УНИВЕРСИТЕТ ИТМО

Филиппов А. Н.

ПРИМЕНЕНИЕ ЯЗЫКА ЗАПРОСОВ SQL В САПР ТП

Санкт-Петербург

2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
УНИВЕРСИТЕТ ИТМО

А.Н. Филиппов

ПРИМЕНЕНИЕ ЯЗЫКА ЗАПРОСОВ SQL В САПР ТП

Учебное пособие

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург
2017

А.Н. Филиппов. Применение языка запросов SQL в САПР ТП/ Учебное пособие // СПб: Университет ИТМО, 2017. – с. 54

Настоящее пособие предназначено для студентов специализации “Технологии приборостроения”. В пособии изложены наиболее важные темы, связанные с описанием методов организации данных и программирования алгоритмов с применением реляционной СУБД, как основы проектирования баз данных технологического назначения.

Адресовано студентам высших учебных заведений, обучающихся по направлению подготовки 09.03.01– Информатика и вычислительная техника.

Рекомендовано к печати Ученым советом мегафакультета КТ и У 14.03.2017г, протокол № 5

В 2009 году Университет стал победителем многоэтапного конкурса, в результате которого определены 12 ведущих университетов России, которым присвоена категория «Национальный исследовательский университет». Министерством образования и науки Российской Федерации была утверждена программа его развития на 2009–2018 годы. В 2011 году Университет получил наименование «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Содержание

Оглавление

СОКРАЩЕНИЯ, ПРИНЯТЫЕ В ПОСОБИИ

DDL	Data Direction Language (Операторы управления данными)
DML	Data Manipulation Language (Операторы манипулирования данными)
DCL	Data Control Language (Операторы определения доступа к данным)
TCL	Transaction Control Language (Операторы управления транзакциями)
NoSQL	Not only SQL
SQL	Structure Query Language (Язык структурных запросов)
No RDBMS	Нереляционная база данных
БД	База данных
БЗ	База знаний
ИМ	Интерфейсный модуль
ИПС	Информационно-поисковая система
САПР ТП	Система автоматизации проектирования технологических процессов

Введение

В настоящее время при разработке CAD/CAM систем технической и технологической подготовки производства получили широкое применение реляционные СУБД. И, несмотря на прогресс NoSQL-технологий, реляционные базы данных по-прежнему занимают ведущее место в САПР ТП.

Настоящее учебное пособие предназначено для выполнения лабораторных работ по дисциплинам “Системы баз данных” по направлению 09.03.01 – Информатика и вычислительная техника. В работе описаны инструментальные программные средства и методы построения структур данных при создании САПР ТП с применением реляционной СУБД MicroSoft SQL. Однако наиболее полно освоить данный курс студент может только, используя все пособия по данной дисциплине, входящие в комплект, а также литературу, список которой приведен в заключительной части пособия.

Лабораторные работы могут выполняться с использованием СУБД Microsoft SQL, а также любой СУБД, поддерживающей стандарт ANSI/ISO SQL2. Язык SQL - Structure Query Language (Язык структурных запросов) является одним из доминирующих языков, используемых в реляционных базах данных. Родоначальником серии SQL Server и его основой является язык запросов SQL. Данный язык был создан компанией IBM в начале 1970г. Изначально он назывался SEQVEL (Structured English Query Language). В основу языка SQL, используемого в SQL Server, легла разновидность языка T-SQL (Transact - SQL). В качестве стандартного языка баз данных он принят такими организациями, как American National Standards Insnitute – ANSI (Американский национальный институт стандартов) и International Standards Organization – ISO (Международная организация стандартов).

SQL является непроцедурным языком, что позволяет ему обрабатывать наборы записей и автоматизировать процесс управления данными. SQL не требует от пользователя спецификации метода доступа к данным, пользователь вообще не должен описывать, как должна выполняться та или иная операция над данными, он просто описывает, что должно быть сделано с данными, предоставляя право системе решать, как это сделать.

В пособии дано краткое описание языка запросов SQL, приведены четыре примера лабораторных работ, выполненных с применением SQL.

1. Краткое описание SQL

SQL[1] включает в себя четыре основные составляющие:

- Операторы управления данными Data Direction Language (DDL)
- Операторы манипулирования данными Data Manipulation Language DML)
- Операторы определения доступа к данным *Data Control Language* (DCL)
- Операторы управления транзакциями Transaction Control Language (TCL)

1.1. Операторы DDL

Функции языков DDL определяются первым словом в предложении (часто называемом запросом), которое почти всегда является глаголом. В случае с SQL эти глаголы**:**

- CREATE – создать;

- ALTER – изменить;

- DROP – удалить.

1.1.1. Генерация уникального идентификатора для новых строк при создании БД

Синтаксис:

```
--Simple CREATE TABLE Syntax (common if not using options)
CREATE TABLE
    [ database_name . [ schema_name ] . | schema_name . ]
table_name
    ( { <column_definition> } [ ,...n ] )
[ ; ]
```

Подробно **см.** [4**]**

Пример применения

```
/* CREATE   MySQL */
CREATE TABLE employees (
     id INTEGER NOT NULL AUTO_INCREMENT,
     first_name CHAR(50) NOT NULL,
     last_name CHAR(75) NOT NULL,
     dateofbirth DATE NULL,
     PRIMARY KEY (id));
```

[1]см. приложение Синтаксис SQL

```
/* CREATE  Microsoft SQL */
     CREATE TABLE employees (
     id INTEGER PRIMERY KEY IDENTITY,
     first_name CHAR(50) NOT NULL,
     last_name CHAR(75) NOT NULL,
     dateofbirth DATE NULL);
```

1.1.2. Изменение определения существующей таблицы

Операторы **ALTER, DROP.**

Синтаксис:

```
ALTER TABLE [ database_name . [ schema_name ] . | schema_name .
] table_name
```

Подробно см. [5]

Пример применения

- **Добавить колонку status**
  ```
  ALTER TABLE ADD status TINYINT;
  ```
- **Удалить колонку dateofbirth**
  ```
  ALTER TABLE DROP COLUMN dateofbirth;
  ```
- **Удалить таблицу employees**
  ```
  DROP TABLE employees.
  ```

1.2. Язык манипулирования данными (DML)

Функции языков DML определяются первым словом в предложении (часто называемом запросом), которое почти всегда является глаголом. В случае с SQL эти глаголы:

- **INSERT** – вставить;
- **SELECT** - выбрать;
- **UPDATE** - обновить;
- **DELETE** – удалить.

1.2.1. Добавить строки в таблицу INSERT (DML)

INSERT— оператор языка SQL, который позволяет добавить строки в таблицу, заполняя их значениями. Значения можно вставлять перечислением с помощью слова values и перечислив их в круглых скобках через запятую или оператором select.

Синтаксис

```
INSERT INTO table (column1, [column2, ... ]) VALUES
(value1a, [value1b, ...]), (value2a, [value2b, ...]), ...
```

Пример применения

```
INSERT INTO employees
(first_name , last_name , dateofbirth)
VALUES ('Алексей', 'Иванов', '1987-05-22'),
('Пётр', 'Зайцев', '1966-05-02'),
('Пётр', 'Васильев', '1962-09-14'),
('Валерий', 'Иванов', '1964-12-29');
```

1.2.2. Выбрать из таблицы SELECT (DML)

Оператор **SELECT** является одним из основных операторов языка **SQL.** Именно с его помощью происходит выборка значений, хранящихся в базе данных. В структуру запроса оператора **SELECT** могут быть включены многие дополнительные операторы: уточняющие условие выборки, производящие группировку, сортировку выходных значений и т.д.

Синтаксис

```
SELECT [ALL | DISTINCT] select_expr
[FROM table_references]
[WHERE where_condition]
[GROUP BY {col_name | expr | position}
[ASC | DESC]
[HAVING where_condition]
[ORDER BY {col_name | expr | position}
[ASC | DESC]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

Оператор **SELECT** имеет несколько опциональных параметров.

WHERE - определяет критерии выборки.

GROUP BY - группирует строки по общему значению одного или нескольких полей.

HAVING- выбирает из групп определённых в условии **GROUP BY**. При отсутствии последнего работает аналогично **WHERE.**

ORDER BY - определяет порядок сортировки строк.

Пример применения

- Выбрать все записи из таблицы employees.

```
SELECT * FROM employees;
```

- Выбрать поля **first_name, last_name, dateofbirth** из таблицы **employees** и отсортировать по убыванию поля **last_name.**

```
SELECT first_name, last_name, dateofbirth   FROM employees
ORDER BY last_name DESC;
```

Таблица 1.1.Список сотрудников

first_name	last_name	dateofbirth
Петр	Зайцев	1966-02-02
Петр	Васильев	1962-09-14
Валерий	Иванов	1964-12-29
Алексей	Иванов	1987-05-22

SELECT DISTINCT (DML)

Пример применения

```
SELECT DISTINCT first_name FROM employees;
```

DISTINCT исключает возможность вывода одинаковых строк в выборке

Таблица 1.2.

first_name
Петр
Валерий
Алексей

SELECT LIKE (DML)

Условие **LIKE** применяется для задания поиска по определенному шаблону (паттерну).

Синтаксис:

```
SELECT имя_колонки FROM имя_таблицы WHERE имя_колонки LIKE
паттерн
```

Пример применения

- Выбирает всех студентов, имена которых начинаются на букву ‘P%’

```
SELECT * FROM STUDENT first_name LIKE ‘P%’
```

- Выбирает всех студентов, имена которых кончаются на комбинацию букв ‘ов’

```
SELECT * FROM STUDENT last_name LIKE ‘%ов’
```

- Выбирает студентов, имена которых содержат комбинацию букв ‘%ре%’

```
SELECT * FROM STUDENT first_name LIKE ‘%ре%’
```

SELECT (Оператор BETWEEN…AND) (DML)

Оператор **BETWEEN**…**AND** выбирает данные в определенном диапазоне, включая края.

Синтаксис:

```
SELECT имя_колонки FROM имя_таблицы WHERE имя_колонки
BETWEEN значение_1 AND значение_2
```

Значения, отмечающие края диапазона, могут быть числовыми, текстовыми или в формате даты.

Пример применения

Выбрать из таблицы номера зачеток и оценки всех студентов, оценки которых в диапазоне от 3 до 5:

```
SELECT SN, OCENKA FROM USP WHERE OCENKA BETWEEN 3 AND 5
```

Внесение большого количества строк в таблицу из одной или более таблиц (DML).

Команда **INSERT ... SELECT**

Синтаксис:

```
INSERT [LOW_PRIORITY] [IGNORE] [INTO] tbl_name [(column
list)] SELECT ...
```

Команда **INSERT ... SELECT** обеспечивает возможность быстрого внесения большого количества строк в таблицу из одной или более таблиц.

Пример применения

Вывод из таблицы tblTemp1 в таблицу tblTemp2:

```
INSERT INTO tblTemp2 (fldID) SELECT tblTemp1.fldOrder_ID
FROM tblTemp1 WHERE tblTemp1.fldOrder_ID > 100;
```

Оператор EXIST (DML)

Оператор **EXIST** оценивает подзапрос как ИСТИНА или ЛОЖЬ
Синтаксис

```
EXISTS (subquery)
```

Аргументы - вложенный запрос.
Ограниченная инструкция SELECT. Ключевое слово INTO не допускается. Дополнительные сведения см. в разделе сведения о вложенных запросах [6].
Типы результата - логическое значение.
Результирующие значения - возвращает значение TRUE, если вложенный запрос содержит хотя бы одну строку.

Пример применения

UNUM – код факта сдачи учебной дисциплины,
UDATE – дата сдачи**.**

Таблица 1.3. –Успеваемость

USP (УСПЕВАЕМОСТЬ)				
UNUM	**OCENKA**	**UDATE**	**SN**	**PNUM**
1001	***5***	***10/06/2016***	***3412***	***2001***
1002	4	10/06/2016	3413	2003
1003	3	11/06/2016	3414	2005
1004	4	12/06/2016	3415	2003
1005	***5***	***10/06/2016***	***3415***	***2004***

Извлекать данные из таблицы успеваемости, если в ней есть отличные оценки.

