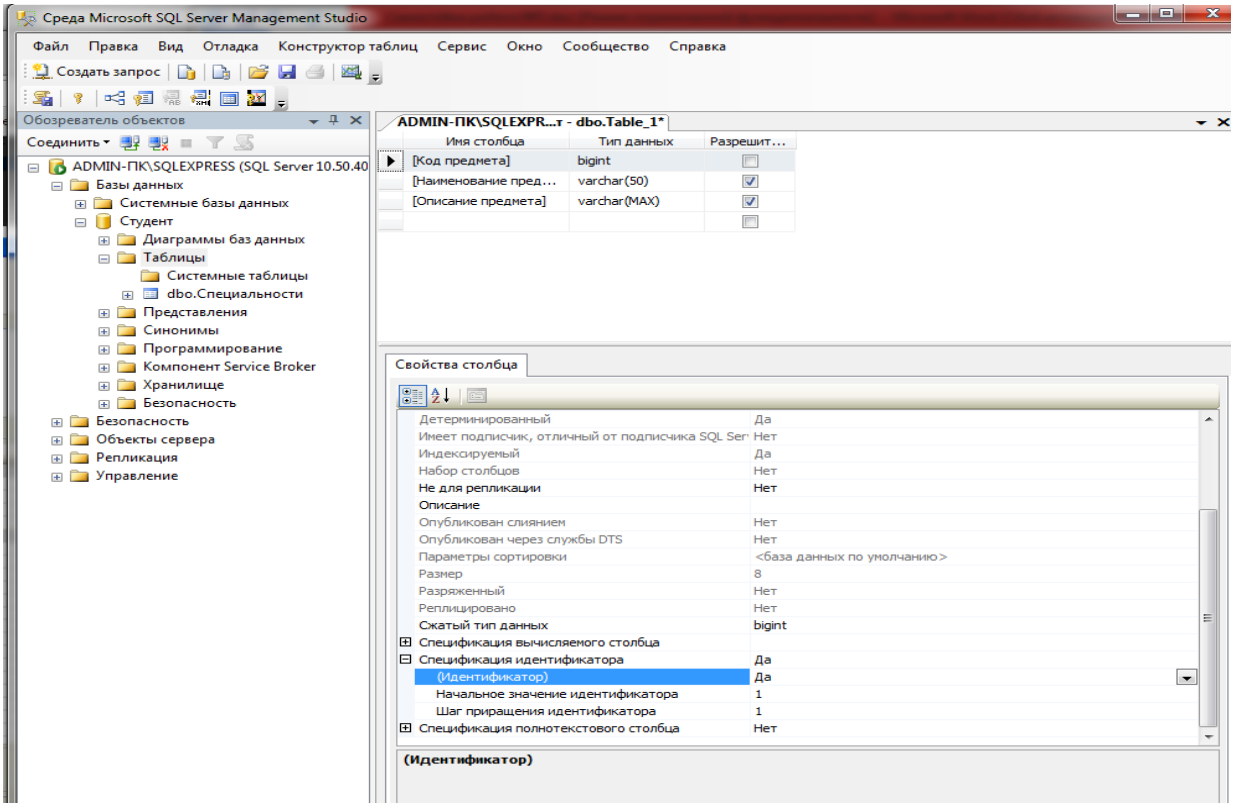


Рис. 3.6

Зробіть поле «**Код предмета»** числовим лічильником і ключовим полем, як це було зроблено в таблиці **«Специальности»**. Закрийте вікно створення нової таблиці. У вікні **«Выбор имени»** задайте ім'я **«Предметы»** (рис. 3.7).

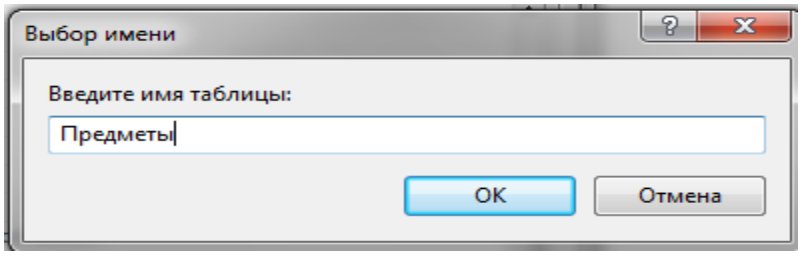


Рис. 3.7

Таблиця «Предмети» з'явиться в папці **«Таблицы»** в провіднику об'єктів

Після створення таблиці **«Предметы»** створіть таблицю **«Студенты»** (рис. 3.8).

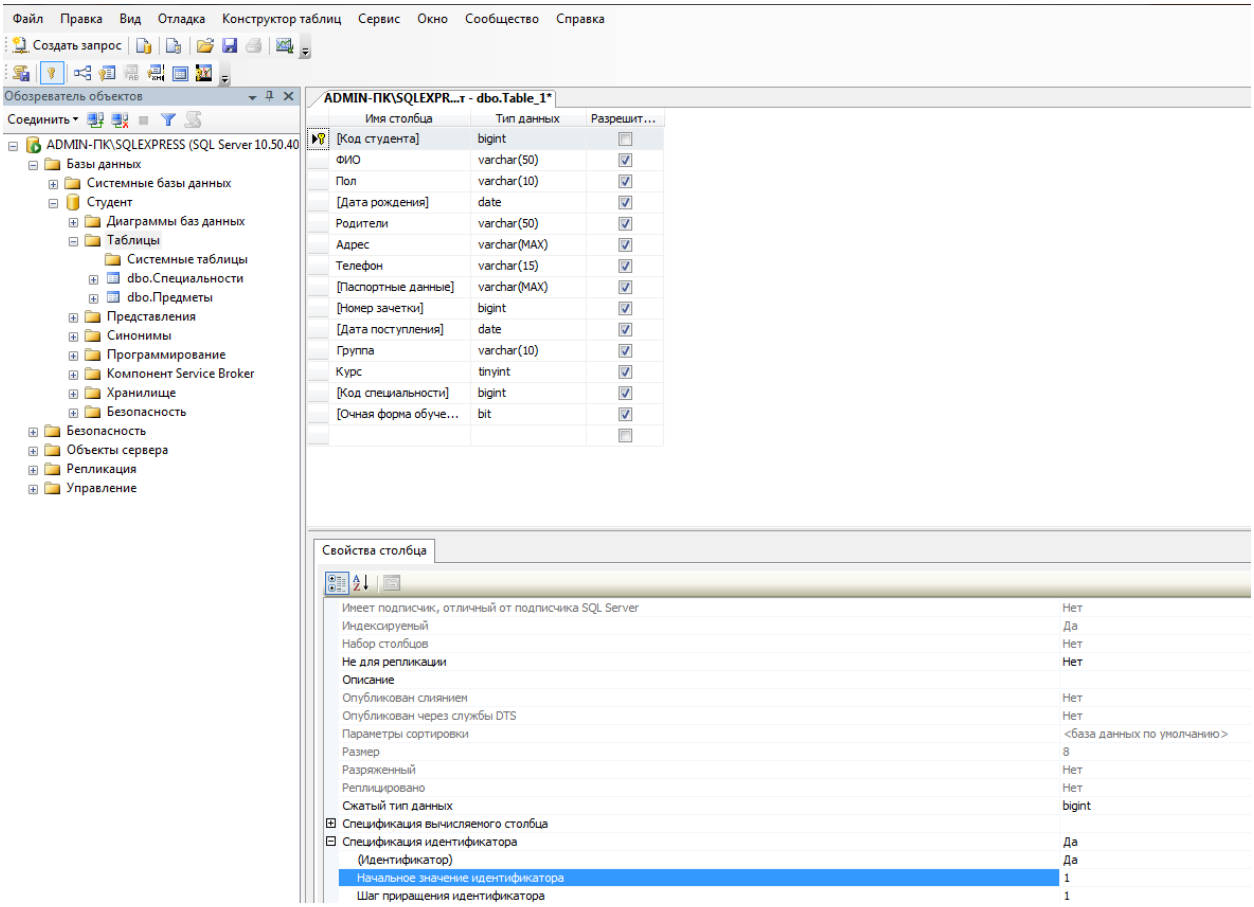


Рис. 3.8

Розглядаючи поля нової таблиці можна прийти до наступних висновків:

- Поле **«Код студента»** – це первинне поле для зв'язку з таблицею оцінки. Отже, дане поле необхідно зробити числовим лічильником і ключовим (див. Створення таблиці **«Специальности»** вище).
- Поля **«ФИО», «Пол», «Родители», «Адрес», «Телефон», «Паспортные данные»** і **«Группа»** є текстовими полями різної довжини (для завдання довжини виділеного текстового поля необхідно в таблиці властивостей виділеного поля встановити властивість **Length** рівне максимальній кількості знаків тексту вводиться в поле).
- Поля **«Дата рождения» і «Дата поступления»** призначені для зберігання дат. Тому вони мають тип даних **«date».**
- Поле «**Очная форма обучения**» є логічним полем. В «Microsoft SQL Server 2008» такі поля повинні мати тип даних «**bit».**
- Поля «**Номер зачетки» і «Курс»** є цілочисельними. Єдиною відмінністю є розмір полів. Поле «**Номер зачетки»** призначено для зберігання цілих чисел в діапазоні -263 ... + 263 (тип даних **«bigint»**). Поле **«Курс»** призначено для зберігання цілих чисел у діапазоні 0 ... 255 (тип даних **«tinyint»**).

- Поле **«Код специаальности»** – це поле зв’язку з таблицею **«Специальности»**. Однак, дане поле зв’язку є вторинним, тому його можна зробити просто цілочисловим, тобто, **«bigint»**.

Після визначення полів таблиці **«Студенты»**, закрийте вікно створення нової таблиці. У вікні **«Выбор имени»** задайте ім’я нової таблиці як **«Студенты»** (рис. 3.9)**.**

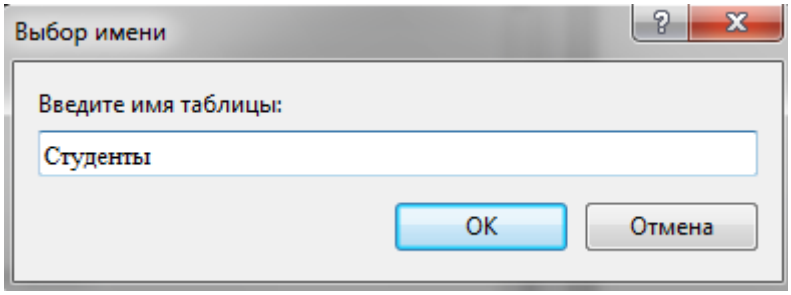


Рис. 3.9

Таблиця **«Студенты»** з’явиться в папці **«Таблицы»** в провіднику об’єктів. Нарешті, створимо таблицю **«Оценки»** (рис. 3.10).

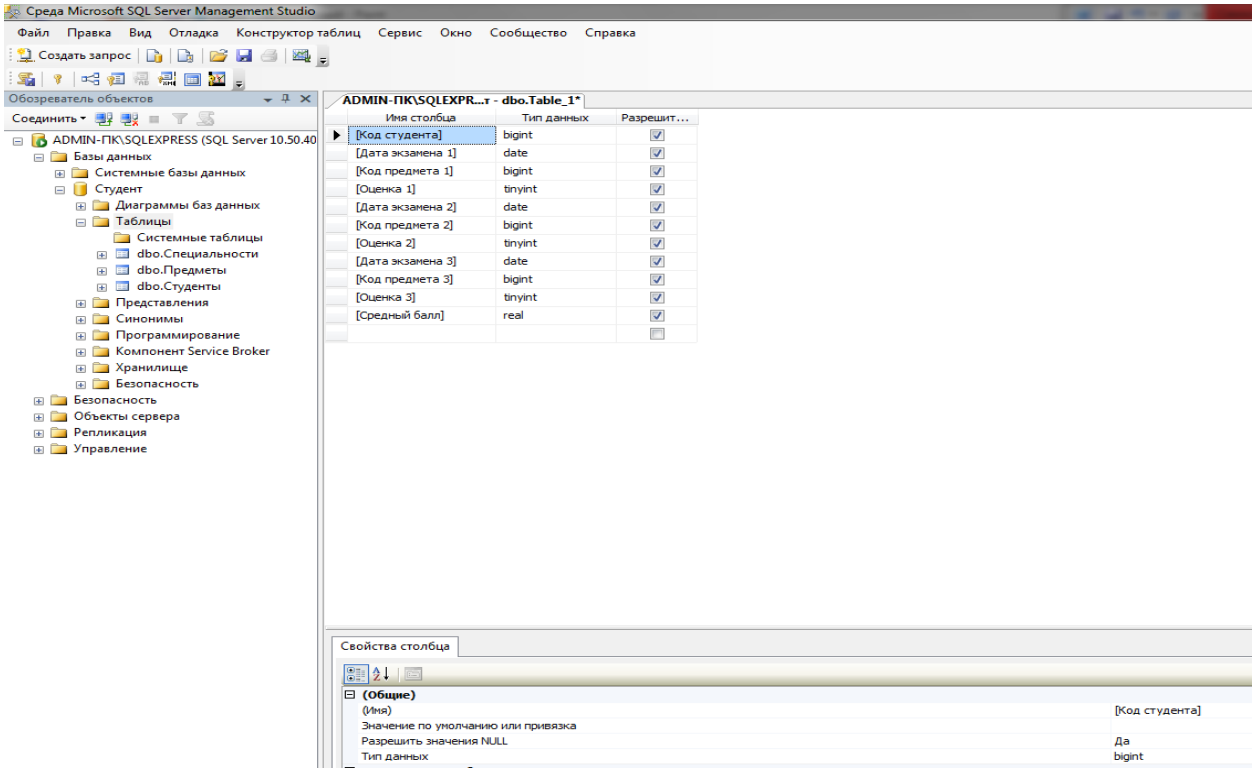


Рис. 3.10

Таблиця «**Оценки»** не має первинних полів зв’язку. Отже, ця таблиця не має ключових полів. Поля **«Код предмета 1», «Код предмета 2»** і **«Код предмета 3»** є вторинними полями зв’язку, призначеними для зв’язку з таблицею **«Предмети»**, тому вони є цілочисельними (тип даних **«bigint»**). Поля **«Дата экзамена 1», «Дата экзамена 2» і «Дата екзамена 3»** призначені для зберігання дат (тип даних «**date»**). Поля **«Оценка 1»,**

«Оценка 2», і «Оценка 3» призначені для зберігання оцінок. Задайте тип даних для цього поля **«tinyint»**. Нарешті, поле «**Средний балл»** зберігає дробові числа і має тип «**Real».** Закрийте вікно створення нової таблиці, задавши ім'я таблиці як **«Оценки»** (рис. 3.11)**.**

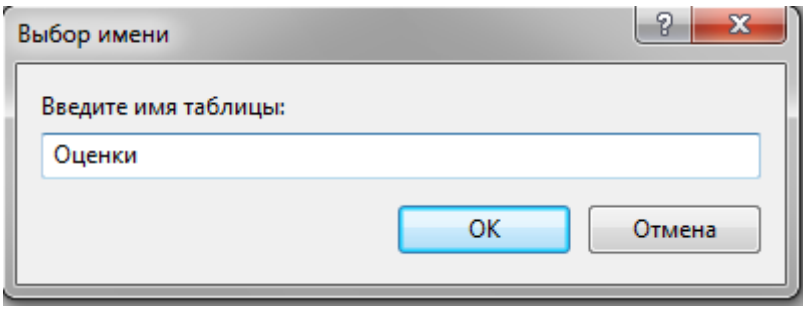


Рис. 3.11

На цьому ми закінчуємо створення таблиць БД **«Студент»**. Після створення всіх таблиць вікно оглядача об'єктів буде виглядати так (рис. 3.12).