```
SELECT * FROM USP WHERE USP.OCENKA<5 AND EXIST (SELECT *
FROM USP.OCENKA=5);
```

Операторы ANY, SOME (DML)

ANY – синоним **SOME.**

Синтаксис

```
SELECT …
WHERE выражение сравнение {ALL | ANY | SOME} (подзапрос)
```

Подробно см. [7].

Сравнивает скалярное значение с набором значений из одного столбца. Если хотя бы для одного значения V, получаемого из подзапроса, результат операции "<значение выражения> <оператор сравнения> V" равняется **TRUE,** то предикат **ANY** также равняется **TRUE.**

Пример применения

- `SELECT FROM STUDENTS WHERE SN = ANY(SELECT SN FROM USP)`

Оператор ANY берет все значения, выведенные подзапросом, т.е., в примере – все значения поля SN, и оценивает их как верные, если любое из них равняется номеру студенческого билета текущей строки внешнего запроса.
Отсюда следует, что подзапрос должен выбирать значения такого же типа, как и те, которые сравниваются в основном предикате.
Оператор ALL работает таким образом, что предикат является верным, если каждое значение, выбранное подзапросом, удовлетворяет условию в предикате внешнего запроса.
Например, если необходимо вывести только тех студентов, чьи оценки ниже полученных 12/06/2016, то можно воспользоваться следующим запросом;

```
SELECT * FROM USP WHERE OCENKA < ALL (SELECT
OCENKA FROM USP WHERE UDATE = 12/06/2016)
```

Таблица 1.4. – Успеваемость

USP (УСПЕВАЕМОСТЬ)				
UNUM	**OCENKA**	**UDATE**	**SN**	**PNUM**
1001	5	10/06/2016	3412	2001
1002	***4***	***10/06/2016***	***3413***	***2003***
1003	***3***	***11/06/2016***	***3414***	***2005***
1004	***4***	***12/06/2016***	***3415***	***2003***
1005	5	10/06/2016	3415	2004

1.2.3. Обновить UPDATE (DML)

Синтаксис

```
UPDATE table_reference
    SET col_name1={expr1|DEFAULT}
    [,col_name2={expr2|DEFAULT}] ...
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

Пример применения

- Изменить оценку по номеру предмета:
  `UPDATE USP SET OCENKA = 5 WHERE PNUM = 2003 ;`
- Увеличить стипендию в 2 раза.
  `UPDATE  STUDENTS SET STIP=STIP*2`

1.2.4. Объединение запросов – UNION

UNION – это оператор SQL для объединения результирующего набора данных нескольких запросов, и данный оператор выводит только уникальные строки в запросах, т.е. например, Вы объединяете два запроса и в каждом из которых есть одинаковые данные, другими словами полностью идентичные, и оператор UNION объединит их в одну строку для того чтобы не было дублей [2].

Синтаксис

<запрос1>
UNION [ALL]
<запрос2>
UNION [ALL]
<запрос3>
.....;

Существуют два основных правила, регламентирующие порядок использования оператора UNION:

- Число и порядок извлекаемых столбцов должны совпадать во всех объединяемых запросах;
- Типы данных в соответствующих столбцах должны быть совместимы.

Несоблюдение этих правил приведет к ошибке при формировании результирующей таблицы**.**

Пример применения

Получения списка всех студентов и преподавателей, фамилии которых заключены между буквами 'К' и 'С‘.

```
 SELECT SFAM, SIMA, SOTCH FROM STUDENTS WHERE SFAM BETWEEN 'K'
    AND 'C'
UNION
 SELECT TFAM, TIMA, TOTCH FROM TEACHERS WHERE TFAM BETWEEN  'K'
    AND 'C'
```

Здесь.

STUDENTS – БД студентов, **TEACHERS** – БД преподавателей.

SFAM–фамилия студента, **TFAM**–фамилия преподавателя.

SIMA, SOTCH, TIMA, TOTCH – имя, отчество студента, преподавателя.

1.2.5. Операция удаления DELETE (DML)

В языках, подобных **SQL, DML** - операция удаления записей из таблицы. Критерий отбора записей для удаления определяется выражением **where**. Еесли критерий отбора не определён, выполняется удаление всех записей.

Синтаксис

```
DELETE FROM tbl_name
WHERE where_condition]
ORDER BY ...]
LIMIT row_count]
```

Пример применения

Здесь удаляется запись с заданным номером студенческого билета

```
DELETE FROM STUDENTS WHERE SN = 4316
```

1.2.6. Агрегатные функции

Агрегатные функции выполняют вычисление на наборе значений и возвращают одиночное значение. Агрегатные функции, за исключением COUNT, не учитывают значения NULL. Агрегатные функции часто используются в выражении GROUP BY инструкции SELECT [3].

COUNT - производит подсчет количества строк;

SUM - рассчитывает арифметическую сумму всех выбранных значений данного поля;

AVG - производит усреднение всех выбранных значений данного поля;

MAX - находит и возвращает наибольшее из всех выбранных значений данного поля;
MIN - находит и возвращает наименьшее из всех выбранных значений данного поля.

Пример применения

```
SELECT COUNT( DISTINCT first_name) FROM employees;
```
Результат: **3**

Обратите внимание, что в приведенном выше примере используется ключевое слово **DISTINCT** - это означает, что подсчитываться будет количество уникальных значений в колонке **first_name** таблицы **employees**.

Если опустить его, результат изменится:
Результат: **4**

1.3. Операции определения доступа к данным (DCL)

Команда GRANT используется для назначения привилегий и разрешает определённому пользователю выполнять указанные действия.

Синтаксис

```
GRANT { ALL [ PRIVILEGES ] }
      | permission [ ( column [ ,...n ] ) ] [ ,...n ]
      [ ON [ class :: ] securable ] TO principal [ ,...n ]
      [ WITH GRANT OPTION ] [ AS principal ]
```

REVOKE, соответственно, их запрещает.
С помощью команды **REVOKE** осуществляется отмена привилегий.

Синтаксис

```
REVOKE [ GRANT OPTION FOR ]
      {
        [ ALL [ PRIVILEGES ] ]
        |
       permission [ ( column [ ,...n ] ) ] [ ,...n ]
      }
      [ ON [ class :: ] securable ]
      { TO | FROM } principal [ ,...n ]
      [ CASCADE] [ AS principal ]
```

Пример применения

```
GRANT ALL PRIVILEGES ON enterprise.employees TO
'testuser'@'localhost' IDENTIFIED BY 'P@SSW0RD'
REVOKE DELETE ON enterprise.employees FROM 'testuser';
```

Подробно см. [8,9]

Следующие привилегии могут быть **GRANTED TO** или **REVOKED**

FROM, CONNECT, SELECT, INSERT, UPDATE, DELETE, EXECUTE, USAGE

1.4. Представление (VIEW)

Представление (VIEW) (называется также "просмотр") - это объект данных, который не содержит никаких данных его владельца. Это тип таблицы, чьё содержание выбирается из других таблиц с помощью выполнения запроса. По мере изменения значений в таблицах, эти изменения автоматически отражаются представлением [13].

Определение

Представление - динамический результат одной или нескольких реляционных операций над базовыми отношениями с целью создания некоторого иного отношения.

Достоинства представлений:

- разграничение данных для отображения;
- удобство способа доступа к данным;
- упрощение сложных операций с базовыми отношениями.

Действия с представлениями:

запрос, модификация, вставка в, удаление из,
соединение с другими таблицами и представлениями

Создание представления CREATE VIEW

Синтаксис

Представление создается командой

CREATE VIEW имя представления, которое нужно создать
слова AS (КАК) запрос

Пример применения

Создать представление **SWERLORight из SWERLO**

```
CREATE VIEW SWERLORight  AS SELECT * FROM SWERLO  WHERE NV = 1;
```

Действия с представлениями

запрос, модификация, вставка в, удаление из
соединение с другими таблицами и представлениями
CREATE VIEW

Таблица 1.5. – Список студентов.

STUDENTS				
SN	SFAM	SIMA	SOTCH	STIP
3412	Поляков	Анатолий	Алексеевич	25.50
3413	Старова	Любовь	Михайловна	17.00
3414	Гриценко	Владимир	Николаевич	0.00
3415	Котенко	Анатолий	Николаевич	0.00
3416	Нагорный	Евгений	Витальевич	25.00

Пример создания представления

На основе таблицы 1.5. **STUDENTS** создать представление **OTLSTUD**, которое содержит информацию о студентах со стипендией 25.50.

```
CREATE VIEW OTLSTUD AS SELECT FROM STUDENTS WHERE STIP=25.50
```

Применение представления

```
SELECT * FROM OTLSTUD
```

Выполняется запрос, описанный при создании представления **OTLSTUD**

Таблица 1.6. – Представление **OTLSTUD**

OTLSTUD				
SN	**SFAM**	**SIMA**	**SOTCH**	**STIP**
3412	Поляков	Анатолий	Алексеевич	25.50
3416	Нагорный	Евгений	Витальевич	25.00

Возвращает всё из его вывода:

Модифицирование представлений

Представление может изменяться командами модификации DML, но модификация не будет воздействовать на само представление. Команды будут на самом деле перенаправлены в базовую таблицу:

```
- UPDATE OTLSTUD SET SOTCH='Антонович' WHERE SN = 3412;
```

Его действие идентично выполнению той же команды в таблице **STUDENTS.**

```
- UPDATE OTLSTUD SET SOTCH='Антонович' WHERE GROUPS = 4657;
```

Запрос будет отвергнут, так как поле **GROUPS** отсутствует в представлении **STUDENTS.**
Не все представления могут быть модифицированы [14].

1.5. Операторы Transaction Control Language (TCL)

Транзакция (англ. transaction) — в информатике, группа последовательных операций, которая представляет собой логическую единицу работы с данными. Транзакция может быть выполнена целиком либо успешно, соблюдая целостность данных и независимо от параллельно идущих других транзакций, либо не выполнена вообще и тогда она не должна произвести никакого эффекта. Транзакции обрабатываются транзакционными системами, в процессе работы которых создаётся история транзакций. Операторы управления транзакциями (TCL) компьютерный язык и часть SQL, используемый для обработки транзакций.

Примеры TCL команд

COMMIT применяет транзакцию.
ROLLBACK "откатывает" все изменения сделанные транзакцией.
SAVEPOINT делит транзакцию на более мелкие участки.
COMMIT (TCL)

Оператор COMMIT применяется для того, чтобы:

- сделать «постоянными» все изменения, сделанные в текущей транзакции;
- очистить все точки сохранения данной транзакции;
- завершить транзакцию;
- освободить все блокировки данной транзакции.