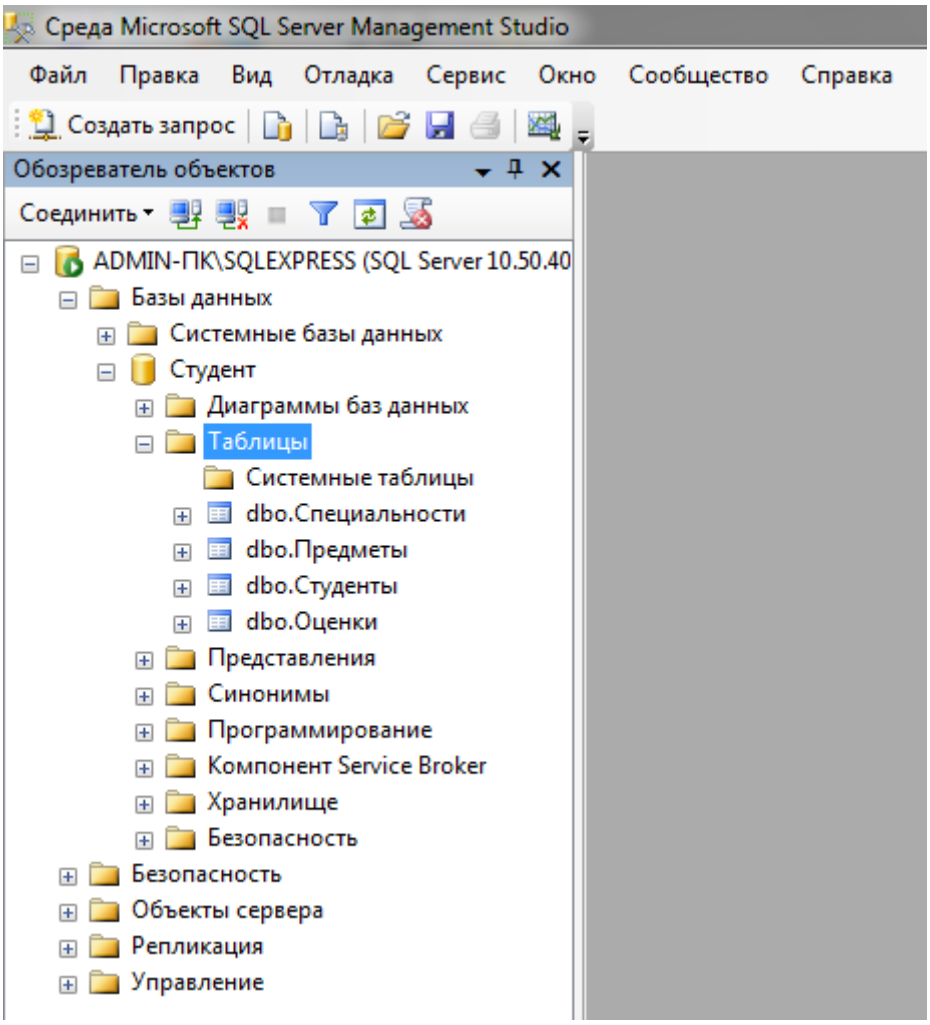


Рис. 3.12

Тепер розглянемо операцію заповнення таблиць початковими даними. Для початку заповнимо таблицю **«Специальности»**. Для заповнення цієї таблиці в оглядачі об'єктів клацніть правою кнопкою миші по таблиці **«Специальности»** і в меню виберіть пункт **«Изменить первые 200 строк»** (Редагувати перші 200 записів.). У робочій області

«**Microsoft SQL Server Management Studio»** проявиться вікно заповнення таблиць. Заповніть таблицю **«Специальности»** (рис. 3.13)**.**

	Код специальности	Наименование специальности	Описание специальности
	1	ММ	Математические методы
	2	ПИ	Прикладная информатика
	3	СТ	Статистика
	4	МО	Менеджмент организаций
▶	5	БУ	Бухгалтерский учёт
*	NULL	NULL	NULL

Рис. 3.13

***Зауваження*:** Заповнення таблиць відбувається повністю аналогічно табличному процесору «Microsoft Excel 2000».

***Зауваження*:** Так як поле **«Код специальности»** є первинним полем зв'язку і ключовим числовим лічильником, то воно заповнюється автоматично (заповнювати його не потрібно).

Закрийте вікно заповнення таблиці **«Специальнисти»** клацнувши по кнопці закриття вікна.

Після заповнення таблиці «**Специальности»** заповнимо таблицю **«Предметы»**. Відкрийте її для заповнення як описано вище, і заповніть (рис. 3.14).

	Код предмета	Наименование предмета	Описание предмета
	1	Операционые системы	Microsoft Windows Vista
	2	Офисные пакеты	Microsoft Office 2007
	3	Базы данных	Microsoft Access 2007
	4	Языки программирования	Microsoft Visual Studio 2008
▶	5	Проектирование информационных систем	Microsoft SQL server 2008
*	NULL	NULL	NULL

Рис. 3.14

Закрийте вікно заповнення таблиці «**Предметы»** і перейдіть до заповнення таблиці **«Студенты»**. Відкрийте таблицю **«Студенты»** для заповнення і заповніть її (рис. 3.15).

ACER-3AD5E49... dbo.Студенты

	Код сту...	ФИО	Пол	Дата рож...	Родители	Адрес	Телефон	Паспортные д...	Номер за...	Дата пост...	Группа	Курс	Код спец...	Очная форма...
	1	Иванов А.И.	Мужской	1983-12-12	Отец, Мать	Москва	+74957895674	8567-567543	13245	2007-09-01	ММ11	1	1	True
	2	Петрова И.И.	Женский	1982-11-01	Мать	Москва	+74957889876	4567-765432	34563	2006-08-01	ПИ21	2	2	False
	3	Мухин М.А.	Мужской	1982-05-14	Отец	Самара	+78462875690	5438-098787	56732	2006-07-05	СТ22	2	3	False
	4	Сидорова В.К.	Женский	1981-09-27	Нет	Саратов	+79027868909	1287-987509	27543	2005-06-23	МО31	3	4	True
	5	Кожевников А.А.	Мужской	1981-04-12	Мать	Казань	+79160563476	2312-675460	34217	2005-07-21	БУ33	3	5	True
	6	Пальчикова Н.Е.	Женский	1983-09-02	Отец, Мать	Челябинск	+74569098723	8743-856780	43278	2007-08-01	ММ12	1	1	False
	7	Царегородцев Е.В.	Мужской	1980-02-17	Отец	Самара	+78462234769	6543-834521	43765	2004-07-04	ПИ41	4	2	True
	8	Баранова Г.В.	Женский	1980-07-09	Отец, Мать	Чебоксары	+79027874638	2133-896567	10387	2004-08-09	СТ42	4	3	False
	9	Леухин П.Г.	Мужской	1979-02-26	Нет	Казань	+79067453678	2769-634904	67348	2003-07-23	МО51	5	4	True
▶	10	Николаева А.П.	Женский	1979-03-17	Мать	Саратов	+78546456432	3256-496932	45287	2003-06-21	БУ53	5	5	False
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рис. 3.15

***Зауваження*:** Для заповнення дат як роздільник можна використовувати знак «.». Дати можна заповнювати у форматі «день.місяць.рік».

***Зауваження*:** Поле **«Код специальности»** є вторинним полем зв'язку (для зв'язку з таблицею «**Специальности»**). Отже, значення цього поля необхідно заповнювати значеннями поля «**Код специальности»** таблиці «**Специальности»**. У нашому випадку це значення від 1 до 5. Якщо у Вас коди спеціальностей в таблиці «**Специальности»** мають інші значення, то внесіть їх в таблицю **«Студенты»**.

По закінченні заповнення, закрийте вікно заповнення таблиці «**Студенты»**.

Нарешті заповнимо таблицю **«Оценки»** (рис. 3.16).

ACER-3AD5E49... - dbo.Оценки

Код студента	Дата экзамена 1	Код предмета 1	Оценка 1	Дата экзамена 2	Код предмета 2	Оценка 2	Дата экзамена 3	Код предмета 3	Оценка 3	Средний балл
1	2008-02-01	1	5	2008-02-09	4	3	2008-02-14	2	4	0
2	2008-01-30	5	4	2008-02-23	3	5	2008-02-27	1	5	0
3	2008-01-26	3	5	2008-02-05	1	3	2008-02-15	5	3	0
4	2007-12-26	2	3	2008-01-05	4	4	2008-01-21	3	4	0
5	2008-01-12	4	4	2008-01-18	5	4	2008-01-25	1	4	0
6	2007-12-17	2	4	2007-12-26	4	5	2008-01-05	1	3	0
7	2008-02-21	5	2	2008-02-25	1	2	2008-02-27	2	4	0
8	2008-02-03	3	3	2008-02-12	5	3	2008-02-20	4	5	0
9	2008-01-25	1	5	2008-02-02	3	5	2008-02-14	5	5	0
10	2007-12-28	4	4	2008-01-11	1	4	2008-01-23	2	3	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рис. 3.16

***Зауваження*:** Поля з датами заповнюються, як і в таблиці **«Студенты».**

***Зауваження*:** Поля **«Код предмета 1», «Код предмета 2» і «Код предмета 3»** є вторинними полями зв'язку з таблицею **«Предметы»**. Тому вони повинні бути заповнені значеннями поля **«Код предмета «** з цієї таблиці, тобто значеннями від 1 до 5.

Закрийте вікно заповнення таблиці «**Оценки».** На цьому ми закінчуємо створення та заповнення таблиць нашої БД **«Студент»**.

Лекція 4: Створення запитів і фільтрів. Обчислення за допомогою оператора SELECT. Вбудовані функції

Мета:

1. Вивчити створення запитів і фільтрів.
2. Зрозуміти процес виконання обчислень за допомогою оператора SELECT. Вбудовані функції.

Створення запитів і фільтрів

Запити призначені для зв'язку однієї або декількох таблиць, також вони можуть здійснювати відбір окремих полів з таблиці і виробляти фільтрацію даних згідно з умовою, накладеною на одне або кілька полів, такі запити називають фільтрами.

Для реалізації запитів використовують спеціальну мову запитів SQL (Structured Query Language).

У ІВ запити можуть знаходитися як на стороні клієнтського додатка, так і на стороні сервера. Якщо запит зберігається на стороні клієнта, то він прописується всередині об'єкта зв'язку. У цьому випадку клієнтський додаток не залежить від файла даних. Файл даних містить тільки таблиці, тому, ми легко можемо модифікувати клієнтську програму, не зачіпаючи файл даних, але в цьому випадку запит передається серверу через мережу, що може викликати проблеми з безпекою.

Якщо запит зберігається або виконується на сервері, то сам запит виступає в якості компонента БД, вся передача інформації відбувається всередині файла даних, тобто всередині самого сервера, клієнтського додатка тільки передаються результати виконання запиту. У цьому випадку забезпечується високий захист даних, але у випадку зміни запиту доведеться міняти сам файл даних.

Всі запити діляться на:

1. Статичні.
2. Динамічні.

Структура статичних запитів незмінна в ході роботи з програмою, а динамічні запити можуть змінюватися залежно від ситуації.

***Зауваження*:** Зазвичай динамічні запити можуть бути реалізовані тільки за допомогою запитів, що виконуються на стороні клієнта. Якщо необхідно реалізувати динамічні запити, які виконуються на стороні сервера, то в цьому випадку необхідно використовувати збережені процедури.

Збережені процедури – SQL запит, збережений на стороні сервера і цей запит має параметри, які підставляються всередину SQL коду. Під час виклику збереженої процедури необхідно передавати в неї значення параметра.

В основному запит або збережена процедура або реалізує зв'язок між таблицями, або здійснює фільтрацію даних, деякі SQL запити також можуть робити обчислення.

У разі зв'язків між таблицями одна таблиця завжди виступає первинною, а інша вторинною, зв'язок відбувається за допомогою полів зв'язку. Під час зв'язку порівнюються записи з однаковими значеннями полів зв'язку. Первинна таблиця завжди заповнюється першою, а її поля зв'язку заповнюється автоматично (тип даних – лічильник). Вторинна таблиця завжди заповнюється після заповнення первинної таблиці, значення її поля зв'язку підставляється з значень поля зв'язку первинної таблиці. Поля зв'язку повинні мати однаковий тип даних.

Існує чотири види зв'язку між таблицями:

1. Одна до однієї – одному полю в первинній таблиці відповідає одне поле у вторинній таблиці.
2. Одна до багатьох – одному полю в первинній таблиці відповідає декілька полів у вторинній таблиці.
3. Багато до однієї – декількох полях в первинній таблиці відповідає одне поле у вторинній таблиці.
4. Багато до багатьох – одному полю в первинній таблиці відповідає декілька полів у вторинній таблиці і навпаки.

Запити з першими видами зв'язку називаються простими, а з рештою видів зв'язку – складними. Якщо в БД є хоча би дві пов'язаних таблиці, то БД називається реляційною.

Щоб створити запит необхідно зробити активною БД для якої створюється запит, потім в робочій області редактора запитів створити запит за допомогою команди SELECT, що має наступний синтаксис:

SELECT [ALL | DISTINCT]
[TOP | PERCENT n]
<Список полів>
[INTO <Ім'я нової таблиці>]
[FROM <Ім'я таблиці>]
[WHERE <Умова>]
[GROUP BY <Поле>]
[ORDER BY <Поле> [ASC | DESC]]
[COMPUTE AVG | COUNT | MAX | MIN | SUM (<Вираз>)]

Тут параметри **ALL | DISTINCT** показують, які записи обробляються: **ALL** обробляє всі записи, **DISTINCT** тільки унікальні, видаляються повторення записів.

TOP n визначає, яка кількість записів обробляється, якщо вказано PERCENT, то n вказує відсоток від загального числа записів. **<Список полів>** – тут вказуються відображувані поля з таблиць через кому.

***Зауваження*:**

1. Якщо імена полів, що відображаються в різних таблицях не повторюються, то ми можемо вказувати тільки імена стовпців або полів без вказівки самих таблиць (ПІБ, посада). Якщо відображаються **поля з різних таблиць з однаковими іменами потрібно вказувати і ім'я таблиці <Ім'я поля>. <Ім'я таблиці>.**
2. Тут же можна привласнювати псевдоніми полях, наступним чином **<Ім'я поля> AS <Псевдонім>**.
3. Якщо необхідно вивести всі поля з таблиці, то їх можна замінити значком «*».

Розділ **INTO**. Якщо присутній цей розділ, то на основі результатів запиту створюється нова таблиця.

Розділ **FROM**. Тут вказуються таблиці і запити через кому, які беруть участь в новому запиті.

***Зауваження*:** У розділі **FROM** так само можна задавати складні зв'язки, зв'язок поля однієї таблиці, з декількома полями іншої таблиці. У цьому випадку розділ FROM буде мати наступний вигляд:

FROM <Таблиця 1> INNER JOIN <Таблиця 2> ON <Таблиця 1>. <Поле 1> оператор <Таблиця 2>. <Поле 2> ...

Тут встановлюється взаємозв'язок **Таблиці 1 і Табліці 2** по **Полі 1 і Полі 2** залежно від оператора порівняння. Таких розділів **INNER JOIN** може бути скільки завгодно.

Розділ **WHERE**. Даний розділ використовують для створення простих запитів, в цьому випадку в якості умови вказуємо зв'язувані поля, або цей розділ використовують для створення фільтрів, тут вказують умови відбору. В умовах відбору ми можемо використовувати стандартні логічні оператори **NOT, OR, AND.**

***Зауваження*:** У своєму стандартному вигляді запити можуть реалізовувати тільки статичні фільтри, але не динамічні. Для реалізації динамічних фільтрів використовуються збережені процедури.

Розділ **GROUP BY** – визначає поле для групування записів у запиті.

Розділ **ORDER BY** – визначає поле для сортування записів у запиті. Якщо вказаний параметр **ASC**, то буде проводитися сортування за зростанням, якщо **DESC** – спаданням. За замовчуванням використовується сортування за зростанням.

Розділ **COMPUTE** дозволяє в кінці результатів виконання запиту вивести деякі підсумкові обчислення за запитом. Можливі такі види обчислень: **AVG** – середня параметра; **COUNT** – кількість значень параметра не дорівнює **NULL; MAX і MIN** – максимальні і мінімальні значення параметра; **SUM** - сума всіх значень параметра, де **<Вираз>** – сам параметр. Як параметр зазвичай виступають які-небудь поля таблиць, що беруть участь у запиті.

Приклад: Даний запит пов'язує дві таблиці «Співробітники» та «Посади» на полях Код. Під час виконання він відображає перші 20 відсотків співробітників з обох таблиць. З таблиці «Співробітники» відображаються всі поля, а з таблиці «Посади» тільки поле посада. Наприкінці результатів виводиться кількість відображених співробітників.

SELECT TOP 20 PERCENT *. Співробітники, Посада. Посади
FROM Співробітники, Посади
WHERE Код. Співробітники = Код. Посади
COMPUTE COUNT (ПІБ. Співробітники)

Приклад: Даний запит з таблиці Операції виводить всі записи, значення поля Місяць у яких дорівнює «Травень». Дані в результаті групуються по полю операція і сортуються за сумою операції. Наприкінці результатів запиту відображається загальна сума відібраних операцій за травень. Результати даного запиту зберігаються в таблиці «Угоди за травень».

SELECT ALL Операція, Сума
INTO [Угоди за Травень]
FROM Операції
WHERE Місяць = травень
GROUP BY Операція
ORDER BY Сума
COMPUTE SUM (Сума)

***Зауваження*:** У даному запиті при позначенні назви полів таблиці явно не вказуються, оскільки використовувалася тільки одна таблиця.

Виконання обчислень за допомогою оператора SELECT. Вбудовані функції

Крім зв'язування таблиць і відбору даних оператор **SELECT** може використовуватися для обчислень. У цьому випадку він має синтаксис:

SELECT <Вираз>

де **<вираз>** – якийсь математичний вираз чи функція. Вираз має стандартний вид (як в Visual Basic), він може включати в себе вбудовані функції сервера.

Зауваження: Ми можемо використовувати вбудовані функції і вирази в обчислюваних полях під час створення таблиць.

У SQL Server існують наступні вбудовані функції, розбиті на групи.

Математичні функції

***Зауваження*:** В якості параметрів функції будемо вказувати відповідний їм тип даних.

- **ABS (numeric)** – модуль числа.
- **ACOS / ASIN / ATAN (Float)** – арккосинус, арксинус, арктангенс в радіанах.
- **COS / SIN / TAN / COT (Float)** – косинус, синус, тангенс, котангенс.
- **CEILING (Numeric)** – найменше ціле, більше або рівне параметру в дужках.
- **DEGREES (Numeric)** – перетворює радіани в градуси.
- **EXP (Float)** – експонента, ex.
- **FLOOR (Numeric)** – найбільше ціле менше або рівне висловом numeric.
- **LOG (Float)** – натуральний логарифм ln.
- **LOG10 (Float)** – десятковий логарифм log10.
- **PI ()** – число пі.
- **POWER (Numeric, y)** – зводить вираз Numeric в ступінь y.
- **RADIANS (Numeric)** – перетворює градуси в радіани.
- **RAND ()** – генерує випадкове число типу даних Float, розташоване між нулем і одиницею.
- **ROUND (Numeric, Довжина)** – округлює вираз Numeric до заданої Довжини (кількість знаків після комою).
- **SIGN (Numeric)** – виводить знак числа +/- або нуль.
- **SQUARE (Float)** – обчислює квадрат числа Float.
- **SQRT (Float)** – обчислює квадратний корінь числа Float.

Приклади використання математичних функцій:

- **SELECT ABS (-10)** результат 10.
- **SELECT SQRT (16)** результат 4.
- **SELECT ROUND (125.85,0)** результат 126.
- **SELECT POWER (2,4)** результат 16.

Рядкові функції

Рядкові функції дозволяють робити операції з однією або кількома рядками.

- **'Рядок 1' + 'рядок 2'** приєднує Рядок 1 до Рядок 2.
- **ASCII (Char)** – повертає ASCII код з самого лівого символу виразу Char.
- **CHAR (Int)** – виводить символ відповідний ASCII кодом у виразі Int.
- **CHARINDEX (Зразок, Вираз)** – виводить позицію зразка виразу, тобто де знаходиться зразок у вираженні.
- **DIFFERENCE (Вираз 1, Вираз 2)** – порівнює два вирази, виводить числа від 0 до 4: 0 – вирази абсолютно різні; 4 – вирази абсолютно ідентичні. Обидва висловлювання типу даних Char.
- **LEFT (Char, Int)** – виводить з рядка Char Int символів зліва.

- **RIGHT (Char, Int)** – виводить з рядка Char Int символів праворуч.
- **LTRIM (Char)** – видаляє з рядка Char пробіл зліва.
- **RTRIM (Char)** – видаляє з рядка Char пробіл праворуч.
- **WCHAR (Int)** – виводить вираз Int у форматі Unicode.
- **REPLACE (рядок 1, рядок 2, рядок 3)** – змінює в рядок 1 всі елементи рядок 2 на елементи рядок 3.
- **REPLICATE (Char, Int)** – повторює рядок Char Int разів.
- **REVERSE (Char)** – виробляє інверсію рядка Char, тобто розпорядженні символи у зворотному порядку.
- **SPACE (Int)** – виводить Int пробіла.
- **STR (Float)** – переводить число Float в рядок.
- **STUFF (Вираз1 , Початок, Довжина, Вираз 2)** – видаляє з Виразу 1 починаючи з позиції символу Початок кількість символів рівне параметру Довжина, натомість підставляє Вираз 2.
- **SUBSTRING (Вираз, Початок, Довжина)** – з Виразу виводиться рядок заданої Довжини починаючи з позиції Початок.
- **UNICODE (Char)** – виводить код у форматі Unicode першого символу в рядку Char.
- **LOWER (Char)** – переводить рядок Char в маленькі букви.
- **UPPER (Char)** – переводить рядок Char в заголовні букви.

Приклади застосування строкових функцій:

- **SELECT ASCII ('G')** результат 71.
- **SELECT LOWER ('ABC')** результат abc.
- **SELECT RIGHT ('ABCDE', 3)** результат CDE.
- **SELECT REVERSE ('МИР')** результат РИМ.

***Зауваження*.** У всіх строкових функціях значення виразу типу Char виділяється в одинарні лапки.

Функції дат

***Зауваження*:** В деяких функціях дат використовується так звана частина дат, яка кодується спеціальними символами:

dd – число дат (від 1 до 31);

dy – день року (число від 1 до 366);

hh – значення години (0 – 23);

ms – значення секунд (від 0 до 999);

mi – значення хвилин (0 – 59);

qq – значення (1– 4);

mm – значення місяців (1–12);

ss – значення секунд (0 – 59);

wk – значення номерів тижнів у році;

dw – значення днів тижня, тиждень починається з неділі (1 – 7);

yy – значення років (1753 – 999).

Функції дат призначені для роботи з датами або часу. Існують декілька наступних функцій дат:

- **DATEADD (частина, число, date)** – додає до дати date частину дати збільшену на число.
- **DATEDIFF (частина, date 1, date 2)** – виводить кількість частин дати між date 1 і date 2.
- **DATENAME (частина, date)** – виводить символьне значення частин дати до заданої дати (назва днів тижня).
- **DATEPART (частина, date)** – виводить числове значення частини дати з заданої дати (номер місяці).
- **DAY (date)** – виводить кількість днів в заданій даті.
- **MONTH (date)** – виводить кількість місяців в заданій даті.
- **YEAR (date)** – виводить кількість років в заданій даті.
- **GETDATE ()–** виводить поточну дату встановлену за комп'ютером.

***Зауваження*:** Дати виводяться в Американському форматі: місяць / день / рік.

Приклади функції робіт з датами:

- **SELECT DATEADD (dd, 5,11 / 20/07)** результат Nov / 25/2007.
- **SELECT DATEDIFF (dd, 11/20/07, 11/25/07)** результат 5 днів.
- **SELECT DATENAME (mm, 11/20/07)** результат November.
- SELECT DATEPART (mm, 11/20/07) результат 11.

***Зауваження*:** У виразах оператора SELECT можна використовувати операції порівняння. У результаті буде або істина TRUE, або брехня FALSE. Можна використовувати наступні оператори: =, <,>,> =, <=, <>,! <(не менше),!> (не більше),! = (Не дорівнює). Пріоритет операції задається круглими дужками.

Системні функції

Системні функції призначені для отримання інформації про базу даних та її вмісті. У SQL сервері існують наступні системні функції:

- **COL_LENGTH (таблиця, поле)** – виводить ширину поля.
- **DATALENGTH (вираз)** – виводить довжину виразу.
- **GETANSINULL (ім'я БД)** – виводить допустимо або неприпустимо використане в БД значення NULL.
- **IDENT_INCR (таблиця)** – виводить крок збільшення поля лічильника в таблиці.

- **IDENT_SEED (таблиця)** – виводить початкове значення лічильників в таблиці.
- **ISDATE (вираз)** – виводить одиницю, якщо вираз є датою і нуль, якщо не є.
- **ISNUMERIC (вираз)** – виводить одиницю, якщо вираз є числовим і нуль, якщо не числовим.
- **NULIFF (вираз 1, вираз 2)** – виводить NULL якщо вираз 1 рівний виразу 2.

Агрегатні функції

Агрегатні функції – дозволяють обчислювати підсумкові значення на полях таблиці.

- **AVG (поле)** – виводить середнє значення поля.
- **COUNT (*)** – виводить кількість записів у таблиці.
- **COUNT (поле)** – виводить кількість всіх значень поля.
- **MAX (поле)** – виводить максимальне значення поля.
- **MIN (поле)** – виводить мінімальне значення поля.
- **STDEV (поле)** – виводить середньоквадратичне відхилення всіх значень поля.
- **STDEVP (полі)** – виводить середньоквадратичне відхилення різноманітних значень поля.
- **SUM (поле)** – підсумовує всі значення поля.
- **TOP n [Percent]** – виводить n перших записів з таблиці або n % записів з таблиці.
- **VAR (поле)** – виводить дисперсію всіх значень поля.
- **VARP (поле)** – виводить дисперсію всіх різних значень поля.

Приклади використання агрегатних функцій:

- **SELECT AVG (вік) FROM Студенти** – виводить середній вік студента з таблиці «Студенти».
- **SELECT COUNT (ПІБ) FROM Студенти** – виводить кількість різних ПІБ з таблиці «Студенти».
- **SELECT Top 100 * FROM Студенти** – виводить перші 100 студентів з таблиці «Студенти».

Самостійна робота 4: Створення запитів і фільтрів

Мета: навчитися створювати запити і фільтри.

Перейдемо до створення статичних запитів. У оглядачі об'єктів «Microsoft SQL Server 2008» всі запити БД знаходяться в папці «**Представления»** (рис. 4.1).

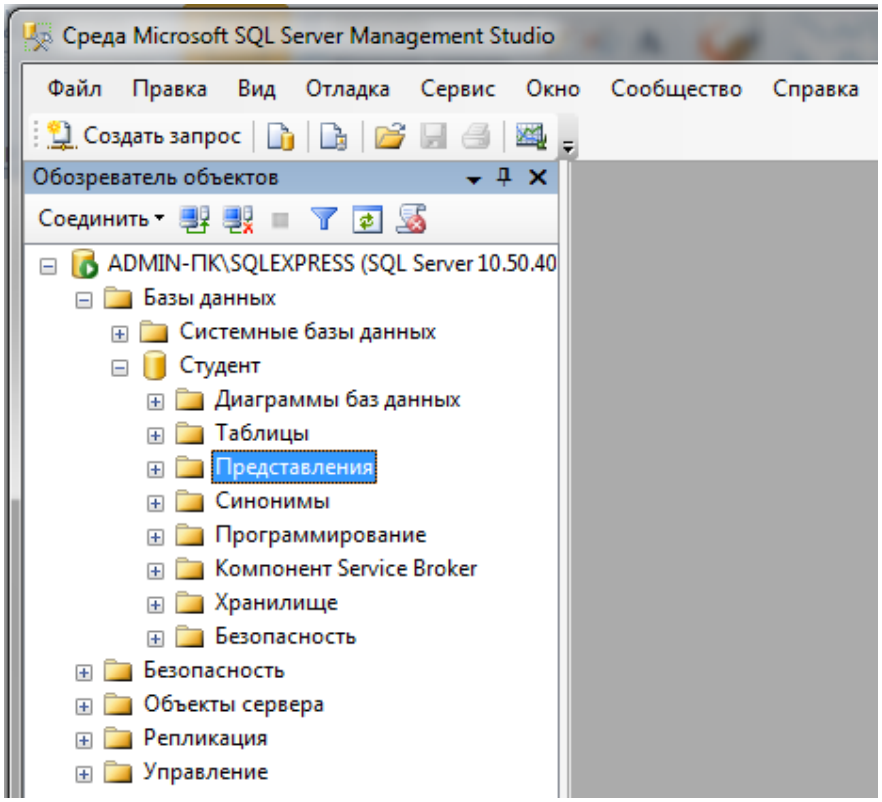


Рис. 4.1

Створимо запит «**Запрос Студенты+Специальности»**, що зв'язує таблиці «**Студенты**» і «**Специальности»** по полю зв'язку «**Код специальности»**. Для створення нового запиту необхідно в оглядачі об'єктів у БД «**Студент**» клацнути ПКМ по папці «**Представления»**, потім в меню вибрати пункт «**Создать представления**». З'явиться вікно «**Добавление таблицы»**, призначене для вибору таблиць і запитів, що беруть участь в новому запиті (рис. 4.2).

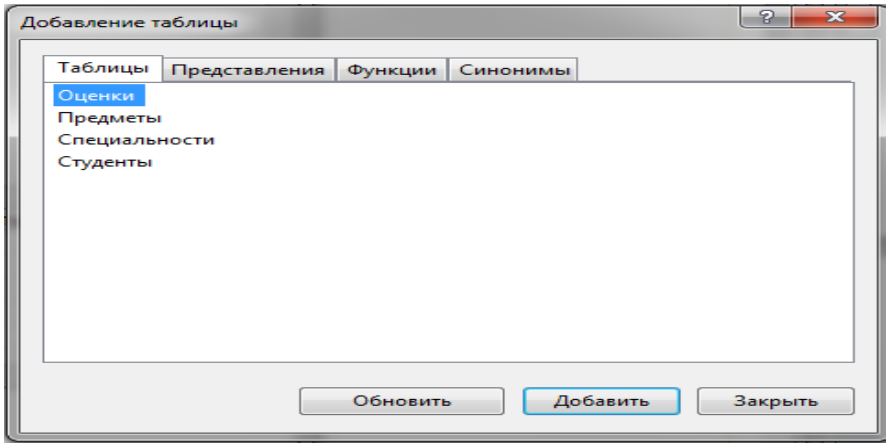


Рис. 4.2

Додамо в новий запит таблиці «**Студенты»** і «**Специальности»**. Для цього у вікні «**Добавление таблицы»** виділіть таблицю «**Студенты**» і натисніть кнопку «**Добавить»**. Аналогічно додайте таблицю «**Специальности»**. Після додавання таблиць закрийте вікно «**Добавление таблицы»** натиснувши кнопку «**Закрыть»**. З'явиться вікно конструктора запитів (рис. 4.3).

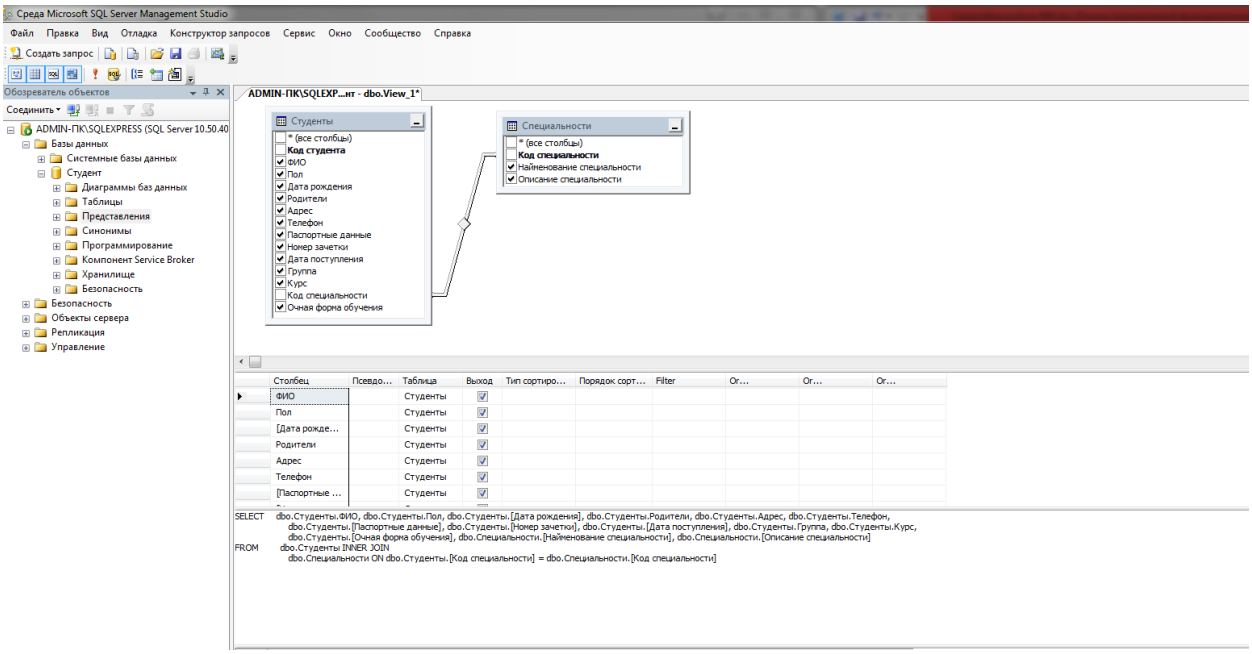


Рис. 4.3

***Зауваження*:** Вікно конструктора запитів складається з наступних панелей:

1. **Схема данных** – відображає поля таблиць і запитів, що беруть участь у запиті, дозволяє вибирати відображувані поля, дозволяє встановлювати зв'язки між учасниками запиту за спеціальними полями зв'язку. Ця панель вмикається та вимикається наступній кнопкою на панелі інструментів  .

2. **Таблица отображаемых полей** – показує відображувані, дозволяє задавати їм псевдоніми, дозволяє встановлювати тип сортування записів по одному або декількох полях, дозволяє задавати порядок сортування, дозволяє задавати умови відбору записів у фільтрах. Також ця таблиця дозволяє змінювати порядок відображення полів у запиті. Ця панель вмикається та вимикається наступній кнопкою на панелі інструментів  .

3. **Код SQL** – код створюваного запиту на мові T–SQL. Ця панель вмикається та вимикається наступній кнопкою на панелі інструментів  .

4. **Результат** – показує результат запиту після його виконання. Ця панель вмикається та вимикається наступній кнопкою на панелі інструментів  .

***Зауваження*:** Якщо необхідно знову відобразити вікно «**Добавляние таблицы»** для додавання нових таблиць або запитів, то для цього на панелі інструментів «**Microsoft SQL Server 2008»** потрібно натиснути кнопку  .

Зауваження: Якщо необхідно видалити таблицю або запит зі схеми даних, то для цього потрібно клацнути **ПКМ** і в меню вибрати пункт «**Удалить**».

Тепер перейдемо до зв'язування таблиць «**Студенты**» і «**Специальности**» по полях зв'язку «**Код специальности**». Щоб створити зв'язок необхідно в схемі даних перетягнути мишею поле «**Код специальности**» таблиці «**Специальности**» на таке ж поле таблиці «**Студенты**». Зв'язок відобразиться у вигляді ламаної лінії та з'єднає ці два поля зв'язку.

***Зауваження*:** Якщо необхідно видалити зв'язок, то для цього необхідно клацнути по ній **ПКМ** і в меню вибрати пункт «**Удалить**».

***Зауваження*:** Після зв'язування таблиць (а також при будь-яких змінах в запиті) в області коду T-SQL буде відображатися T-SQL код редагованого запиту.

Тепер визначимо поля, відображувані під час виконання запиту. Відображувані поля позначаються галочкою (зліва від імені поля) на схемі даних, а також відображаються в таблиці видимі поля. Щоб зробити поле відображуваним при виконанні запиту необхідно клацнути мишею по порожньому квадрату (зліва від імені поля) на схемі даних, в квадраті з'явиться галочка.

***Зауваження*:** Якщо необхідно зробити поле невидимим при виконанні запиту, то потрібно прибрати галочку, розташовану зліва від імені поля на схемі даних. Для цього просто клацніть мишею по галочці.

***Зауваження*:** Якщо необхідно відобразити всі поля таблиці, то необхідно встановити галочку зліва від пункту « * **Все поля**», що належить відповідній таблиці на схемі даних.

Визначте відображувані поля нашого запиту. (Відображаються всі поля крім полів з кодами, тобто полів зв'язку).