Пример добавления строки к таблице MyTable и сохранения изменения:

```
BEGIN TRANSACTION;
INSERT INTO MyTable VALUES ('50', 'some string');
COMMIT WORK;
ROLLBACK
```

Оператор **ROLLBACK** применяется для того, чтобы:

- отменить все изменения, внесённые начиная с момента начала транзакции или с какой-то точки сохранения (**SAVEPOINT**);
- очистить все точки сохранения данной транзакции;
- завершить транзакцию;
- освободить все блокировки данной транзакции.

Пример применения

Добавление записи к таблице MyTable и последующая отмена этого действия:

```
BEGIN TRANSACTION;
INSERT INTO MyTable VALUES ('50', 'some string');
ROLLBACK WORK;
```

1.5.2. Курсоры

Курсор в SQL – это область в памяти базы данных, которая предназначена для хранения последнего оператора SQL. Если текущий оператор – запрос к базе данных, в памяти сохраняется и строка данных запроса, называемая текущим значением, или текущей строкой курсора. Указанная область в памяти поименована и доступна для прикладных программ:.

- курсоры SQL применяются в основном внутри триггеров, хранимых процедур и сценариев;
- курсоры сервера действуют на сервере и реализуют программный интерфейс приложений для ODBC, OLE DB, DB_Library;
- курсоры клиента реализуются на самом клиенте. Они выбирают весь результирующий набор строк из сервера и сохраняют его локально, что позволяет ускорить операции обработки данных за счет снижения потерь времени на выполнение сетевых операций.

Команды управления курсорами:

DECLARE – создание курсора;

OPEN – открытие\заполнение данными курсора;

FETCH – выборка\изменение строк курсора;

CLOSE – закрытие курсора;

DEALLOCATE – удаление курсора.

Пример применения

Создаем тестовую таблицу и заполняем данными.

```
CREATE TABLE [dbo].[test](
    id INT,
    name VARCHAR(30),
    lastname VARCHAR(30)
)

INSERT INTO [dbo].[test] VALUES(1,'Name1','lastanme1')
INSERT INTO [dbo].[test] VALUES(2,'Name2','lastanme2')
INSERT INTO [dbo].[test] VALUES(3,'Name3','lastanme3')
```

Переменные для работы

```
DECLARE @id INT
DECLARE @name VARCHAR(30), @lastname VARCHAR(30)
```

--Создадим курсор

```
DECLARE cur CURSOR FOR
SELECT id, name, lastname FROM [dbo].[test]
```

--Откроем курсор

```
OPEN cur

--Выборка данных первой строки
FETCH NEXT FROM cur INTO @id, @name, @lastname

Пока есть данные в курсоре - выборка циклом
WHILE @@FETCH_STATUS = 0
BEGIN
    SELECT @id, @name, @lastname
    --Выборка следующей строки
    FETCH NEXT FROM cur INTO @id, @name, @lastname
END
```

Закрываем курсор

```
CLOSE cur
```

```
Уничтожаем курсор
DEALLOCATE cur
DROP TABLE [dbo].[test]
```

Подробно см. [10, 11, 12]

2. Практическая часть работы с СУБД MS SQL

2.1. Создание БД и хранимых процедур

Цель работы

Научиться самостоятельно создавать, заполнять и редактировать базы данных в СУБД MySQL. Закрепить навыки работы с БД, в частности, навыки написания и выполнения хранимых процедур, а также просмотра результатов их работы.

Постановка задачи

Разработать БД режущего инструмента следующего содержания:

Таблица 2.1.1. – Патрон для нарезания резьбы плашкой

Обозначение, **V_OB**	Конус Морзе хвостового патрона, **V_NKH**	Диаметр посадочной втулки, **V_DPO**	Доп. вылет патрона, **V_VI**
6610-0021	3	38.0	199
6610-0022	4	45.0	222
6610-0023	4	55.0	237
6610-0024	4	65.0	257

Для этого:

1. Разработать структуру БД. В качестве имен полей применять латинскую часть названий граф (типы полей определить по содержанию граф);
2. Заполнить БД (данную в задании таблицу дополнить до 30 строк);
3. Разработать хранимую процедуру для поиска:
   a. По заданному конусу Морзе;
   b. По диаметру и вылету;
4. Результат поиска показать на экране и вывести в файл для печати.

Ход работы

Создадим базу данных **ИНСТРУМЕНТ** и добавим в нее таблицу **Patron**. В созданной таблице определим следующие поля:

Таблица 2.1.2. – Поля таблицы **Patron**

Название поля	**Смысл поля**	**Тип**	**Первичный ключ**
V_OB	Обозначение	nvarchar(10)	Да
V_NKH	Конус Морзе хвостового патрона	smallint	Нет
V_DPO	Диаметр посадочной втулки	float	Нет
V_VI	Доп. вылет патрона	smallint	Нет

Заполним таблицу данными двумя способами: изученным раннее графическим способом и с помощью команды INSERT языка DML (Data Manipulation Language – язык манипуляции данными). Код запроса на заполнение приведен в листинге 2.1.1 приложения к настоящему отчету.

Таблица 2.1.3. – Заполненная таблица **Patron**

V_OB	**V_NKH**	**V_DPO**	**V_VI**
6610-0021	3	38	199
6610-0022	4	45	222
6610-0023	4	55	237
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....
6610-0047	8	76	269
6610-0048	8	76	270
6610-0049	9	76	275

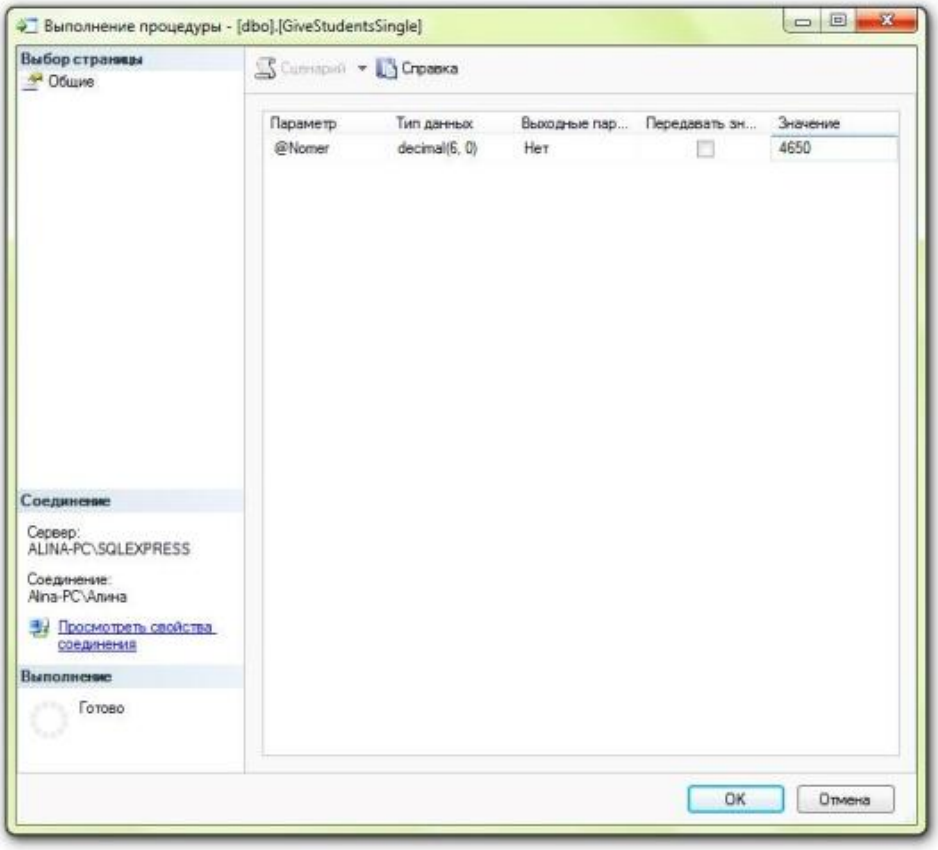


Рисунок 2.1.1. Выполнение процедуры поиска по конусу Морзе патрона

Напишем хранимую процедуру для поиска по коду Морзе хвостового патрона (Листинг 2.1.2). Для этого введем параметр @morse типа smallint, на равенство которому будем проверять поле V_NKH таблицы Patron.

В результате получим выборку данных со значением конуса Морзе, соответствующем заданному (Рисунок 2.1.2.).

	V_OB	V_NKH	V_DPO	V_VI
1	6610-0030	6	70	260
2	6610-0031	6	70,4	260
3	6610-0032	6	71	260
4	6610-0033	6	71	262
5	6610-0034	6	72	262
6	6610-0035	6	72	264

Рисунок 2.1.2. Результат поиска по значению конуса Морзе

По аналогии напишем код хранимой процедуры selectWithDPOandVI, осуществляющей поиск по двум параметрам:

- @dpo – диаметр посадочной втулки;
- @vi – дополнительный вылет патрона.

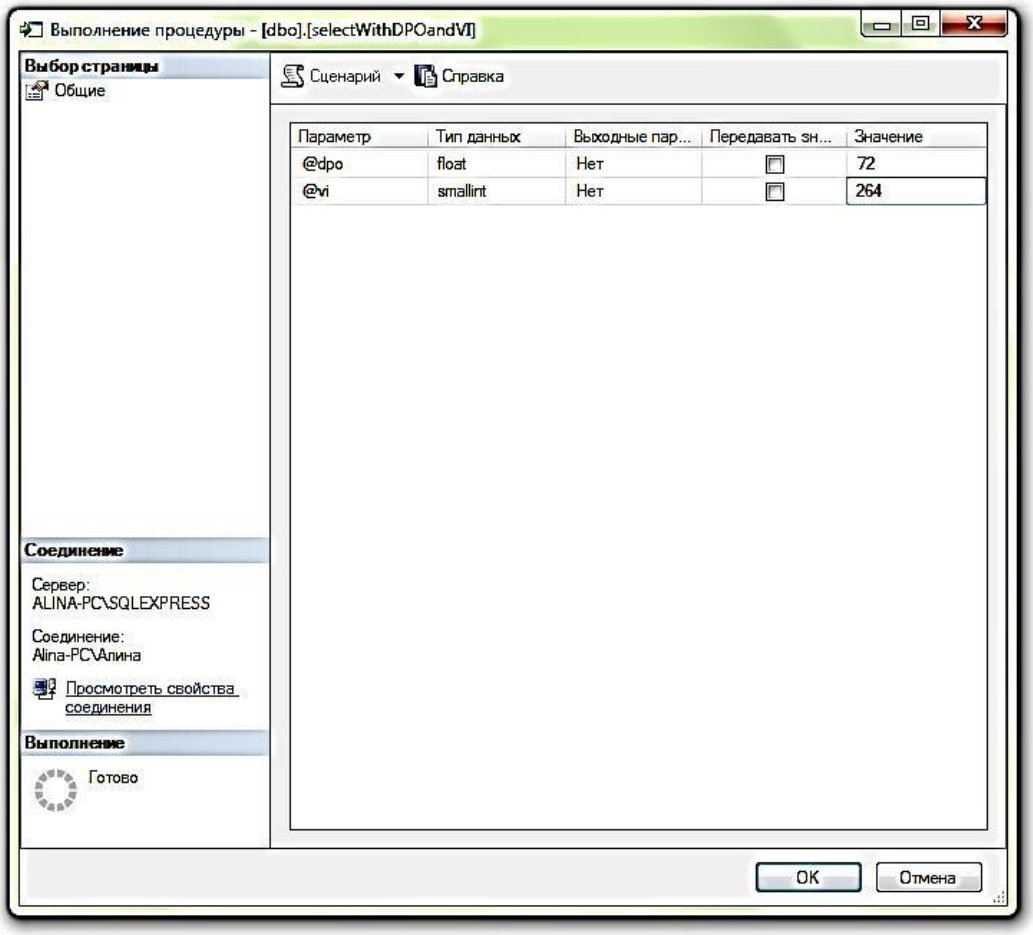
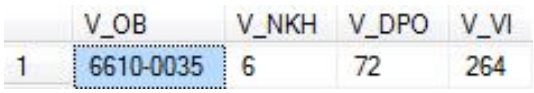


Рисунок 2.1.3. Выполнение процедуры поиска selectWithDPOandVI

В результате выполнения процедуры получим данные о патроне, соответствующем запрашиваемому (**Ошибка! Источник ссылки не найден.**).