На цьому настройку нового запиту можна вважати закінченою. Перед збереженням запиту перевіримо його працездатність, виконавши його. Для запуску запиту на панелі інструментів натисніть кнопку  . Або клацніть **ПКМ** в будь-якому місці вікна конструктора запитів і в меню виберіть пункт « **Выполнить *SQL***». Результат виконання запиту з'явитися у вигляді таблиці в області результату.

***Зауваження*:** Якщо після виконання запиту результат не з'явився, а з'явилося повідомлення про помилку, то в цьому випадку перевірте, чи правильно створений зв'язок. Ламана лінія зв'язку повинна з'єднувати поля «**Код Специальности**» в обох таблицях. Якщо лінія зв'язку з'єднує інші поля, то її необхідно видалити і створити заново, як це описано вище.

Якщо запит виконується правильно, то необхідно зберегти. Для збереження запиту закрийте вікно конструктора запитів, клацнувши мишею по кнопці закриття розташованої у верхньому правому куті вікна конструктора (над схемою даних). З'явиться вікно з питанням про збереження запиту (рис. 4.4).

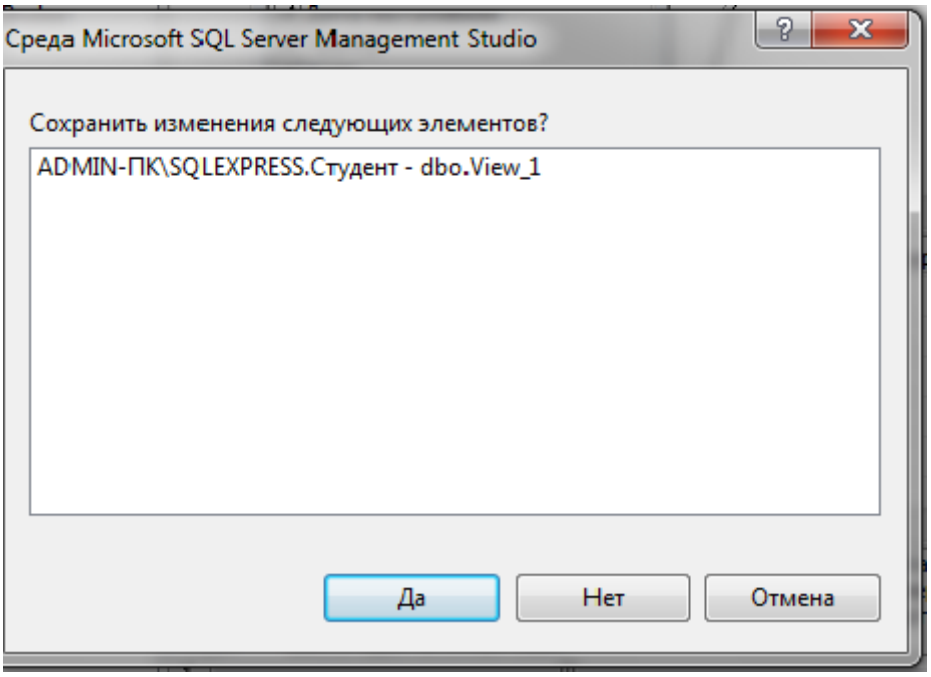


Рис. 4.4

У даному вікні необхідно натиснути кнопку «**Да**». З'являється вікно «**Выбор имени**» (рис. 4.5).

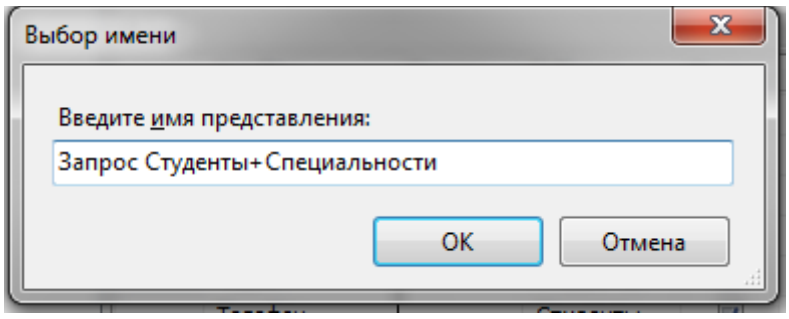


Рис. 4.5

У даному вікні задамо ім'я нового запиту «**Запит Студенты + Специальности**» і натиснемо кнопку «**Ok**». Запит з'явиться в папці «**Представления**» БД «**Студент**» в провіднику об'єктів (рис. 4.6).

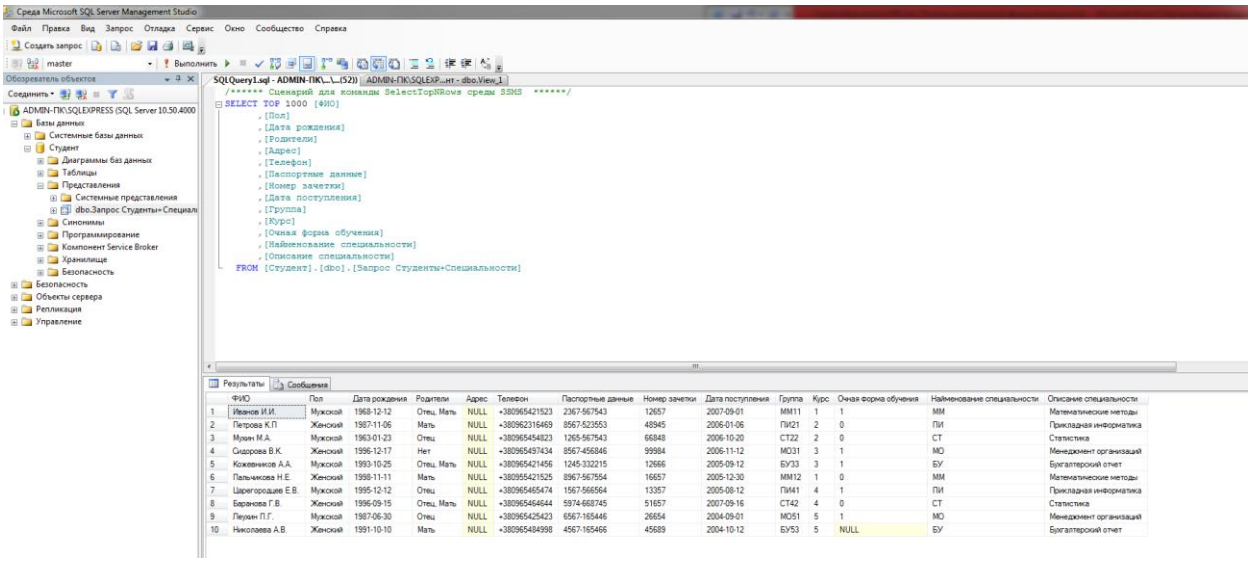


Рис. 4.6

Перевіримо роботу створеного запиту поза конструктором запитів. Запустимо новостворений запит «**Запрос Студенты + Специальности**» без використання конструктора запитів. Для виконання вже записаний запит необхідно клацнути **ПКМ** за запитом та в меню вибрати пункт «**Выбрать первые 1000 строк**». Виконайте цю операцію для запиту «**Запрос Студенты + Специальности**».

Перейдемо до створення запиту «**Запрос Студенты + Оценки**». В оглядачі об'єктів у БД «**Студент**» клацніть **ПКМ** по папці «**Представления**», потім в меню оберіть пункт «**Создать представление**». З'явитися вікно «**Добавление таблицы**». У запиті «**Запрос Студенты + Оценки**» ми пов'язуємо таблиці «**Студенты**» і «**Оценки**» по полях зв'язку «**Код студента**». Отже, у вікні **«Добавление таблицы «** в новий запит додаємо таблиці «**Студенты**» і «**Оценки**». Більш того, в даному запиті таблиця «**Оценки**» зв'язується з таблицею «**Предметы**» не по одному полю, а по трьох полях. Тобто поля «**Код предмету 1**», «**Код предмету 2**» і «**Код предмету 3**» таблиці «**Оценки**» пов'язані з полем «**Код предмету**» таблиці «**Предметы**». За цього додамо в запит три екземпляри таблиці «**Предметы**» (по одному примірнику для кожного поля зв'язку таблиці Оценки). У підсумку в запиті повинні брати участь таблиці «**Студенты**», «**Оценки**» і три екземпляри таблиці «**Предметы**» (в запиті вони будуть називатися «**Предметы**», «**Предметы_1**» і «**Предметы_2**»). Після додавання таблиць закрийте вікно «**Добавление таблицы**», з'явиться вікно конструктора запитів.

У вікні конструктора запитів встановіть зв'язки між таблицями і визначте відображувані поля (рис. 4.7).

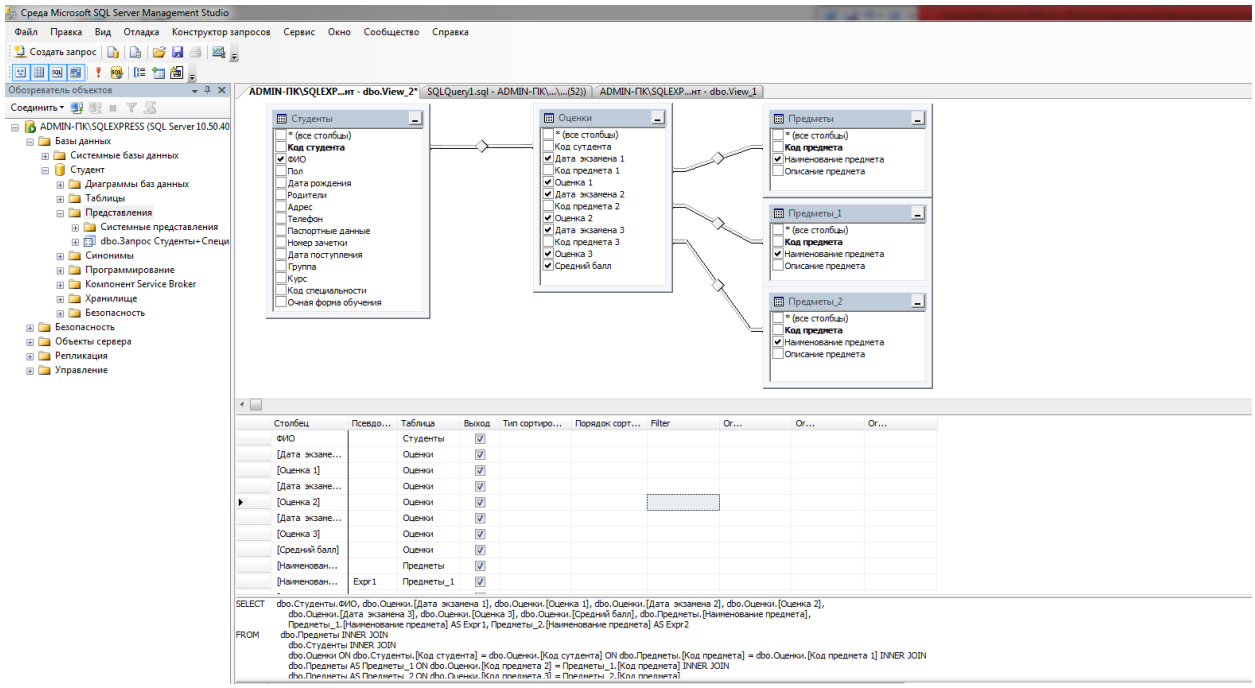


Рис. 4.7

Тепер змініть порядок полів, що відображаються у запиті, для цього в таблиці видимі поля необхідно перетягнути поля мишею вгору або вниз за заголовок рядка таблиці. Розташуйте відображувані поля в таблиці відображуваних полів (рис. 4.8).

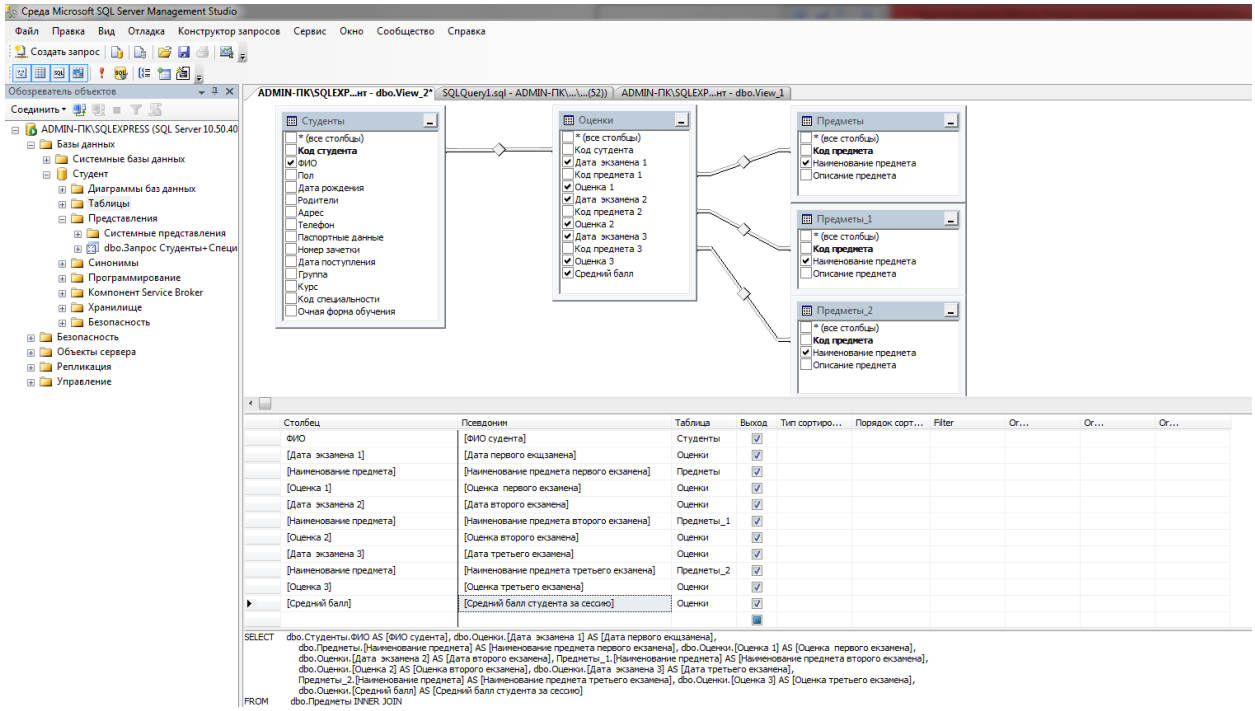


Рис .4.8

Задайте псевдоніми для кожного з полів, просто записавши псевдоніми в стовпці «**Псевдоним**» таблиці видимі поля.

Перевірте працездатність нового запиту, виконавши його. Зверніть увагу на те, що реальні назви полів були замінені їх псевдонімами. Закрийте вікно конструктора запитів. У вікні «**Выбор имени**» задайте ім'я нового запиту «**Запрос Студенты + Оценки**» (рис. 4.9).

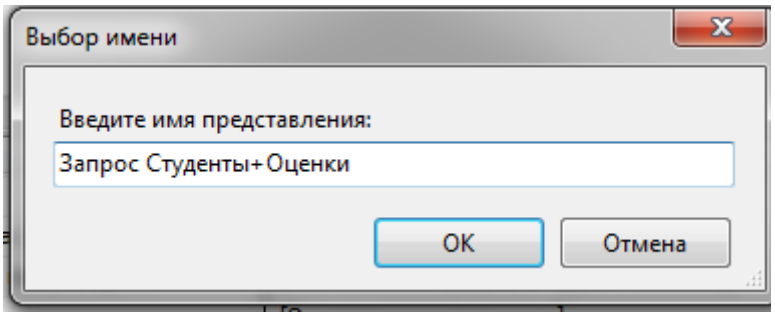


Рис. 4.9

Перевірте працездатність нового запиту поза конструктором. Для цього запустіть запит. Результат виконання запиту «**Запрос Студенты + Оценки**» (рис. 4.10).

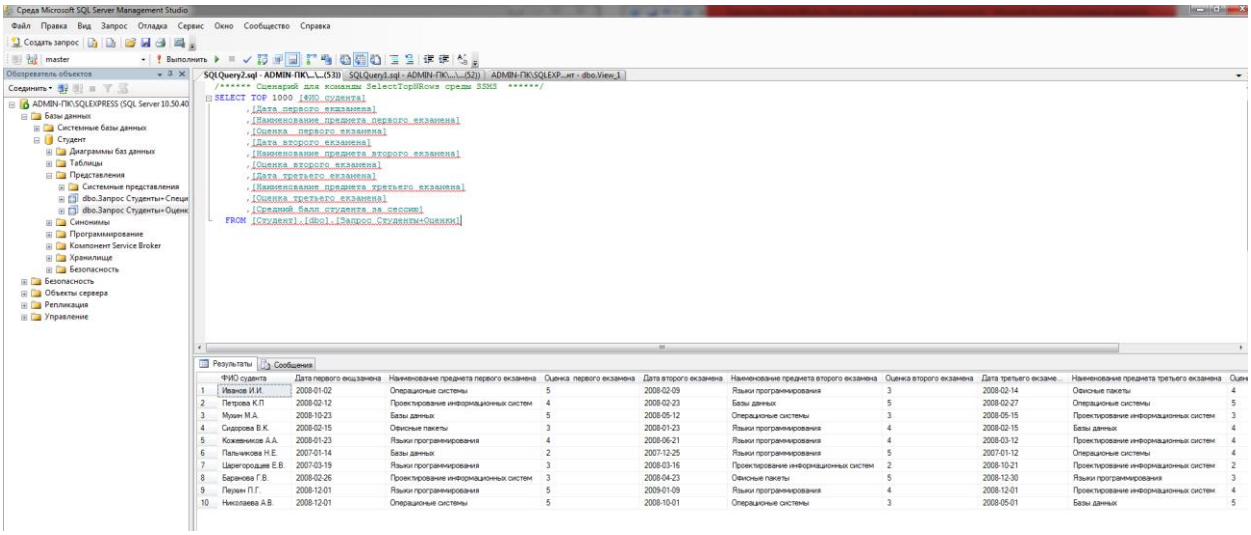


Рис. 4.10

На основі запиту «**Запрос Студенты + Специальности**» створимо фільтри, що відображають студентів окремих спеціальностей. Створіть новий запит. Так як він буде заснований на запиті «**Запрос Студенты + Специальности**», то у вікні «**Добавление таблицы**» перейдіть на вкладку «**Представления**» і додайте в новий запит «**Запит Студенты + Специальности**». Потім закрийте вікно «**Добавление таблицы**» (рис. 4.11).

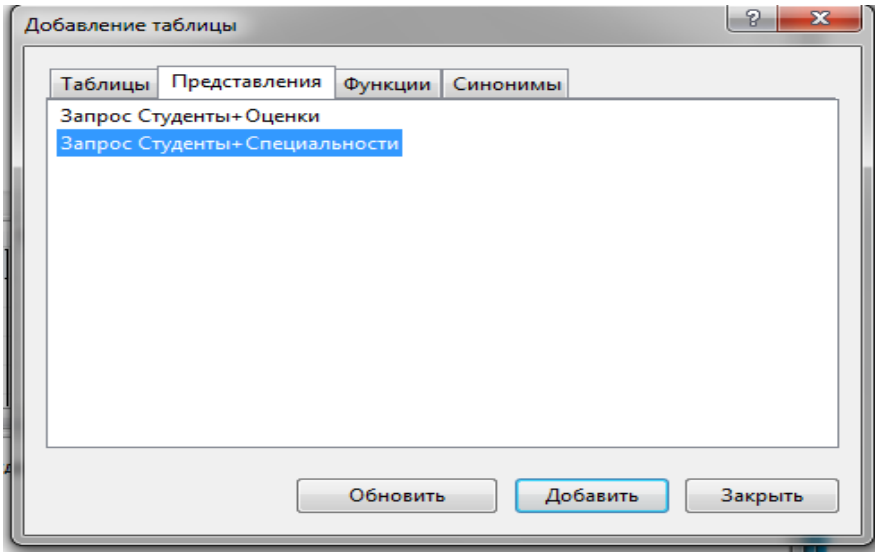


Рис. 4.11

У вікні конструктора запитів визначте в якості полів, що відображаються, всі поля запиту «**Запит Студенты + Специальности**» (рис. 4.12).

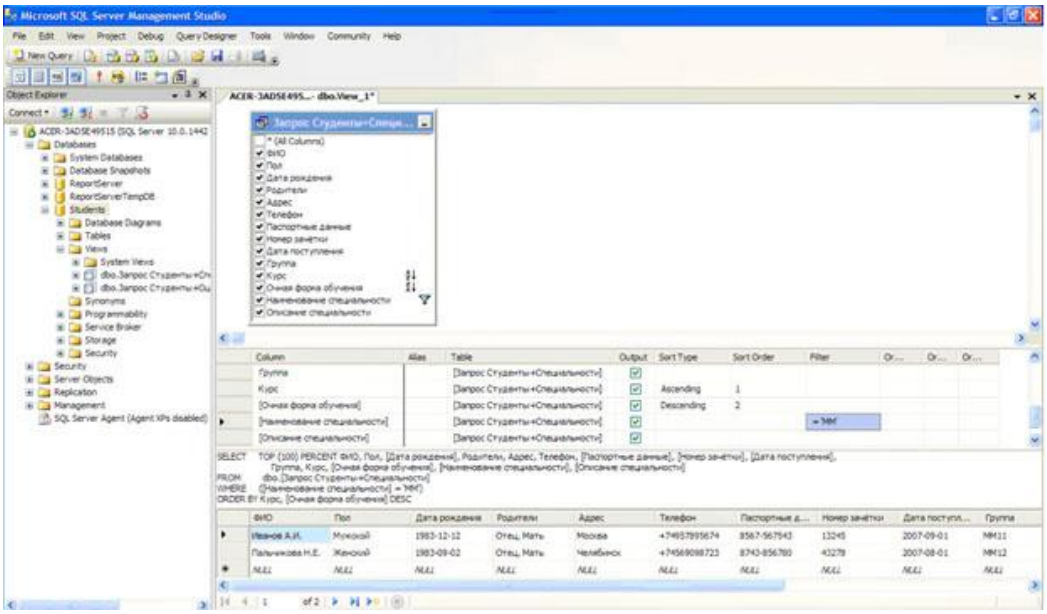


Рис. 4.12

***Зауваження*:** Для відображення всіх полів запиту, в даному випадку, ми не можемо використовувати пункт **« * Все поля»**. Так як в цьому випадку ми не можемо встановлювати критерій відбору записів у фільтрі, а також неможливо встановити сортування записів.

Тепер встановимо критерій відбору записів у фільтрі. Нехай наш фільтр відображає тільки студентів, які мають спеціальність «**ММ**». Для визначення умови відбору записів у таблиці полів, що відображаються в рядку, що відповідає полю, на яке накладається умова, у стовпці «**Filter**», необхідно задати умову. У нашому випадку умова накладається на поле «**Наименование специальности**». Отже, у рядку «**Наименование специальности**», у стовпці «**Filter**» потрібно задати наступну умову відбору «= **'ММ'**».

На завершення налаштуємо сортування записів у фільтрі. Нехай під час виконання фільтра спочатку відбувається сортування записів за зростанням по полю «**Очна форма навчання**», а потім по зменшенню по полю «**Курс**». Для установки сортування записів за зростанням, у таблиці визначаються поля, у рядку для поля «Очна форма навчання», у стовпці «**Тип сортировки**», задайте «**По возростанию**», а в рядку для поля «**Курс**» – задайте «**По убыванию**». Для визначення порядку сортування для поля «Очна форма навчання» в стовпці «**Порядок сортировки**» поставте 1, а для поля «**Курс**» поставте 2. Тобто, під час виконання запиту записи спочатку упорядковано по полю «**Очна форма обучения**», а потім по полю «**Курс**».

***Зауваження*:** Після установки умов відбору і сортування записів на схемі даних навпроти відповідних полів з'являться спеціальні значки. Значки  і  позначають сортування по зростанню і спаданню, а значок  показує наявність умови відбору.

Після установки сортування записів у фільтрі перевіримо його працездатність, виконавши його. Закрийте вікно конструктора запитів. Як ім'я нового фільтра у вікні «**Выбор имени**» задайте «**Фильтр ММ**» і натисніть кнопку «**Ok**» (рис. 4.13).

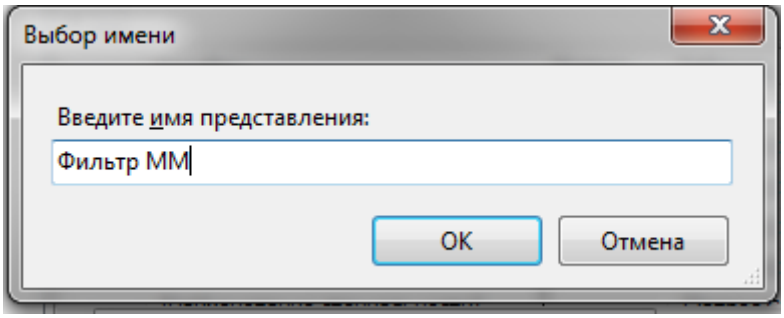


Рис. 4.13

Фільтр «**Фильтр ММ**» з'явиться в оглядачі об'єктів. Виконайте створений фільтр поза вікном конструктора запитів (рис. 4.14).

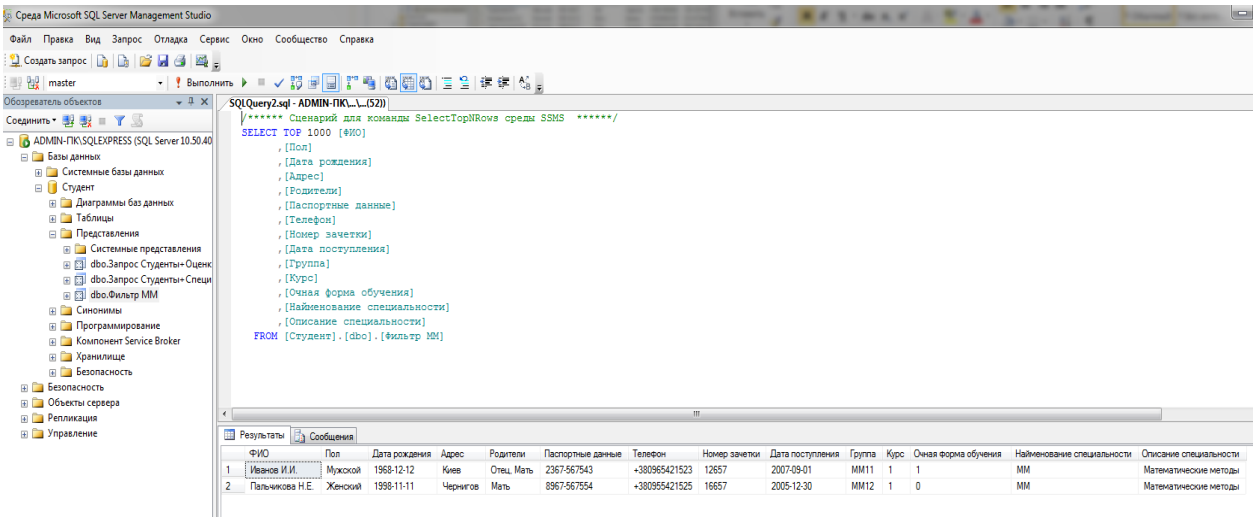


Рис. 4.14

Самостійно створіть фільтри для відображення інших спеціальностей. Дані фільтри створюються аналогічно фільтру «**Фильтр ММ**» (дивися вище). Єдиною відмінністю є умова відбору, що накладається на поле «**Наименование Специальности**», воно повинно бути не «= **'ММ'**», а «= **'Пи'**», «= **'СТ'**», «= **'МО'**» або «= ' **БО'**». Під час збереження фільтрів задаємо їхні імена відповідно їх умовам відбору, тобто «**Фильтр ПИ**», «**Фильтр СТ**», «**Фильтр МО**» або «**Фильтр БО**». Перевірте створені фільтри на працездатність.

Тепер на основі запиту «**Запит Студенты + Специальности**» створимо фільтри, які відображатимуть студентів мають окремих батьків. Для початку створимо фільтр для студентів, з батьків тільки «**Отец**». Створіть новий запит і додайте в нього запит «**Запрос Студенты + Специальности**». Після закриття вікна «**Добавление таблицы**» зробіть відображеними всі поля запиту (рис. 4.15).

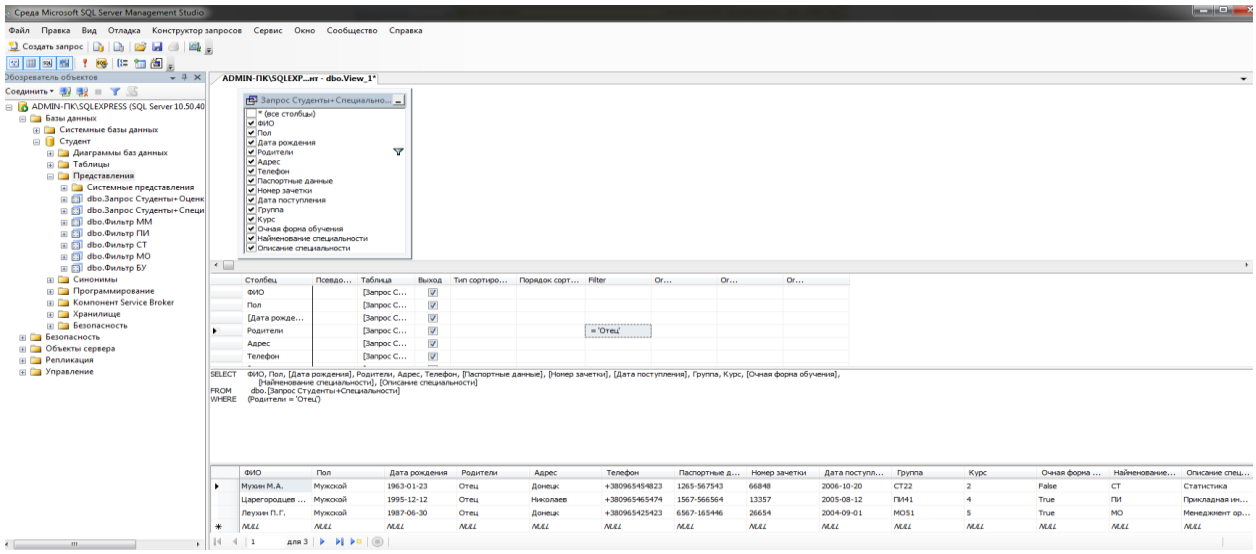


Рис. 4.15

У таблиці видимі поля в рядку для поля «**Родители**», у стовпці «**Filter**», задайте умову відбору рівне «= **Отец**». Перевірте роботу фільтра, виконавши його.

Закрийте вікно конструктора запитів. У вікні «**Выбор имени**» задайте ім'я нового фільтра як «**Фильтр Отец**» (рис. 4.16).

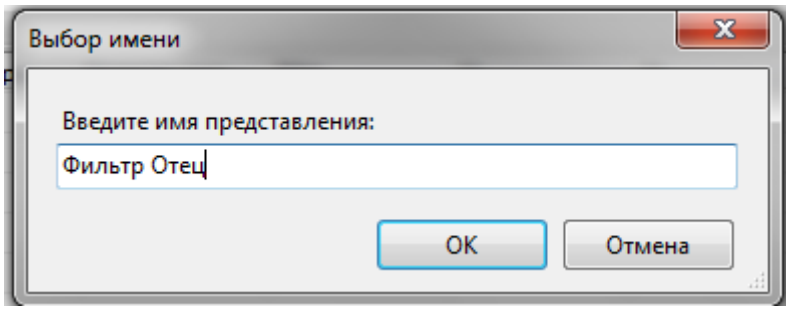


Рис. 4.16

Виконайте фільтр «**Фильтр Отец**» поза конструктором запитів (рис. 4.17).

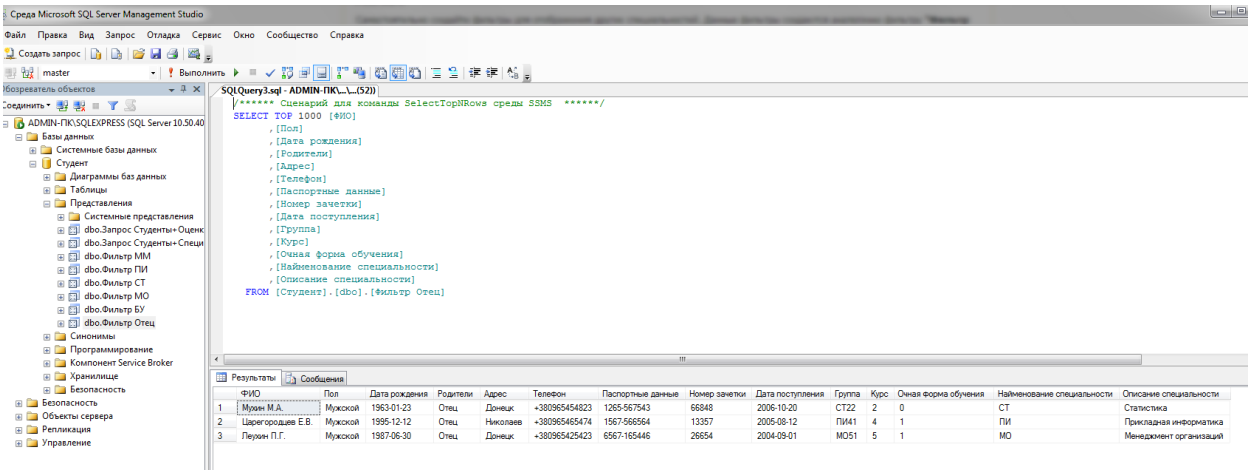


Рис. 4.17

Створіть фільтри для відображення студентів з іншими варіантами батьків. Дані фільтри створюються аналогічно фільтру «**Фильтр Отец**» (дивися вище). Єдиною відмінністю є умова відбору, що накладається на поле «**Родители**», воно повинно бути не

«= **Отец**», а «= **Мать**», «= **Отец**, **Мать**» або «=« **Нет**». Під час збереження фільтрів задаємо їхні імена відповідно їх умовам відбору, тобто «**Фильтр Мать**», «**Фильтр Отец и Мать**» або «**Фильтр Нет родителей**». Перевірте створені фільтри на працездатність.

Нарешті створимо фільтри для відображення студентів очної та заочної форми навчання. Почнемо з очної форми навчання. Створіть новий запит і додайте в нього запит «**Запрос Студенты + Специальности**» (рис. 4.18).