	V_OB	V_NKH	V_DPO	V_VI
1	6610-0035	6	72	264

Рисунок 2.1.4. Результат выполнения процедуры selectWithDPOandVI

Выведем результаты выполнения процедур в файлы для дальнейшей печати. Для этого в контекстном меню окна с таблицей результатов выберем пункт «Сохранить результаты как…». В результате получим файл с расширением .txt (тест) или .csv (таблица) в зависимости от пользовательских настроек.

Таблица 2.1.4.–Результат выполнения процедуры selectWithMorse в формате .cvs

6610-0030	6	70	260
6610-0031	6	70,4	260
6610-0032	6	71	260
6610-0033	6	71	262
6610-0034	6	72	262
6610-0035	6	72	264

Таблица 2.1.5.–Результат выполнения процедуры selectWithDPOandVI в формате .cvs

6610-0035	6	72	264

Выводы по работе

В ходе выполнения лабораторной работы было произведено закрепление базовых навыков работы с СУБД MySQL в среде Microsoft SQL Server Management Studio. Была самостоятельно создана таблица с данными Patron, произведено ее заполнение двумя способами и сгенерированы процедуры поиска данных в таблице. Результаты поиска в таблице были сохранены в файлы .txt и .csv для дальнейшего возможного просмотра, анализа и печати.

2.2. Создание БД Student

Цель работы

Ознакомиться с интерфейсом и возможностями реляционной системы управления базами данных MySQL. Получить фундаментальные навыки

создания баз данных в MySQL и работы с ними: генерации запросов, написания процедур и т.п.

Постановка задачи

В ходе выполнения лабораторной работы необходимо выполнить следующие обязательные пункты:

Создать базу данных Student.
Добавить в готовую БД таблицы Student и Group.
Задать первичный и вторичный ключи в таблицах.
Осуществить запросы.
Создать хранимые процедуры.
Создать отчет (представление).

Ход работы

Создадим базу данных Student и добавим в нее таблицы:

Student (данные о студентах);
Group (данные о группах).

В созданных таблицах определим следующие поля:

- Таблица Student:
  - NumberZachet – номер зачетки (первичный ключ);
  - FirstName – имя;
  - LastName – фамилия;
  - GroupName – номер группы;
  - BirthDate – дата рождения;
  - Sex – пол;
  - Stip – стипендия;
- Таблица Group
  - GroupName – номер группы (первичный ключ);
  - CountMembers – количество студентов;
  - StudyYears – срок обучения;
  - Starosta – фамилия старосты группы;
  - Specializacia – специализация группы.

Заполним готовые таблицы данными и произведем связь таблиц по ключу. Созданные ключи можно увидеть в обозревателе объектов (Object Explorer). Произведем генерацию запроса при помощи дизайнера (графический способ создания запроса - Рисунок 2.2.2.). Выберем поля, которые хотим запросить, из БД. Внизу окна конструктора запросов можно увидеть код запроса, который дизайнер генерирует автоматически.

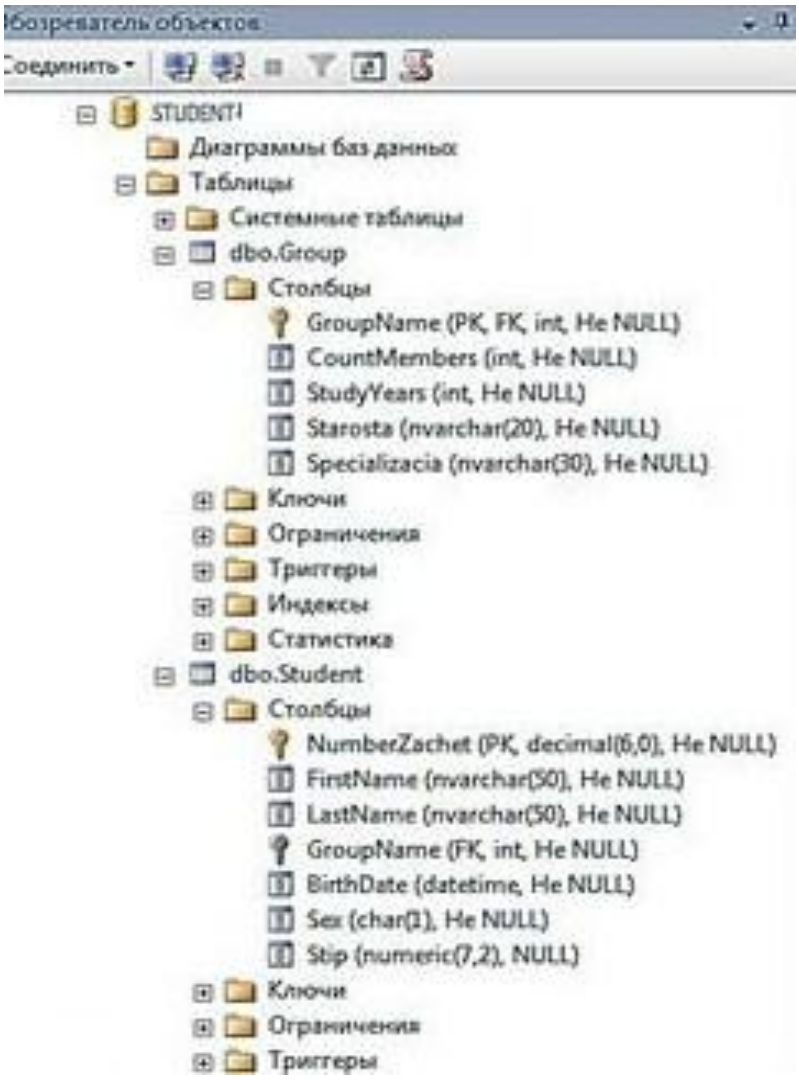


Рисунок 2.2.1. Таблицы Student и Group в обозревателе объектов

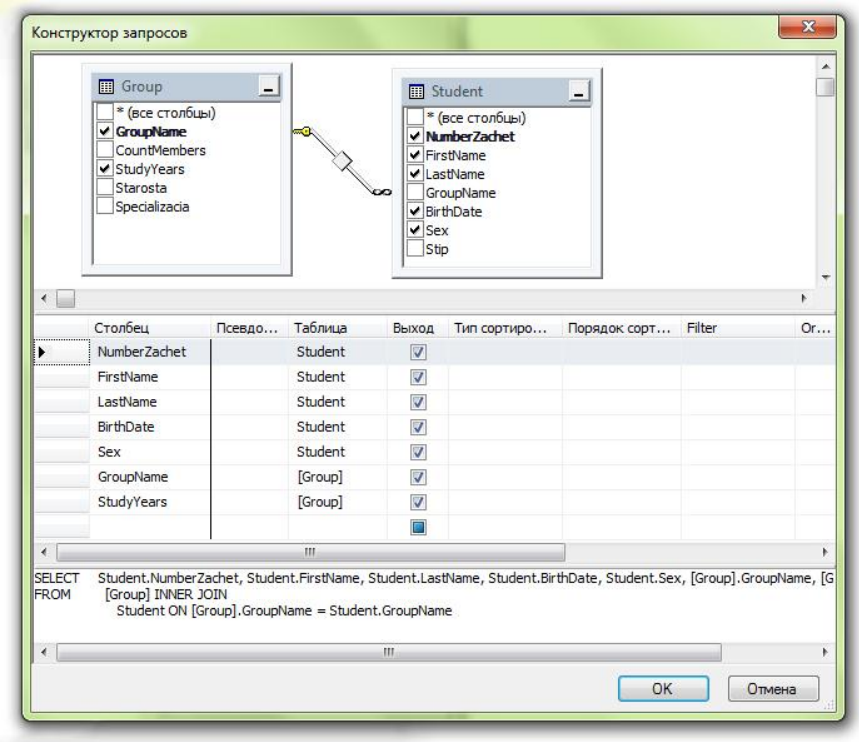


Рисунок 2.2.2. Создание запроса графическим способом

Получим следующий код запроса:

```
SELECT Student.NumberZachet, Student.FirstName,
Student.LastName, Student.BirthDate, Student.Sex,
[Group].GroupName, [Group].StudyYears FROM  [Group] INNER JOIN
Student ON [Group].GroupName = Student.GroupName
```

Выполним запрос и получим неупорядоченные данные обо всех студентах (Рисунок 2.2.3).

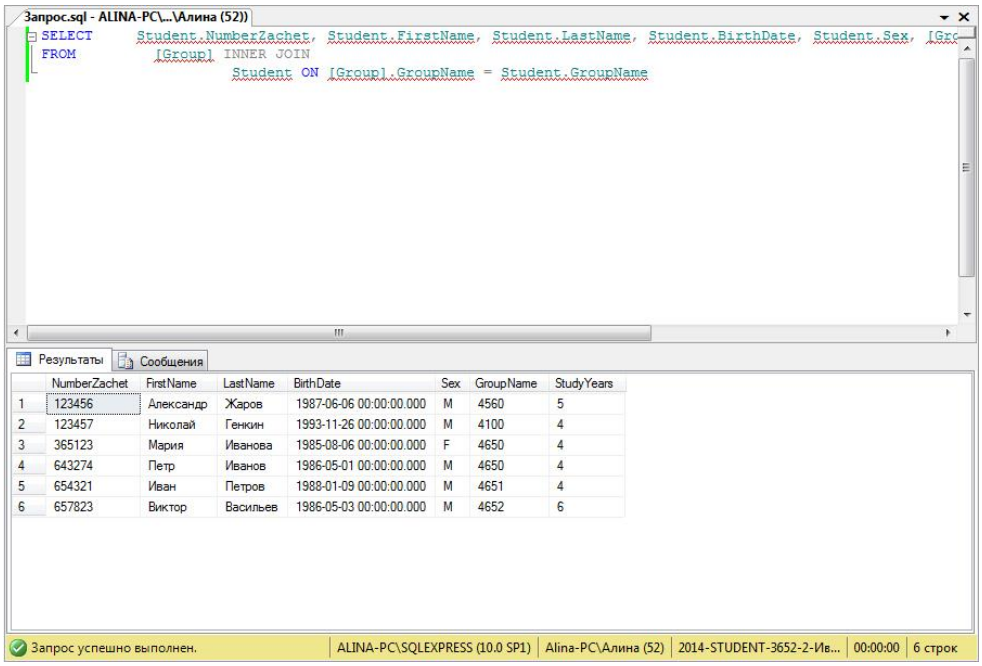


Рисунок 2.2.4. Выполнение первого запроса

Организуем сортировку по номеру группы, дописав в код запроса следующую строку:

ORDER BY GroupName

Использование предложения ORDER BY является единственным способом отсортировать результирующий набор строк. Без этого предложения СУБД может вернуть строки в любом порядке. В данном примере производится упорядочение результирующего набора по числовому столбцу GroupName. Поскольку конкретный порядок сортировки не указан, используется порядок по умолчанию (по возрастанию).
В результате получим следующую таблицу (Рисунок 2.2.8).
Для того, чтобы объединить студентов по группам и сосчитать количество студентов в каждой группе, добавим строку:

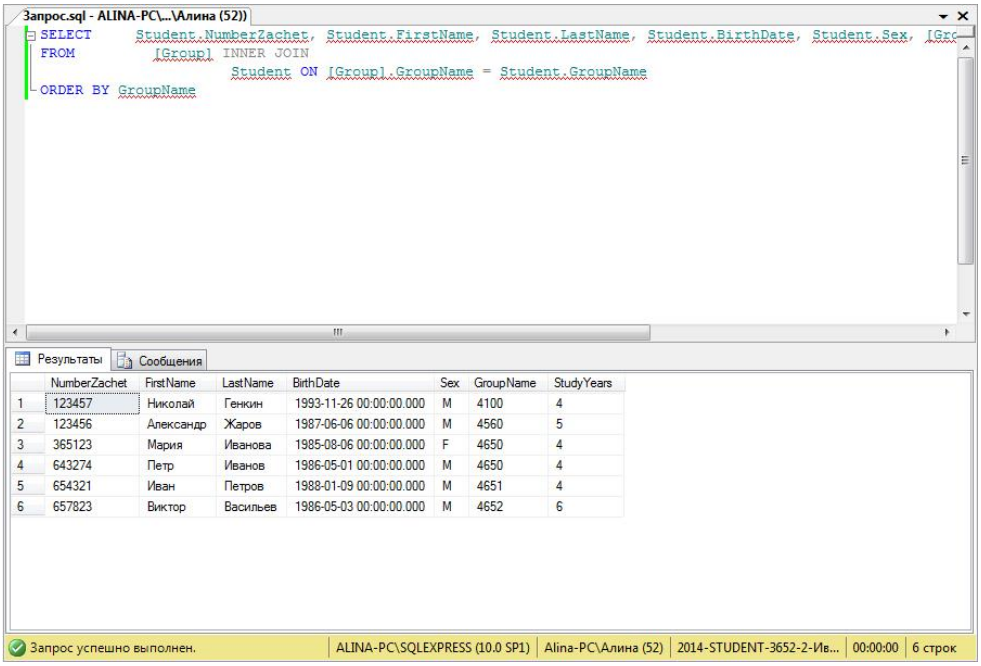


Рисунок 2.2.5. Данные о студентах, отсортированные по номеру группы

```
COMPUTE COUNT(NumberZachet) BY GroupName
```

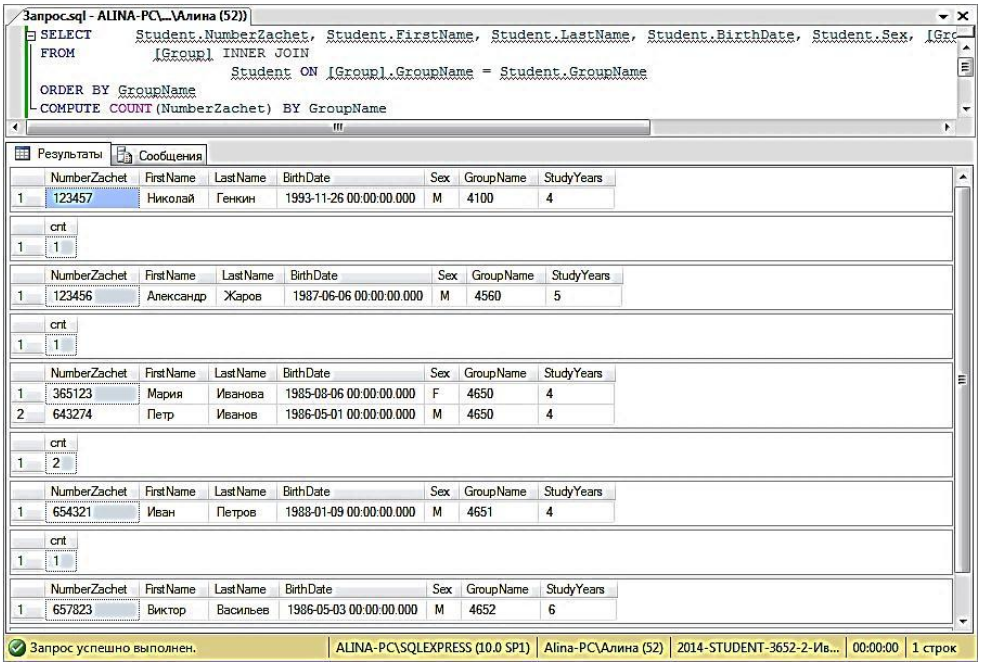


Рисунок 2.2.5. Подсчет количества студентов в каждой группе

Таким образом, будут выведены списки и количества студентов по группам (Рисунок 2.2.6). Используя код последнего запроса, напишем хранимую процедуру GiveStudents (Листинг 2.2.2).

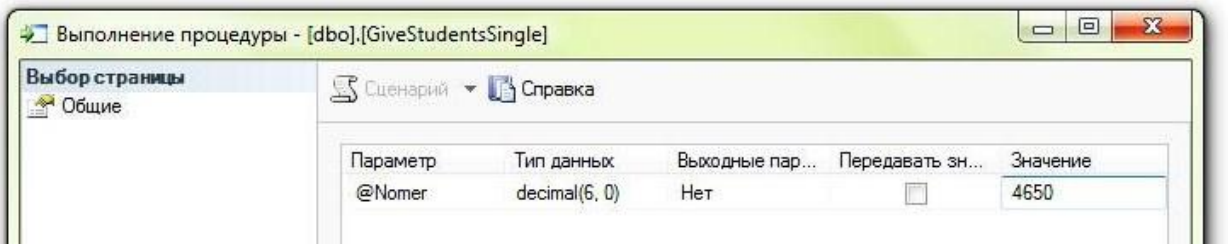


Рисунок 2.2.6. Выполнение процедуры GiveStudentsSingle

Листинг 2.2.1. Хранимая процедура GiveStudents

```
CREATE PROCEDURE GiveStudents
AS
BEGIN
     SELECT     Student.NumberZachet, Student.FirstName,
Student.LastName, Student.BirthDate, Student.Sex,
[Group].GroupName, [Group].StudyYears
     FROM          [Group] INNER JOIN
     Student ON [Group].GroupName = Student.GroupName
     ORDER BY GroupName
     COMPUTE COUNT(NumberZachet) BY GroupName
END
GO
```

Листинг 2.2.2. Хранимая процедура **GiveStudentsSingle**

```
CREATE PROCEDURE GiveStudentsSingle
@Nomer decimal(6, 0)
AS
BEGIN
     SELECT     Student.NumberZachet, Student.FirstName,
Student.LastName, Student.BirthDate, Student.Sex,
[Group].GroupName, [Group].StudyYears
     FROM          [Group] INNER JOIN
                              Student ON [Group].GroupName =
Student.GroupName
     WHERE [Group].GroupName = @Nomer
     COMPUTE COUNT(NumberZachet)
END
GO
```

Результат выполнения процедуры аналогичен результату выполнения последнего запроса, приведенному выше.

Создадим процедуру GiveStudentsSingle (Листинг 2.2.4), принимающую один входной параметр Nomer (номер группы). Эта процедура будет выводить список студентов определенной группы.

При выполнении данной процедуры программа запросит значение параметра Nomer (Рисунок 2.2.6). По заданному запросу пользователь вводит номер группы и запускает процедуру на поиск. Результатом выполнения процедуры будет список студентов выбранной группы (Рисунок 2.2.7).Создадим еще одну хранимую процедуру – GiveMeStudent (Листинг 2.2.3), которая будет выводить информацию только по одному студенту. С помощью этой процедуры получаем полную информацию по конкретному студенту.