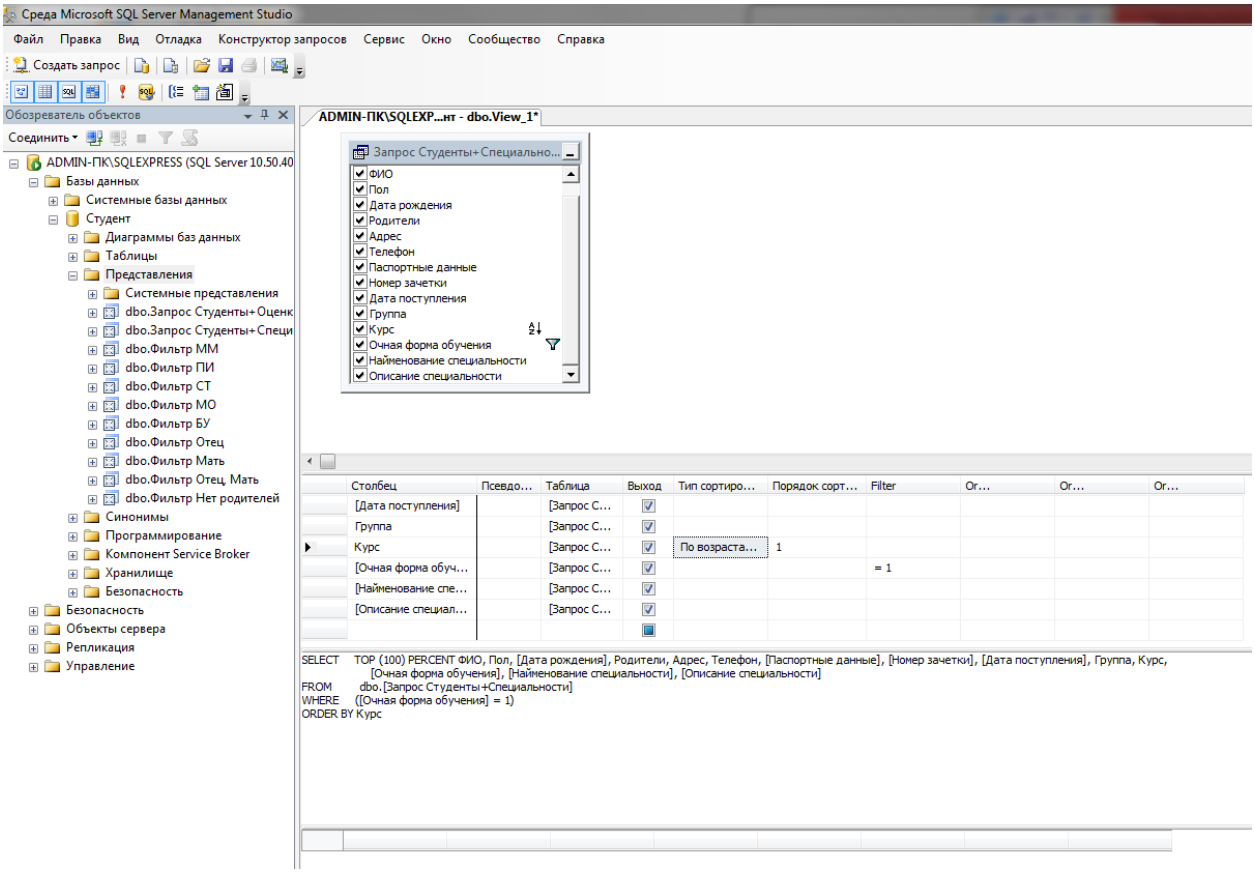


Рис. 4.18

У таблиці видимі поля в стовпці «**Filter**», у рядку для поля «**Очная форма обучения**» встановіть умову відбору рівну «**= 1**»

***Зауваження*:** Поле «**Очна форма обучения**» є логічним полем, воно може приймати значення або «**True**» (Істина), або «**False**» (Брехня). Як синонімів цих значень в «**Microsoft SQL Server 2008**» можна використовувати 1 і 0 відповідно.

Встановіть сортування за зростанням, по полю курс, задавши в рядку для цього поля, в стовпці «**Тип сортировки**», значення «**По возростанию**».

Перевірте роботу фільтра, виконавши його.

Закрийте вікно конструктора запитів. Збережіть фільтр під ім'ям «**Фильтр очная форма обучения**» (рис. 4.19).

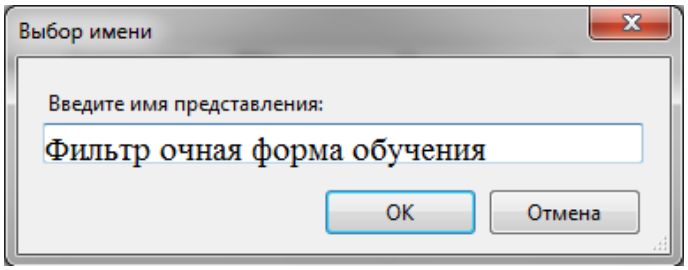


Рис. 4.19

Після появи фільтра «**Фильтр очная форма обучения**» в провіднику об'єктів виконайте фільтр поза вікном конструктора запитів (рис. 4.20).

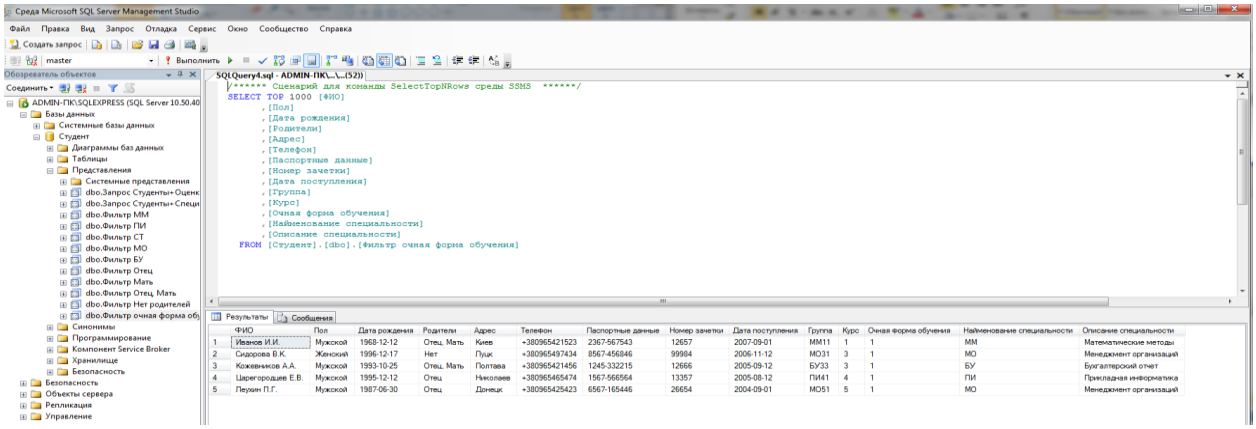


Рис. 4.20

Самостійно створіть фільтр для відображення студентів заочної форми навчання. Даний фільтр створюється точно також як і фільтр «**Фильтр очная форма обучения**». Єдиною відмінністю є умова відбору, що накладається на поле «**Очная форма обучения**», воно повинно бути не «**= 1**», а «**= 0**». Під час збереження фільтра задайте його ім'я як «**Фильтр заочная форма обучения**». Перевірте створений фільтр на працездатність.

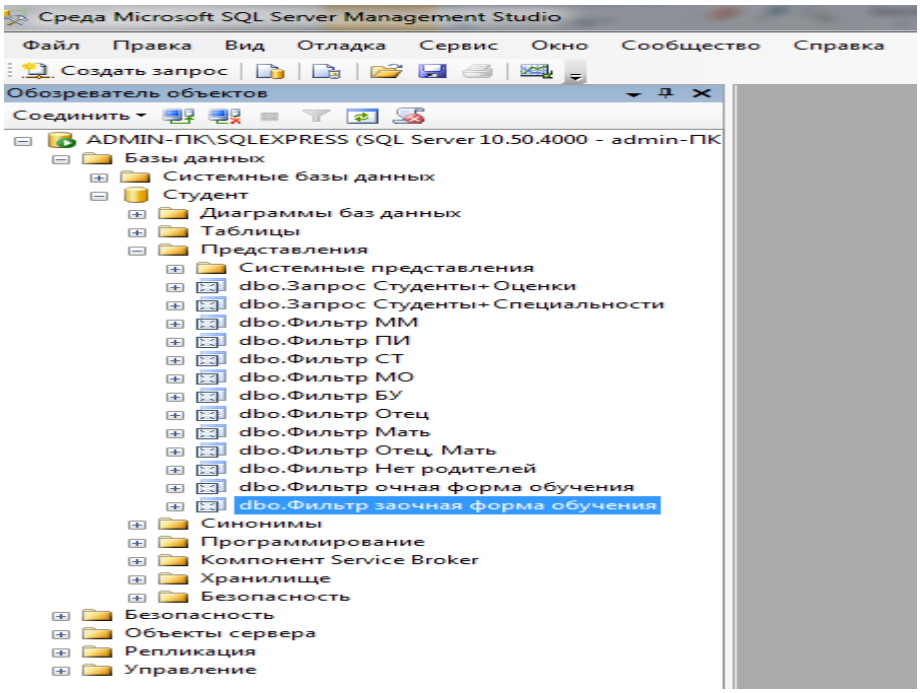


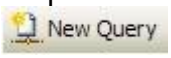
Рис. 4.21

Лекція 5: Створення динамічних запитів за допомогою збережених процедур

Мета: вивчити процес створення динамічних запитів за допомогою збережених процедур.

Збережена процедура – SQL запит, який має параметри, тобто він виконується як звичайна процедура (ми задаємо її ім'я і передаємо в збережену процедуру значення параметрів.). Залежно від значення параметрів процедури ми отримуємо той чи інший результат запиту.

***Зауваження*:** У SQL сервері збережені процедури реалізують динамічні запити, що виконуються на стороні сервера.

Розглянемо створення збережених процедур за допомогою команд мови SQL. Щоб відобразити збережені процедури робочої БД панелі «Object Explorer» потрібно виділити пункт «Хранимые процедуры». Щоб створити нову процедуру за допомогою команд мови SQL потрібно клацнути ЛКМ по кнопці 

на панелі інструментів. У робочій області вікна сервера з'явиться вкладка **SQLQuery1.sql,** де потрібно набрати код нової процедури, який має наступний синтаксис:

```
REATE PROCEDURE <Имя процедуры>
[@<Параметр1> <Тип1>[=<Значение1>],
@<Параметр2> <Тип2>[=<Значение2>], …]
[WITH ENCRYPTION]
AS <Команды SQL>
```

Тут:

- **Имя процедуры** – ім'я створюваної збереженої процедури.
- **Параметр 1, Параметр 2, ...** – параметри передаються в процедуру.
- **Значение 1, Значение 2 ...** – значення параметрів за замовчуванням.
- **Тип 1, Тип 2, ...** – типи даних параметрів.
- **WITH ENCRYPTION** – включає шифрування даних.
- **Команды SQL** – SQL запит, який виконується при запуску процедур.

***Зауваження*:** SQL запит включає в себе параметри, якщо параметри порівнюються з якимись полями або виразами, то вони повинні мати точно такий же тип даних як ці поля або вирази.

***Зауваження*:** Після створення процедура поміщається в розділ **Хранимые процедуры** поточної БД на панелі «**Object Explorer**». Якщо двічі клацнути по процедурі ЛКМ, то вона відкриється для редагування на вкладці «**SQL Запрос**».

Щоб подивитися інформацію щодо збережену процедуру необхідно виконати команду:

```
EXEC SP_HELPTEXT <Имя процедуры>
```

Збережені процедури можуть бути запущені наступною командою

```
EXEC <Имя процедуры> [<параметр 1>, <параметр 2>, ...]
```

Тут:

- **<Имя процедуры>** – ім'я виконуваної процедури.
- **Параметр 1, параметр 2, ...** – значення параметрів.

Приклад: Створення збереженої процедури, яка виводить ім'я студентів, у яких середній бал більше заданої величини:

```
CREATE PROCEDURE СрБАЛЛ

X Real

AS

SELECT *

FROM Студент

WHERE

(Оценка 1+ Оценка 2 + Оценка 3) / 3>X
```

Команда виклику наведеної вище процедури виглядає наступним чином:

```
EXEC СрБАЛЛ 4
```

Команда виводить всіх студентів, у яких середній бал більше 4.

Самостійна робота 5: Збережені процедури

Мета: навчитися працювати з збереженими процедурами.

Перейдемо до створення збережених процедур. Для роботи з збереженими процедурами в оглядачі об'єктів необхідно виділити папку «**Програмирование / Хранимые процедуры**» бази даних «**Студент**» (рис. 5.1).

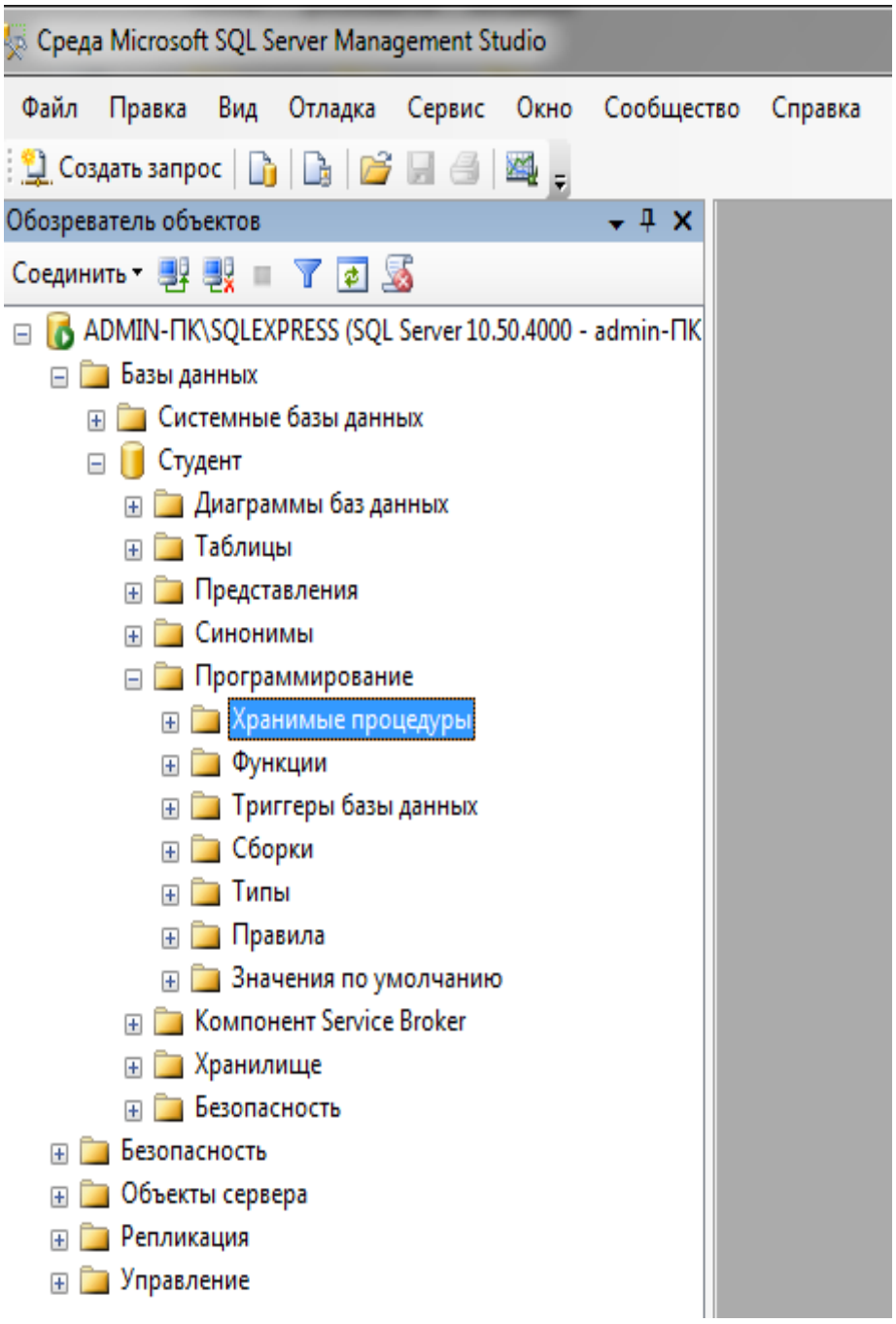


Рис. 5.1

Створюємо процедуру, яка обчислює середнє трьох чисел. Для створення нової збереженої процедури клацніть **ПКМ** по папці «**Хранимые процедуры**» і в меню виберіть пункт «**Создать хранимую процедуру**». З'явиться вікно коду нової збереженої процедури (рис. 5.2).