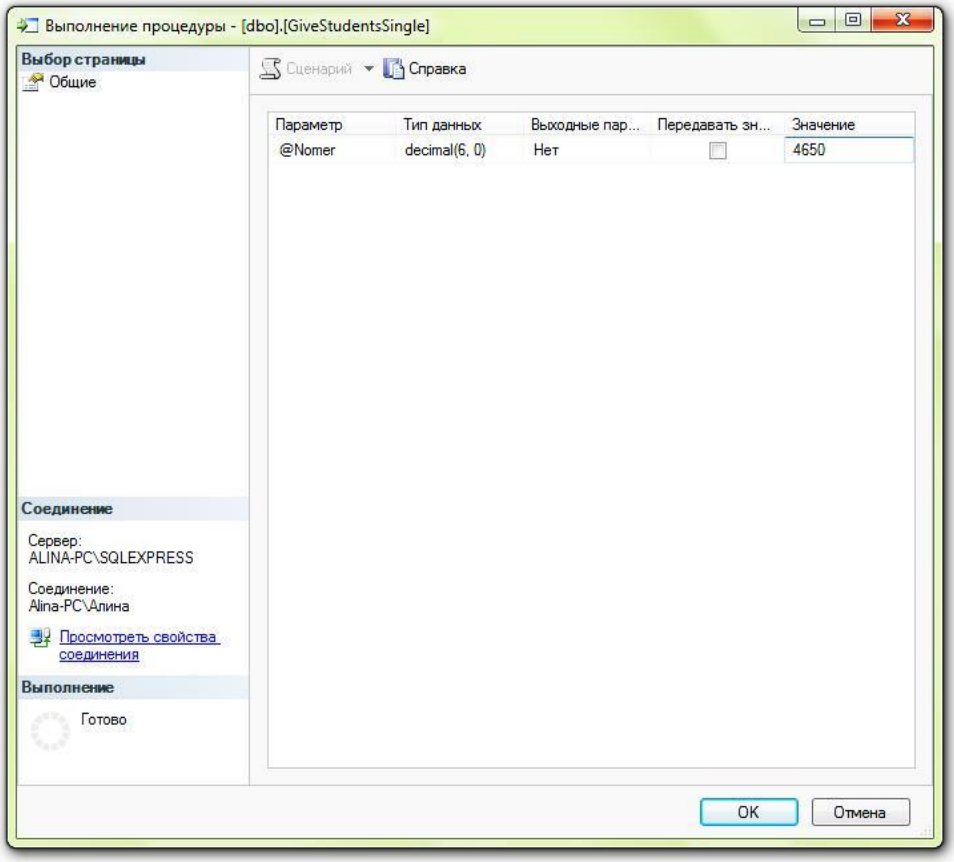


Рисунок 2.2.7. Выполнение процедуры GiveStudentsSingle

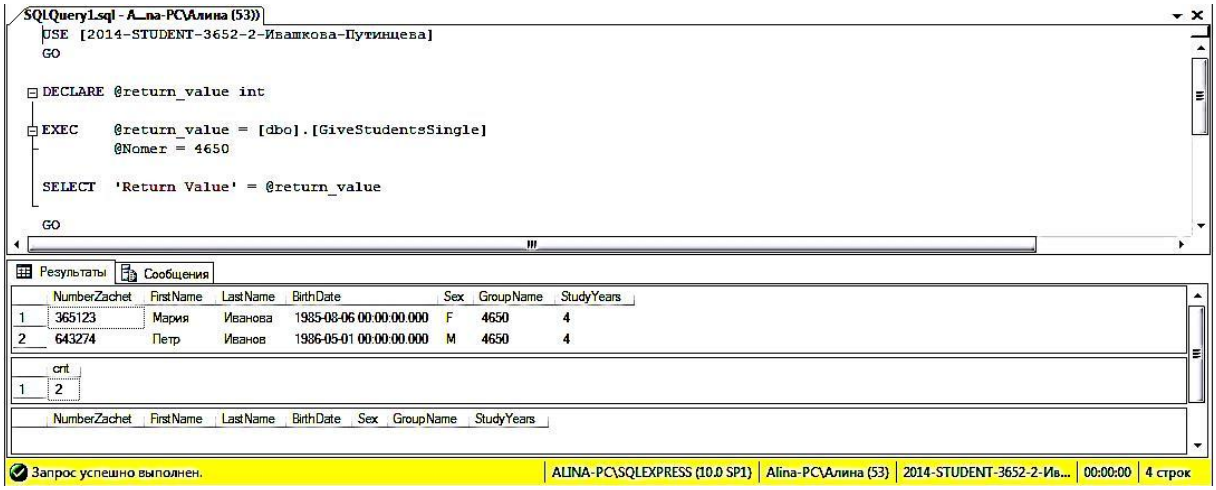


Рисунок 2.2.8. Результат выполнения процедуры GiveStudentsSingle

Листинг 2.2.3. Хранимая процедура GiveMeStudent

```
CREATE PROCEDURE GiveMeStudent
@Nomer decimal(6, 0),
@Name nvarchar(50),
@LName nvarchar(50)
AS
BEGIN
     SELECT     Student.NumberZachet, Student.FirstName,
Student.LastName, Student.BirthDate, Student.Sex,
[Group].GroupName, [Group].StudyYears
     FROM          [Group] INNER JOIN
                                Student ON [Group].GroupName =
Student.GroupName
     WHERE [Group].GroupName = @Nomer AND FirstName LIKE @Name
AND LastName LIKE @LName
     COMPUTE COUNT(NumberZachet)
END
GO
```

В результате выполнения процедуры GiveMeStudent (Рисунок 2.2.9) и ввода значений параметров Nomer, Name и LName, будет выведена информация по запрошенному студенту (Рисунок 2.2.10).

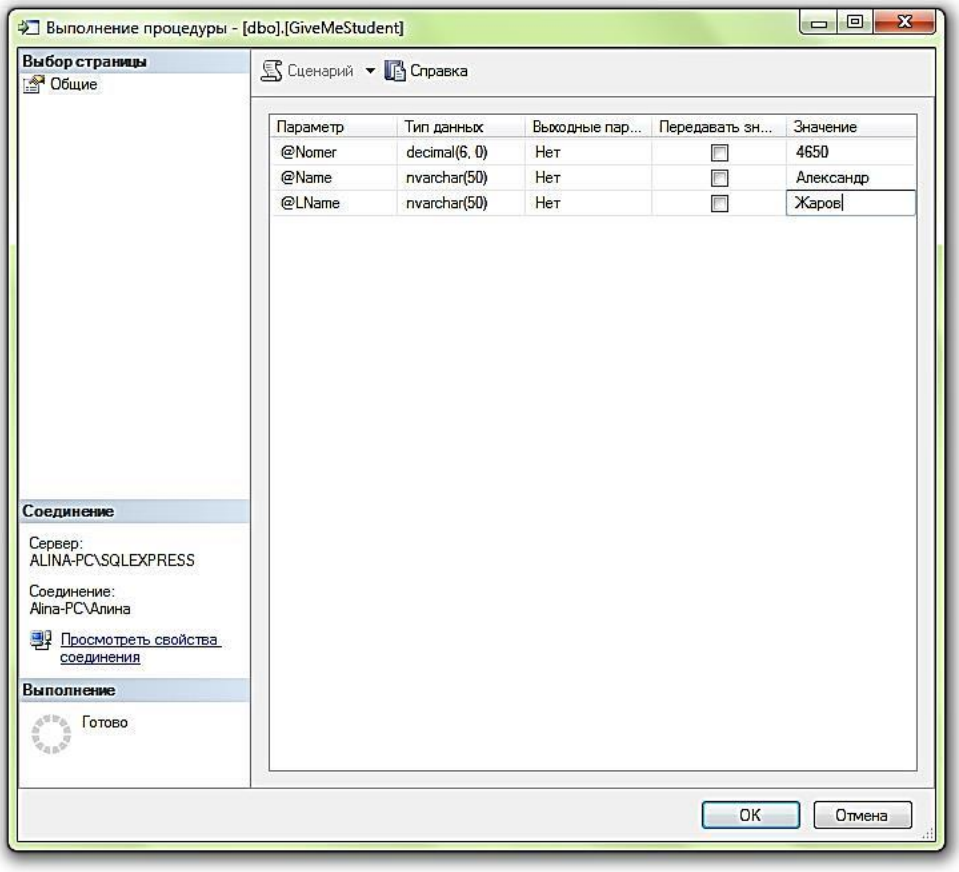


Рисунок 2.2.9. Выполнение процедуры GiveMeStudent

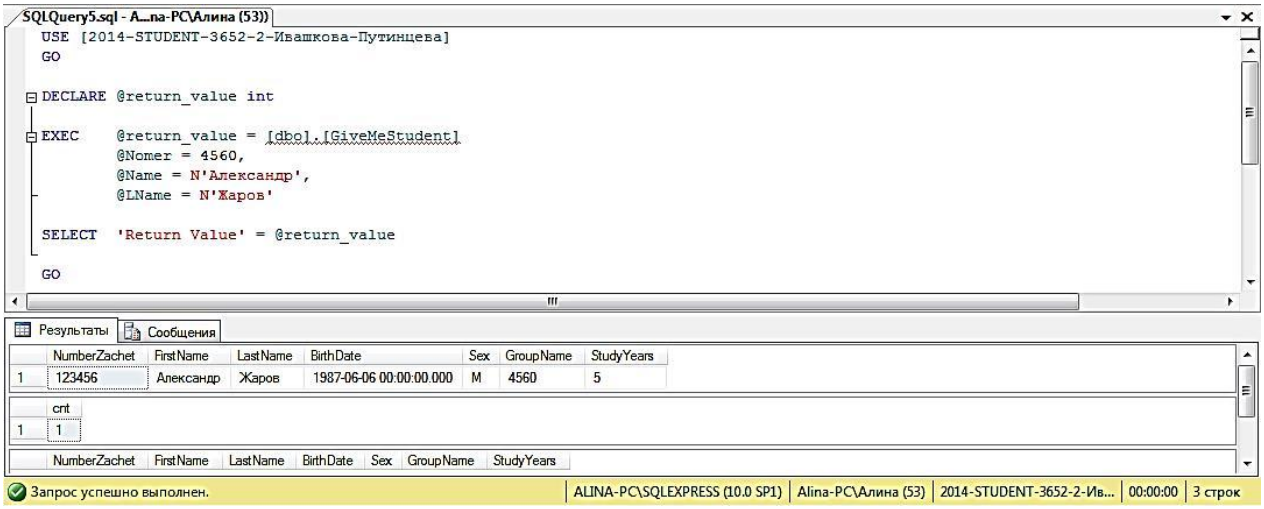


Рисунок 2.2.10. Вывод информации по конкретному студенту

Создадим отчет (представление) на основе имеющихся в БД данных. Для этого выберем таблицу Student и отметим поля, которые требуется отобразить в отчете. Созданный отчет показан на рисунке 2.2.11

Применение представления

Номер зачетки	Имя	Фамилия	Группа	Дата рождения	Пол	Стипендия
123456	Александр	Жаров	4560	1987-06-06 00:...	M	1100,00
123457	Николай	Генкин	4100	1993-11-26 00:...	M	2650,00
365123	Мария	Иванова	4650	1985-08-06 00:...	F	1500,00
643274	Петр	Иванов	4650	1986-05-01 00:...	M	1029,00
654321	Иван	Петров	4651	1988-01-09 00:...	M	NULL
657823	Виктор	Васильев	4652	1986-05-03 00:...	M	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 2.2.11. Отчет

Выводы по работе

В ходе выполнения текущей лабораторной работы было произведено знакомство с СУБД MySQL и средой Microsoft SQL Server Management Studio. На основе имеющейся в задании информации была создана БД Student, содержащая связанные между собой таблицы Student и Group, аккумулирующие данные о студентах и группах соответственно.

Путем использования данных таблиц и примера из задания к работе, были сгенерированы различные запросы, хранимые процедуры и отчет. Приведен пример работы с агрегатными функциями.

Таким образом, были получены базовые навыки работы с СУБД MS SQL и общее представление о логике работы ..

2.3. Создание БД каталог технологических процессов

Цель работы

При помощи СУБД Microsoft SQL разработать базу данных «Каталог технологических процессов».

"Технологический процесс"— это часть производственного процесса, содержащая целенаправленные действия по изменению и (или) определению состояния предмета труда. К предметам труда относят заготовки и изделия.

— ГОСТ 3.1109-82

В зависимости от применения в производственном процессе для решения одной и той же задачи различных приёмов и оборудования различают следующие "виды техпроцессов":

- Единичный технологический процесс (ЕТП) — технологический процесс изготовления или ремонта изделия одного наименования, типоразмера и исполнения, независимо от типа производства.
- Типовой технологический процесс (ТТП) — технологический процесс изготовления группы изделий с общими конструктивными и технологическими признаками.
- Групповой технологический процесс (ГТП) — технологический процесс изготовления группы изделий с разными конструктивными, но общими технологическими признаками.[1]

В промышленности и сельском хозяйстве описание технологического процесса выполняется в документах, именуемых операционная карта технологического процесса (при подробном описании) или маршрутная карта (при кратком описании).

- Маршрутная карта — описание маршрутов движения по цеху изготовляемой детали.
- Операционная карта — перечень переходов, установок и применяемых инструментов.
- Технологическая карта — документ, в котором описан: процесс обработки деталей, материалов, конструкторская документация, технологическая оснастка.

Технологические процессы делят на "типовые" и "перспективные".

- Типовой "техпроцесс" имеет единство содержания и последовательности большинства технологических операций и переходов для группы изделий с общими конструкторскими принципами.

- "Перспективный техпроцесс" предполагает опережение (или соответствие) прогрессивному мировому уровню развития технологии производства.

  Управление проектированием технологического процесса осуществляется на основе маршрутных и операционных технологических процессов".

- "Маршрутный технологический процесс" оформляется маршрутной картой, где устанавливается перечень и последовательность технологических операций, тип оборудования, на котором эти операции будут выполняться; применяемая оснастка; укрупненная норма времени без указания переходов и режимов обработки.
- "Операционный технологический процесс" детализирует технологию обработки и сборки до переходов и режимов обработки. Здесь оформляются операционные карты технологических процессов.