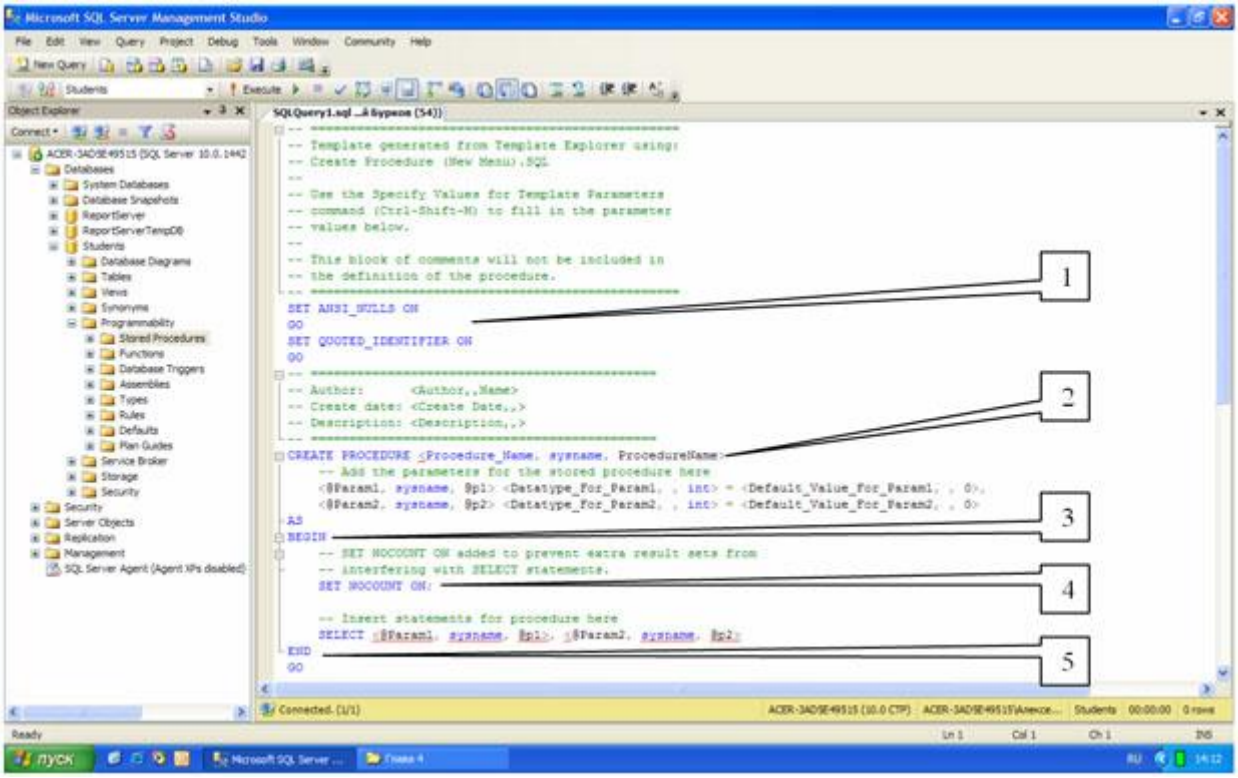


Рис. 5.2

Збережена процедура має наступну структуру:

1. Область налаштування параметрів синтаксису процедури дозволяє налаштовувати деякі синтаксичні правила, використовувані під час набору коду процедури. У нашому випадку це:
    - **SET ANSI_NULLS ON** – включає використання значень NULL в кодуванні ANSI.
    - **SET QUOTED_IDENTIFIER ON** – включає можливість використання подвійних лапок для визначення ідентифікаторів.
2. Область визначення імені процедури (**Procedure_Name**) і параметрів переданих в процедуру **(@ Param1, @ Param2**). Визначення параметрів має наступний синтаксис:
   @ <Ім'я параметра> <Тип даних> = <Значення за замовчуванням>
   Параметри розділяються між собою комами.
3. Початок тіла процедури позначається службовим словом «**BEGIN**».
4. Тіло процедури містить команди мови програмування запитів T-SQL.
5. Кінець тіла процедури позначається службовим словом «**END**».

***Зауваження*:** У коді зеленим кольором виділяються коментарі. Вони не обробляються сервером і виконують функцію пояснень до коду. Рядки коментарів починаються з підрядка «–». Далі в коді, ми не будемо відображати коментарі, вони будуть згорнуті. Зліва від розділу з коментарями стоятиме знак «+», клацнувши по якому можна розгорнути коментар.

Наберемо код процедури яка обчислює середнє трьох чисел (рис. 5.3).

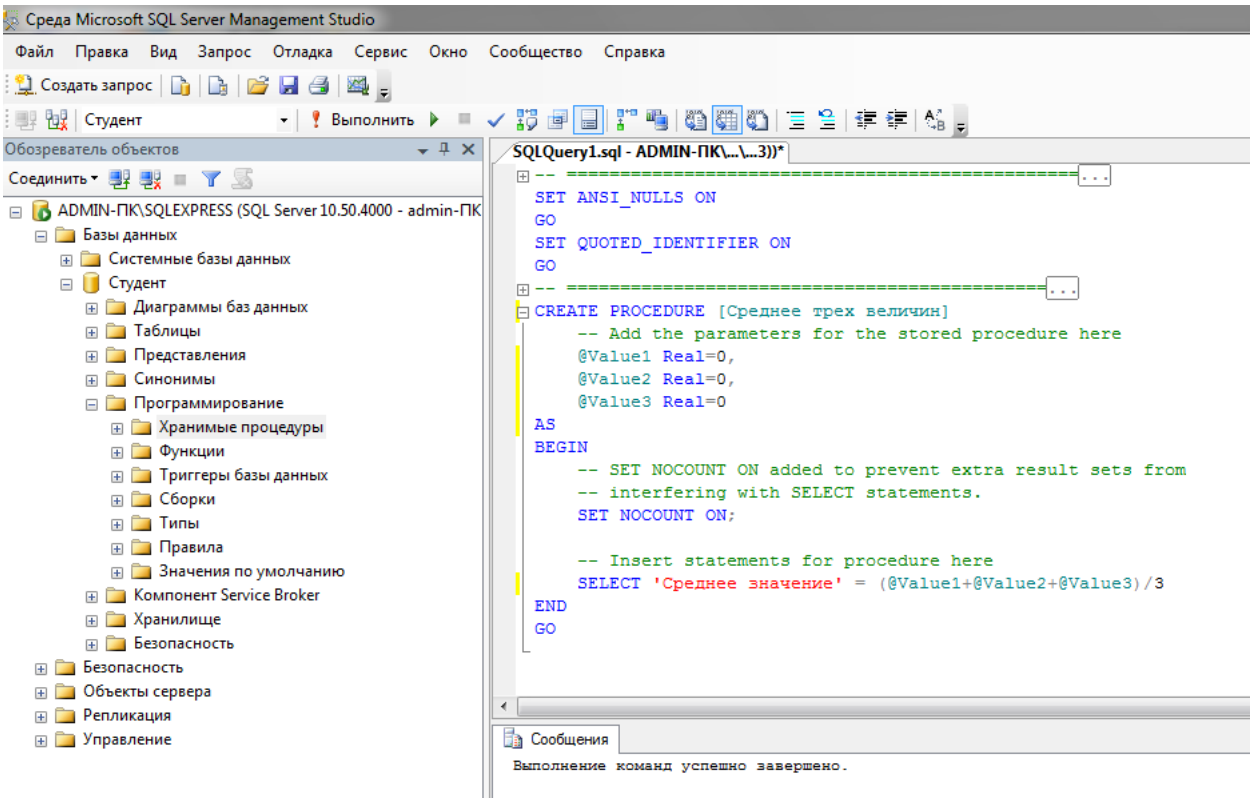


Рис. 5.3

Розглянемо код даної процедури більш докладно:

1. **CREATE PROCEDURE [Середнє трьох величин]** – визначає ім'я створюваної процедури як «Среднее трех величин».
2. **@ Value1 Real = 0, @ Value2 Real = 0, @ Value3 Real = 0** – визначають три параметри процедури **Value1, Value2 і Value3.** Даним параметрам можна привласнити дробові числа (Тип даних Real), значення за замовчуванням рівні 0.
3. **SELECT 'Середнє значення' = (@ Value1 + @ Value2 + @ Value3) / 3** – обчислює середнє і виводить результат з підписом «Середнє значення».

Решта фрагменти коду розглянуті вище.

Для створення процедури, виконаємо вищеописаний код, натиснувши кнопку Execute (Виконати) на панелі інструментів. У нижній частині вікна з кодом з'явитися повідомлення «**Выполнение команд успешно завершено.**». Закрийте вікно з кодом, клацнувши мишею по кнопці закриття , розташованої у верхньому правому куті вікна з кодом процедури.

Перевіримо працездатність створеної збереженої процедури. Для запуску збереженої процедури необхідно створити новий порожній запит, натиснувши на кнопку New Query (Новий запит) на панелі інструментів. У вікні з порожнім запитом наберіть команду **EXEC [Среднее трех величин] 1, 7, 9** і натисніть кнопку Execute на панелі інструментів.

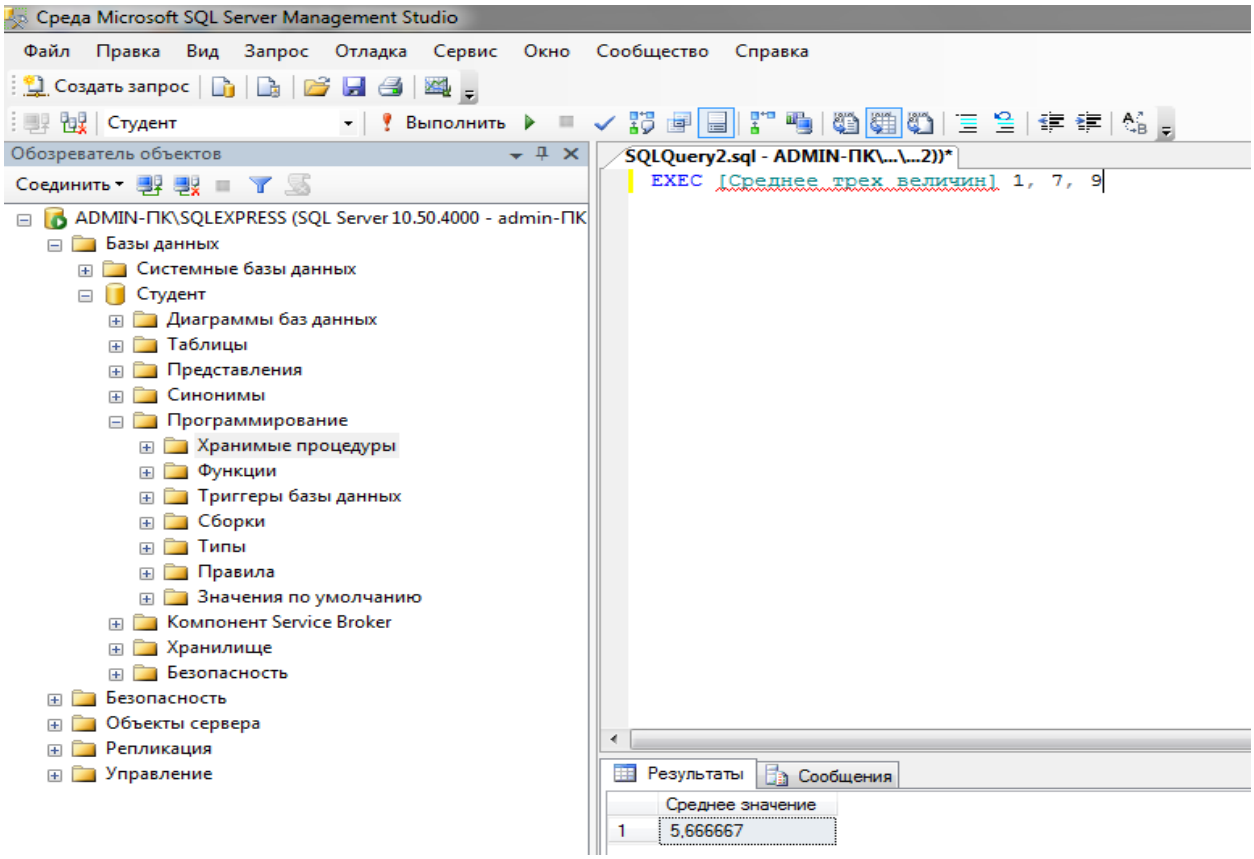


Рис. 5.4

У нижній частині вікна з кодом з'явиться результат виконання нової збереженої процедури: Среднее значение 5,66667 (рис. 5.4).

Тепер створимо збережену процедуру для відбору студентів з таблиці студенти по їх «ФИО». Для цього створіть нову збережену процедуру, як це описано вище, і наберіть код нової процедури (рис. 5.5).

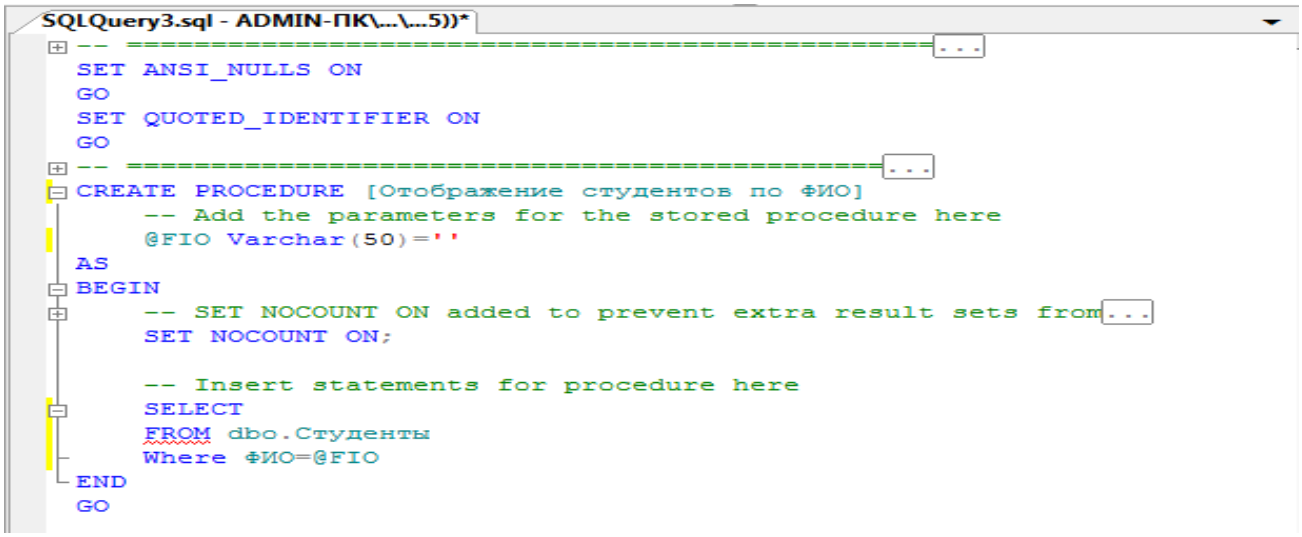


Рис. 5.5

Розглянемо код процедури «Отображение студентов по ФИО» більш докладно:

1. **CREATE PROCEDURE [Отображение студентов по ФИО]** – визначає ім'я створюваної процедури як «Відображення студентів по ПІБ»;

2. **FIO Varchar (50) = ‘‘** – визначають єдиний параметр процедури **FIO**. Параметру можна привласнити текстові рядки змінної довжини, довгою до 50 символів (Тип даних Varchar (50)), значення за замовчуванням рівні порожньому рядку;
3. **SELECT * FROM dbo.Студенты WHERE ФИО= @ FIO** – відобразити всі поля (*) з таблиці студенти (dbo.Студенты), де значення поля ФИО дорівнює значенню параметра **FIO (ФИО = @ FIO)**.

Виконаємо вищеописаний код і закриємо вікно з кодом, як описано вище.

Перевіримо працездатність створеної збереженої процедури. Створіть новий порожній запит. У вікні з порожнім запитом наберіть команду **EXEC [Отображение студентов по ФИО] ‘Иванов И.И.’** і натисніть кнопку 

на панелі інструментів.

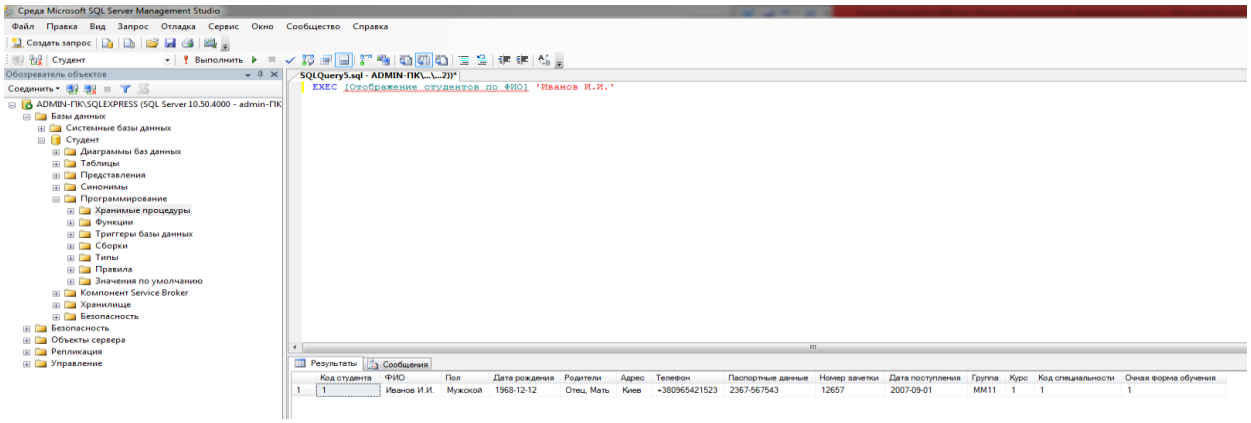


Рис. 5.6

У нижній частині вікна з кодом з’явиться результат виконання процедури «**Отображения студентов по ФИО**» (рис. 5.6).

Тепер перейдемо до більш складної задачі – відобразити студентів, у яких середній бал вище заданого. Створіть нову збережену процедуру і наберіть код нової процедури (рис. 5.7).