Входные данные

Согласно заданию, тело итоговой таблицы должно иметь вид и соответствовать формату:

DAT	date	Дата регистрации
FN	nchar(15)	Фамилия нормоконтролера
FP	nchar(15)	Фамилия проверяющего
FR	nchar(15)	Фамилия разработчика
FU	nchar(15)	Фамилия утверждающего
KIM	numeric(5,3)	Коэффициент использования материала
LI	nchar(4)	Литера
MV	nchar(3)	Вид ТП по методу выполнения
MP	numeric(1)	Вид ТП по методу проектирования
OMK	nchar(5)	Обозначение маршрутной карты
OP	nchar(1)	Обозначение производства
OSK	nchar(5)	Обозначение сопроводительной карты
OTD	nchar(5)	Обозначение комплекта технологической документации
OVO	nchar(5)	Обозначение ведомости оснастки
RN	nchar(5)	Регистрационный номер
RNK1	nchar(5)	Регистрационный номер карты технологического контроля
RNK2	nchar(5)	Регистрационный номер карты технологического контроля.
RNKD	nchar(5)	Регистрационный номер комплекта документации
RNMK	nchar(5)	Регистрационный номер маршр. карты
RNTD	nchar(5)	Регистрационный номер комплекта технологической документации
RNTP	nchar(5)	Регистрационный номер технологического. процесса
RNVO	nchar(5)	Регистрационный номер ведомости оснастки
SOD	nchar(74)	Примечание
TO	nchar(60)	Текст
VOB	nchar(25)	Вид

Для создания таблиц-справочников нам понадобится информация о классификации ТП по методам выполнения и методам проектирования. Различают три основных метода проектирования технологических процессов:

1. Метод адресации к унифицированным (типовым или групповым) технологическим процессам.
2. Метод синтеза технологических процессов.
3. Поиск детали аналога и заимствование процесса на деталь-аналог

Поясним некоторые пункты для более ясного понимания структуры некоторых таблиц:

Метод адресации - это метод, основанный на использовании метода групповой обработки деталей и организации группового производства. Для этого метода характерна высокая типизация решений. Предельная типизация решений достигается при использовании типовых ТП. Разновидностью метода адресации является метод, основанный на заимствовании существующих ТП на основе поиска деталей аналогов. Метод синтеза является универсальным методом, предназначенным для проектирования технологических процессов на детали и сборочные единицы для любых изделий.

Проектирование ТП на основе заимствования технологии детали-аналога. В этом методе в первую очередь выполняют поиск детали-аналога. Поиск детали-аналога можно осуществить 2 способами: вручную (по децимальному номеру в архиве); на ЭВМ с помощью информационно-поисковой системы (ИПС).

Различают четыре метода выполнения технологического процесса:

1. Ручной
2. Кооперированно-ручной
3. Механизированно-ручной
4. Автоматический

Ход работы

Рассмотрим структуру базы данных.

База данных состоит из 5-ти таблиц, из которых 3 являются справочниками (Сотрудники, Методы Выполнения, Методы проектирования), и одна из таблиц является вспомогательной.

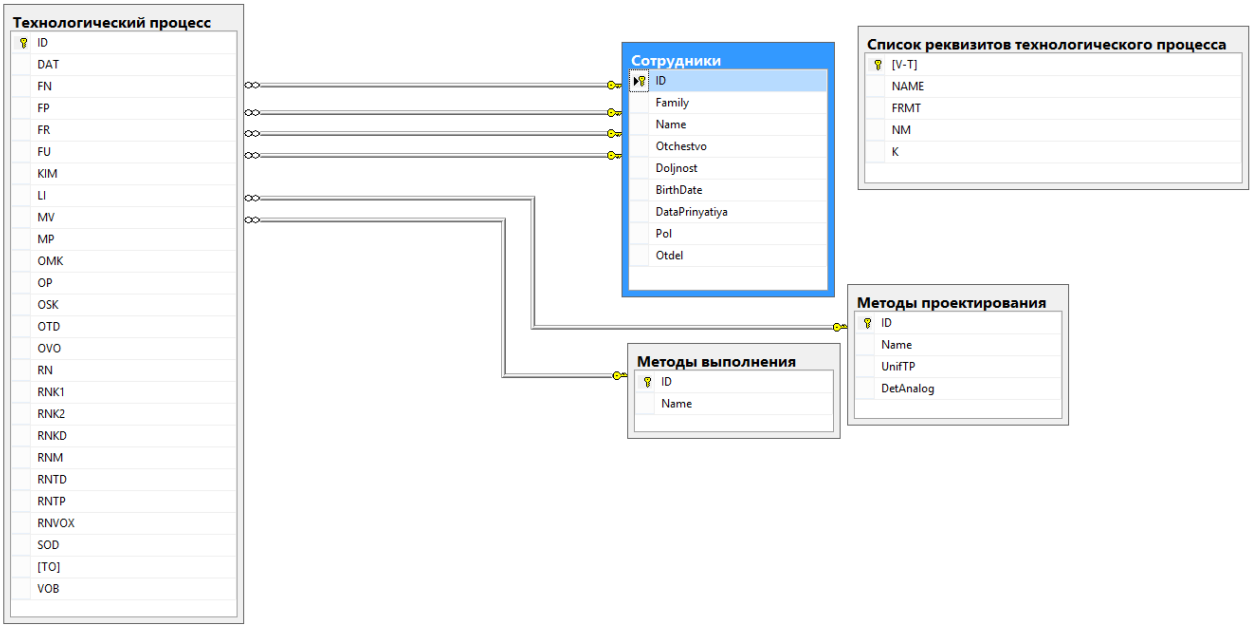


Рисунок 2.3.1 Диаграмма

Между таблицами проведены соответствующие связи «один ко многим», вспомогательная таблица (Список реквизитов технологического процесса) ни с одной таблицей не связана, но при дальнейшем расширении или использовании базы данных может быть востребована.

Рассмотрим отдельно каждую таблицу. Начнем с таблицы Технологический процесс.

Как видно из рисунка 2.3.2, таблица состоит из множества полей, описанных во входных данных, и в целом совпадает с ними по структуре. Поля, не совпадающие по типу данных с указанными в исходных данных, связаны с соответствующими таблицами-словарями. Это вызвано потребностью в уменьшении количества ошибок при вводе данных, а также с расширенными сведениями, например, таблица Сотрудники, которую мы рассмотрим следующей.

CTRLSOFT-1TJ75T\...гический процесс | CTRLSOFT-1TJ75T\...гический процесс | CTRLSOFT-1TJ75T\...ческого проц

ID	DAT	FN	FP	FR	FU	KIM
1	2015-04-03	1	2	3	4	1V555
2	2014-04-02	1	2	3	4	1V000
3	2014-03-02	1	2	3	4	0V005

Рисунок 2.3.2 Пример данных таблицы Технологический процесс

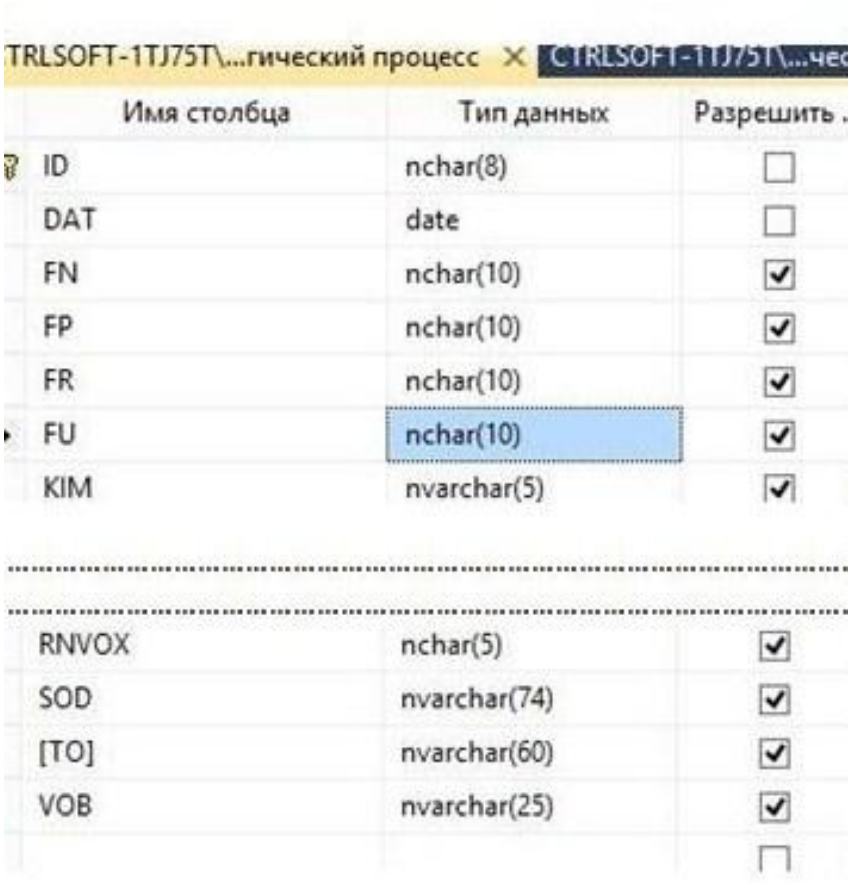


Рисунок 2.3.11 Структура таблицы Технологический процесс

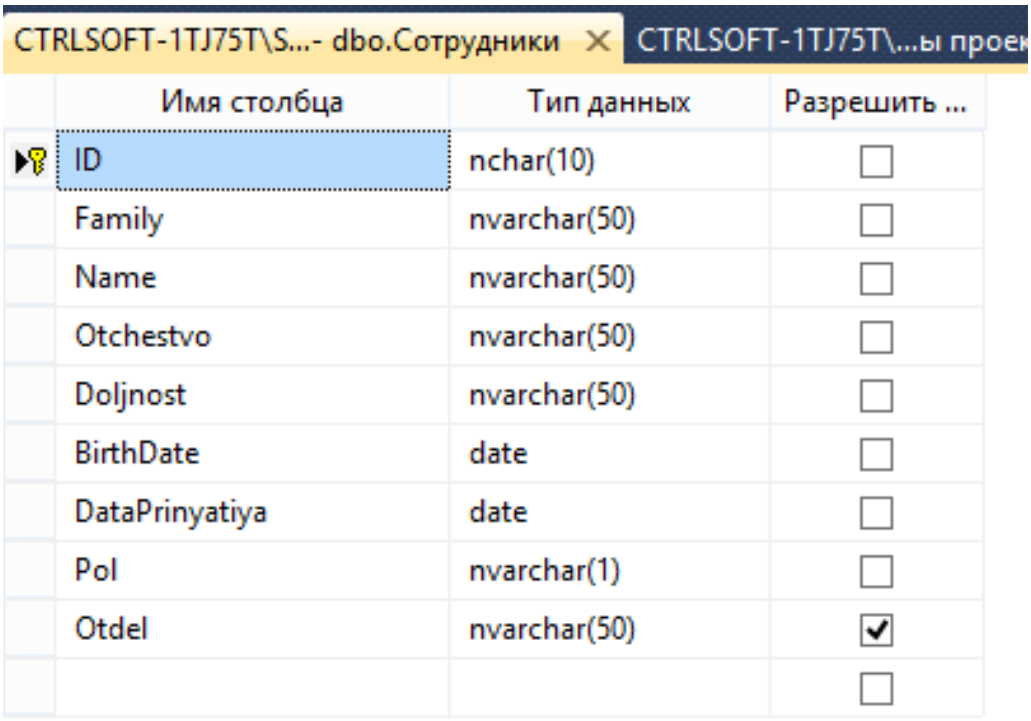


Рисунок 2.3.4 Таблица Сотрудники

Таблица Сотрудники включает в себя связь с четырьмя полями из таблицы Технологический процесс. В запросах и других видах работ с базой

данных исключается прямая связь между этими двумя таблицами, для корректной работы следует обращаться к соответствующим заранее сформированным запросам (Нормоконтролеры, Проверяющие, Разработчики, Утверждающие). Как видно из Рисунка 2.3.4, таблица состоит из основных данных по сотрудникам, стандартных для любой коммерческой организации: Фамилия, Имя, Должность, Дата рождения, Стаж и т.д.