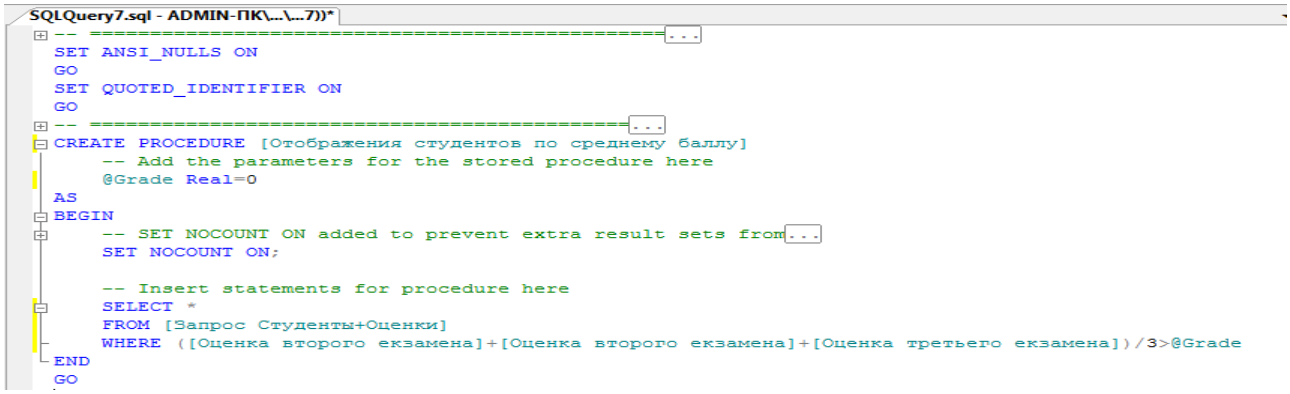


Рис. 5.7

Розглянемо код процедури «**Отображение студентов по среднему баллу**» більш докладно:

1. **CREATE PROCEDURE [Отображение студентов по среднему баллу]** – визначає ім'я створюваної процедури як «Відображення студентів за середнім балом».
2. **Grade Real = 0** – визначають параметр процедури **Grade**. Параметру можна привласнити дробові числа (Тип даних Real), значення за замовчуванням рівні 0.
3. **SELECT * FROM [Запрос Студенты + Оценки] WHERE ([Оценка первого экзамена] + [Оценка второго экзамена] + [Оценка третьего экзамена]) / 3>Grade** – відобразити всі поля (*) із запиту **«Запрос Студенты + Оценки» (Запрос Студенты + Оценки), де середній бал більше ніж значення параметра Grade (([Оценка первого экзамена] + [Оценка второго экзамена] + [Оценка третьего экзамена]) / 3>Grade).**

Виконаємо вищеописаний код і закриємо вікно з кодом, як описано вище. Перевіримо, як працює запит, описаний вище. Для цього, створіть новий запит і в ньому наберіть команду **EXEC [Отображение студентов по среднему баллу] 3.5** і виконайте її (рис. 5.8).

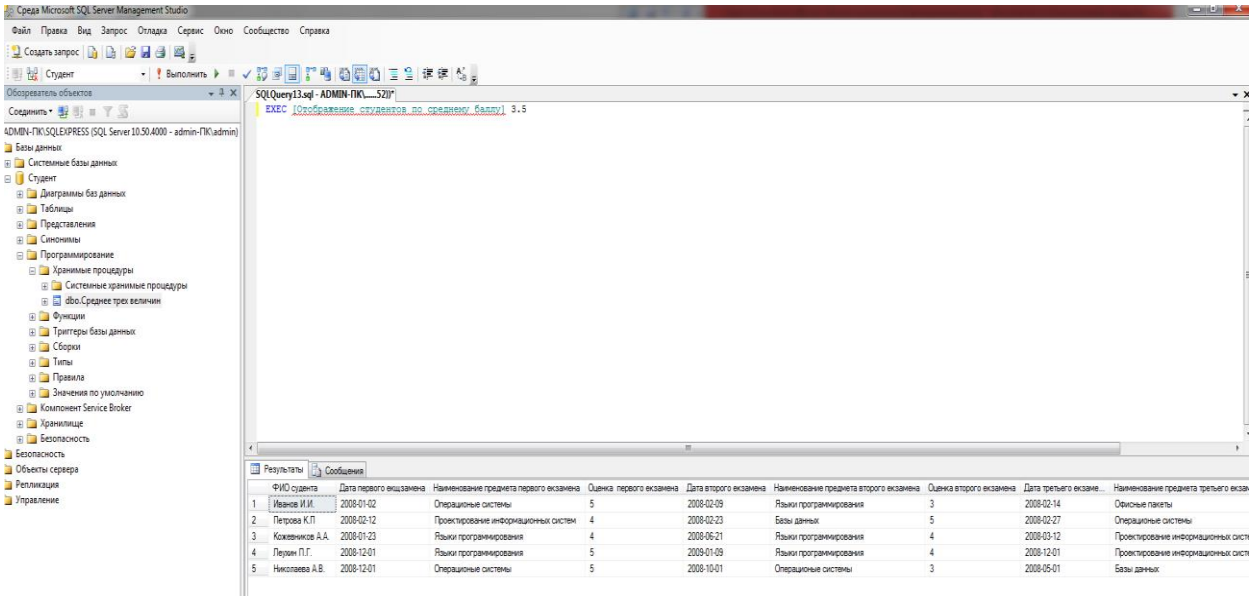


Рис. 5.8

У нижній частині вікна з кодом з'явиться результат виконання процедури «Отображение студентов по среднему баллу «.

На закінчення вирішимо більш складне завдання – відображення студентів старше заданого віку. При чому вік буде автоматично обчислюватися в залежності від дати народження.

Створимо нову збережену процедуру і наберемо код нової процедуру (рис. 5.9).

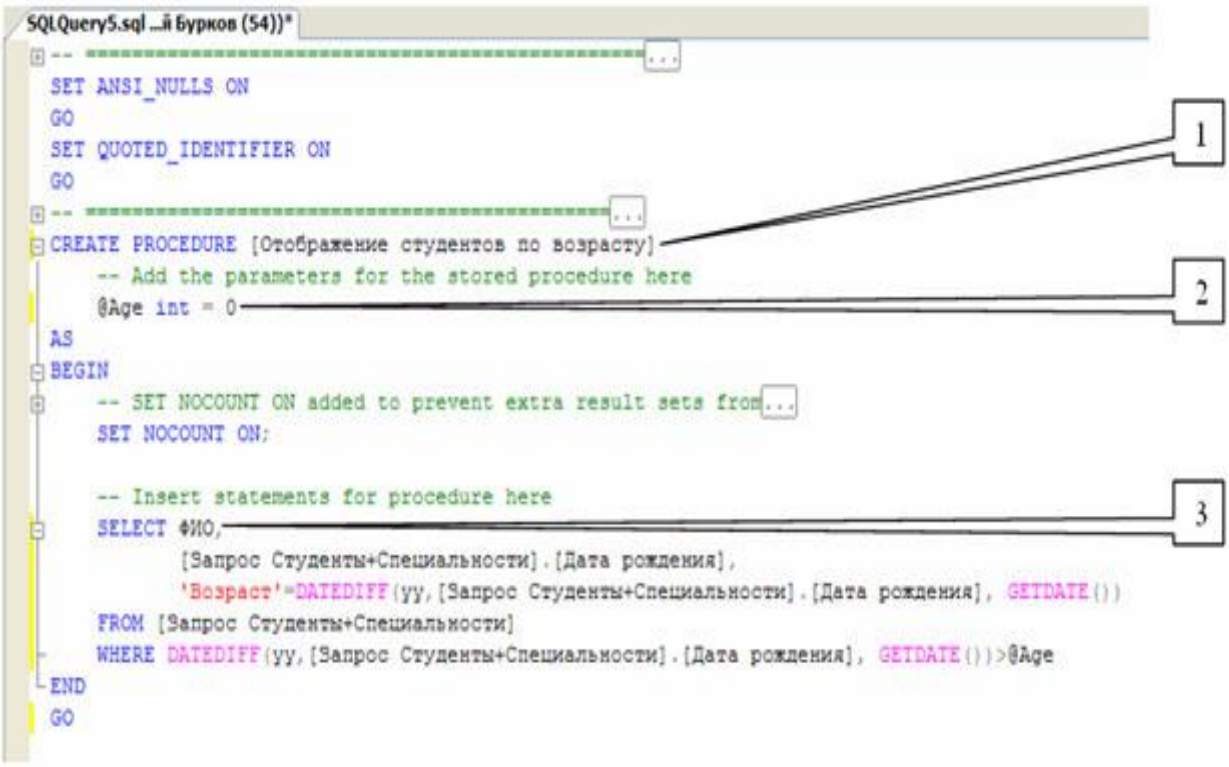


Рис. 5.9

Розглянемо код створюваної процедури «Отображение студентов по возрасту» більш докладно:

1. **CREATE PROCEDURE [Отображение студентов по возрасту]** – визначає ім'я створюваної процедури як «Відображення студентів за віком».
2. **Age Int = 0** – визначають параметр процедури **Age**. Параметру можна привласнити цілі числа (Тип даних int), значення за замовчуванням рівні 0.
3. **ФИО, [Запрос Студенты + Специальности]. [Дата рождения], 'Возрост' = DATEDIFF (yy, [Запрос Студенты + Специальности]. [Дата рождения], GETDATE ())** – відображає із запиту «**Запросу Студенты + Специальности**» (FROM [Запрос Студенты + Специальности]) поля «**ФИО**» і «**Дата рождения**» ([Запрос Студенты + Специальности]. [Дата рождения]), а також відображає вік студента **(Возраст)** в роках (yy), обчислений виходячи з його дати народження та поточної дати **(DATEDIFF (yy, [Запрос Студенты + Специальности]. [Дата рождения], GETDATE ()))**. Більше того, виводяться Студенты вік яких більше певного в параметрі «**Age**» **(DATEDIFF (yy, [Запрос Студенты + Специальности]. [Дата рождения], GETDATE ())> @ Age).**

***Зауваження*:** Вбудована функція **DATEDIFF** обчислює кількість періодів між двома датами, має наступний синтаксис: **DATEDIFF (<період>, <початкова дата>, <кінцева дата>)**

Виконаємо код запиту «**Отображения студентов по возросту**», а потім закриємо вікно з кодом, як описано вище. Перевіримо, як працює запит. Для цього, створимо новий запит і в ньому наберемо команду **EXEC [**Отображение студентов по возрасту**]** 26 і виконайте її (рис. 5.10).

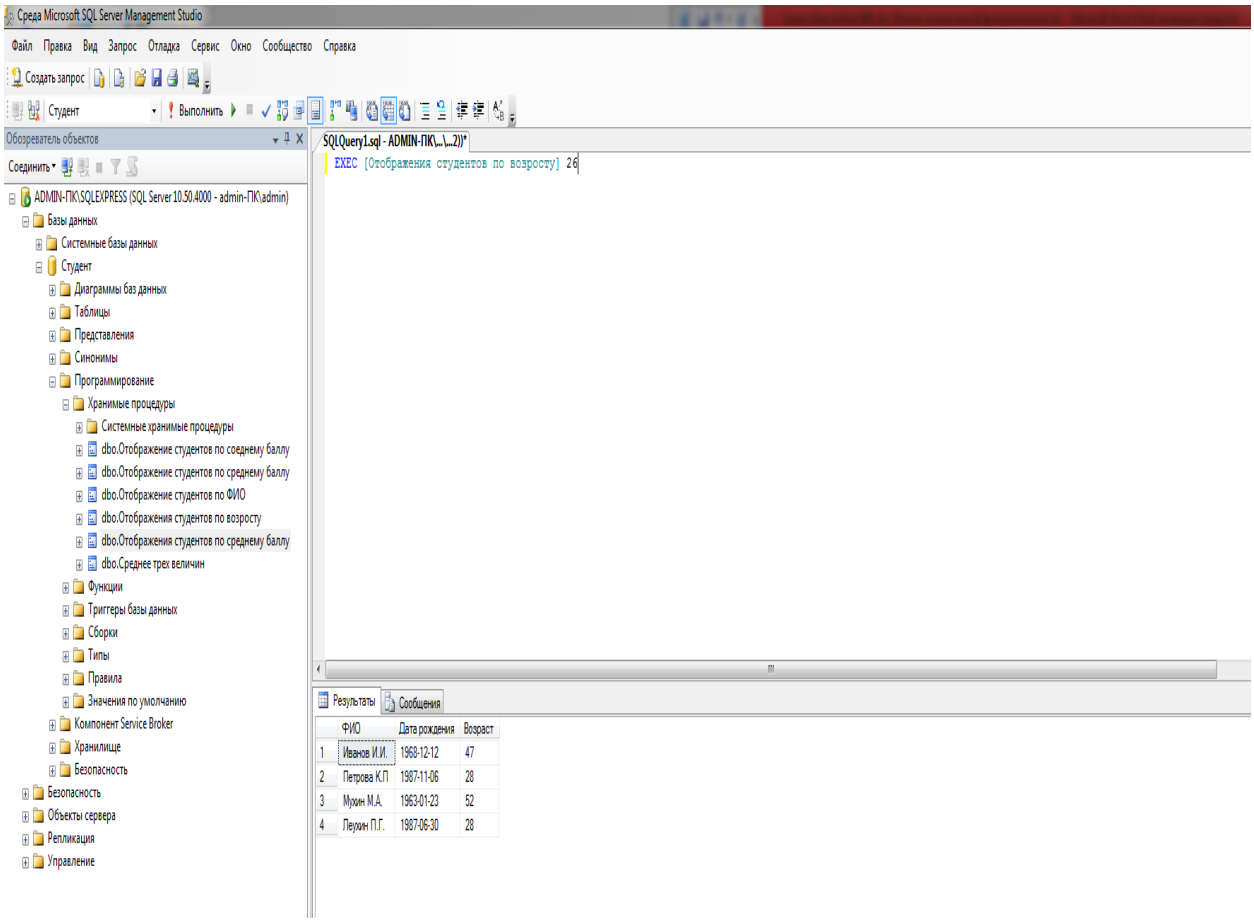

Среда Microsoft SQL Server Management Studio
Файл Правка Вид Запрос Отладка Сервис Окно Сообщество Справка
Создать запрос
Студент
Выполнить
Обозреватель объектов
Соединить
ADMIN-ПК\SQLEXPRESS (SQL Server 10.50.4000 - admin-ПК\admin)
Базы данных
Системные базы данных
Студент
Диаграммы баз данных
Таблицы
Представления
Синонимы
Программирование
Хранимые процедуры
Системные хранимые процедуры
dbo.Отображение студентов по соседнему баллу
dbo.Отображение студентов по среднему баллу
dbo.Отображение студентов по ФИО
dbo.Отображения студентов по возросту
dbo.Отображения студентов по среднему баллу
dbo.Среднее трех величин
Функции
Триггеры базы данных
Сборки
Типы
Правила
Значения по умолчанию
Компонент Service Broker
Хранилище
Безопасность
Безопасность
Объекты сервера
Репликация
Управление
SQLQuery1.sql - ADMIN-ПК\...\...2)*
EXEC [Отображения студентов по возросту] 26
Результаты
Сообщения
ФИО
Дата рождения
Возраст
1 Иванов И.И. 1968-12-12 47
2 Петрова К.П. 1987-11-06 28
3 Мухин М.А. 1963-01-23 52
4 Пехин П.Г. 1987-06-30 28

Рис. 5.10

Тест 2 до Лекція 3, Лекція 4, Лекція 5

№ з/п	*Питання*	*Варіанти відповідей*
1.	**Яка характеристика поля використовується для звернення до поля?**	1. Значення поля. 2. Тип даних поля. 3. Ім'я поля.
2.	**Який тип даних в Microsoft SQL Server +2008 використовується для зберігання дробових чисел?**	1. Float. 2. Smallint. 3. Tinyint.
3.	**Вкажіть функцію в Microsoft SQL Server 2008, яка виводить крок збільшення поля лічильника в таблиці:**	1. Ident_ incr. 2. Col_length. 3. Identincr.
4.	**Яка команда в Microsoft SQL Server 2008 заповнює таблиці?**	1. Insert. 2. Select. 3. Create table. 4. Use.
5.	**Яка команда в Microsoft SQL Server 2008 робить активною зазначену базу даних?**	1. Select. 2. Create table. 3. Use. 4. Delete.
6.	**Який тип даних в Microsoft SQL Server +2008 використовується для зберігання послідовності нулів та одиниць?**	1. Bbit. 2. Binary. 3. Decimal.

7.	**Вкажіть функцію в Microsoft SQL Server 2008, яка видаляє з рядка пробіли праворуч:**	1. Lowerc. 2. Upper. 3. ltrim. 4. Rtrim.
8.	**Розділ команди select, який дозволяє в кінці результатів виконання запиту вивести деякі підсумкові обчислення за запитом:**	1. Order by. 2. Compute. 3. Group by.
9.	**Вкажіть функцію в Microsoft SQL Server 2008, яка виводить знак числа:**	1. Degrees. 2. Sign. 3. Abc.
10.	**Параметр percent команди select в Microsoft SQL Server +2008:**	1. Визначає відсоток від загального числа записів у таблиці, який обробляється. 2. Визначає, яка кількість записів у таблиці обробляється. 3. Визначає, яка кількість записів у таблиці не обробляється.
11.	**Збережена процедура – це**	1. Sql запит, який не має параметрів, підставляється всередину sql коду. 2. Sql запит, збережений на стороні сервера. 3. Sql запит, збережений на стороні клієнта.
12.	**Вкажіть команду в SQL Server 2008, що забезпечує перегляд інформації про збереженій процедурі:**	1. Exec. 2. Exec sp_helptext. 3. Create procedure.

Методичний посібник СУБД Microsoft SQL Server, для здобувачів освітньо-кваліфікаційного рівня фаховий молодший бакалавр спеціальності 122 «Комп'ютерні науки» денної форми навчання /уклад. О.В. Присада – Ковель: КПЕК Луцького НТУ, 2023. – 64

Комп'ютерний набір і верстка: **Олександр ПРИСАДА**

Редактор: **Леся ПРОКОПЧУК**

Підп. до друку «___»_____2023. Формат 60х84/16. папірофіс.

Гарн.Таймс. Ум.друк. арк. ____ Обл.-вид. арк. ___ Зам. ___

Тираж ____ прим.

ВСП «КПЕФК ЛНТУ»

45000 м. Ковель, вул. Заводська, 23

Друк – ВСП «КПЕФК ЛНТУ