Рассмотрим вместе таблицы по методам выполнения и методам проектирования. Они являются стандартными таблицами-справочниками (словарями), нужными для минимизирования количества ошибок оператора при вводе информации в базу данных. Таблица по методам проектирования содержит некоторые дополнительные сведения. Это поля UnifTP и DetAnalog, которые соответственно обозначают номер унифицированного технологического процесса, к которому может обращаться ТП и номер Детали аналога. Подразумевается, что эти данные берутся из других баз данных или добавляются при расширении текущей базы данных.

CTRLSOFT-1TJ75T\S...eccов - Diagram_1 | CTRLSOFT-1TJ75T...етоды выполнения

Имя столбца	Тип данных	Разрешить ...
ID	nchar(3)	☐
Name	nvarchar(50)	☐
		☐

Рисунок 2.3.12 Методы выполнения технологического процесса

CTRLSOFT-1TJ75T\...ы проектирования | CTRLSOFT-1TJ75T...етоды выполн

Имя столбца	Тип данных	Разрешить ...
ID	nchar(1)	☐
Name	nvarchar(50)	☐
UnifTP	nvarchar(50)	☑
DetAnalog	nvarchar(50)	☑
		☐

Рисунок 2.3.13 Методы технологического проектирования

CTRLSOFT-1TJ75T...етоды выполнения* | CTRLSOFT-1TJ75T\S...eccов - Diagra

ID	Name
1	Ручной
2	Кооперированно-ручной
3	механизированно-ручной (автоматизированн...
4	Автоматический

CTRLSOFT-1TJ75T...етоды выполнения* | CTRLSOFT-1TJ75T\S...ессов - Diagra

ID	Name
1	Ручной
2	Кооперированно-ручной
3	механизированно-ручной (автоматизированн...
4	Автоматический

Рисунок 2.3.14 Пример данных таблицы методы выполнения

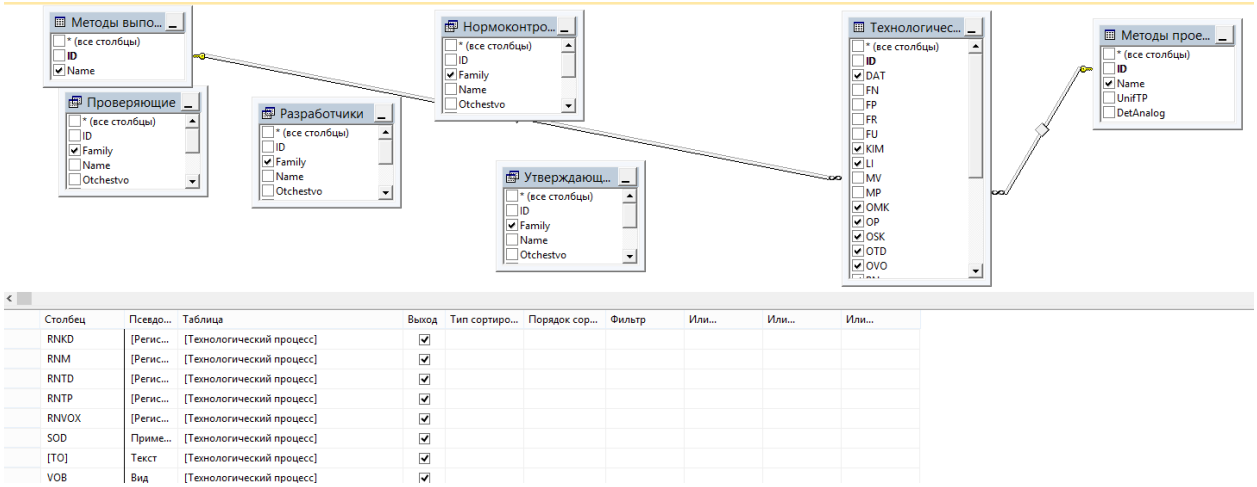
Для работы с базой данных необходимо обращаться не напрямую к таблицам, а к представлениям, сформированным на их основе. В данной работе на этот момент сформировано 5 представлений.

Для отчетности и наглядности было составлено представление, которое максимально совпадает с исходными данными задания (Рисунок 2.3.9. -10). Как видно, все столбцы названы не сокращениями, а полными, понятными названиями, скрыто поле ИД, данные о сотрудниках, методах выполнения и проектирования взяты из соответствующих справочников. Сортировка производится по ИД ТП, однако ее всегда можно изменить по мере надобности. Для демонстрации базы данных, сортировки нет.

Для построения представления использовалась дополнительные представления (Проверяющие, Разработчики и т.п.). Все 4 представления по своей структуре одинаковы и сделаны из одной и той же таблицы Сотрудники. (рис.11) Рассмотрим одно из этих представлений подробнее. Для построения представления была использована функция GROUP BYи фильтр «=Нормоконтролер». В результате, из таблицы Сотрудники берутся все нормоконтролеры:

	ID	Family	Name	Otchestvo	BirthDate	DataPrinyatiya	Pol	Otdel
▶	1	Смирнова	Галина	Олеговна	1966-02-19	1999-03-12	ж	А

Рисунок 2.3.15 Результат обращения к БД



Дата регистра...	Нормоконтро...	Проверяющий	Разработчик	Утверждающий	Коэф-т испол...	Литера	Метод проект...	Метод выпол...	Обозн. марш...	Обозн. произ...	Обозн. сопр. ...	Обозн. компл...
2015-04-03	Смирнова	Иванов	Орехов	Жмякин	1V555	F	синтеза	Ручной	3	2	A1	A1
2014-04-02	Смирнова	Иванов	Орехов	Жмякин	1V000	A	Поиск детали ...	Кооперирован...	4	1	A2	A2
2014-03-02	Смирнова	Иванов	Орехов	Жмякин	0V005	B	синтеза	механизирова...	5	8	A3	A3

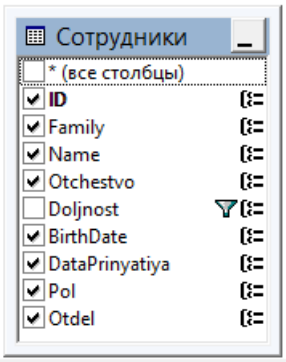
Обозн. компл...	Обозн. ведом...	Регистрацион...	Регистр. номе...	Регистр. номе...	Регистр. номе...	Регистр. номе...	Регистр. номе...	Регистр. номе...	Регистр. номе...	Примечание	Текст	Вид
A1	A1	11111	11111	11111	11111	11111	11111	11111	11111	Технологическ...	Текст	A1
A2	A2	11110	11110	11110	11110	11110	11110	11110	10011	Тех. процесс №2	текст	A8
A3	A3	11100	11100	11100	11100	11100	11100	11100	11100	Тех. процесс №3	текст	И6

Рисунок 2.3.16 Представление технологического процесса (структура)

Дата регистра...	Нормоконтро...	Проверяющий	Разработчик	Утверждающий	Коэф-т испол...	Литера	Метод проект...	Метод выпол...	Обозн. марш...	Обозн. произ...	Обозн. сопр. ...	Обозн. компл...
2015-04-03	Смирнова	Иванов	Орехов	Жмякин	1V555	F	синтеза	Ручной	3	2	A1	A1
2014-04-02	Смирнова	Иванов	Орехов	Жмякин	1V000	A	Поиск детали ...	Кооперирован...	4	1	A2	A2
2014-03-02	Смирнова	Иванов	Орехов	Жмякин	0V005	B	синтеза	механизирова...	5	8	A3	A3

Обозн. компл...	Обозн. ведом...	Регистрацион...	Регистр. номе...	Регистр. номе...	Регистр. номе...	Регистр. номе...	Регистр. номе...	Регистр. номе...	Регистр. номе...	Примечание	Текст	Вид
A1	A1	11111	11111	11111	11111	11111	11111	11111	11111	Технологическ...	Текст	A1
A2	A2	11110	11110	11110	11110	11110	11110	11110	10011	Тех. процесс №2	текст	A8
A3	A3	11100	11100	11100	11100	11100	11100	11100	11100	Тех. процесс №3	текст	И6

Рисунок 2.3.17 Результат обращения к БД



Столбец	Псевдо...	Таблица	Выход	Тип сортиро...	Порядок сор...	Group By	Фильтр
ID		Сотрудни...	☑			Group By	
Family		Сотрудни...	☑			Group By	
Name		Сотрудни...	☑			Group By	
Otchestvo		Сотрудни...	☑			Group By	
BirthDate		Сотрудни...	☑			Group By	
DataPrinyatiya		Сотрудни...	☑			Group By	
Pol		Сотрудни...	☑			Group By	
Otdel		Сотрудни...	☑			Group By	
Doljnost		Сотрудни...	☐			Group By	= N'Нормо...
			■				

```
SELECT      ID, Family, Name, Otchestvo, BirthDate, DataPrinyatiya, Pol, Otdel
FROM        dbo.Сотрудники
GROUP BY ID, Family, Name, Otchestvo, Doljnost, BirthDate, DataPrinyatiya, Pol, Otdel
HAVING      (Doljnost = N'Нормоконтролер')
```

Рисунок 2.3.18 Представление Нормоконтролер

Выводы по работе

В ходе выполнения текущей лабораторной работы было произведено знакомство с СУБД Microsoft SQL. На основе имеющейся в задании информации была создана БД “Технологический процесс”, содержащая связанные между собой таблицы “Сотрудники”, “Методы выполнения” и “Методы проектирования”. Разработана структура представления технологического процесса. Путем использования данных таблиц и примера

из задания к работе, были сгенерированы различные запросы, хранимые процедуры и отчет. Таким образом, были получены базовые навыки работы с СУБД MS SQL и общее представление о логике ее работы.

2.4. Реструктуризация данных технологического назначения»

Цель работы

Преобразовать исходную таблицу 2.4.1, придать ей более удобную для работы форму путем реструктуризации.

Постановка задачи

Данная таблица из справочника. Разработать по ней БД сверл.

Таблица 2.4.1

Сверла цилиндрические с цилиндрическим хвостовиком ГОСТ 5341-73

Сверла нормальной точности классов B1 и B						
Исполнение 1		Исполнение 2		D	L	lr
Правые	Левые	Правые	Левые			
Обозначение	Обозначение	Обозначение	Обозначение			
2300-0661	2300-3042	2300-2751	2300-3201	3.10	49	18
2300-0662	2300-3043	2300-2752	2300-3202	3.15	49	18
2300-0663	2300-3044	2300-2753	2300-3203	3.20	49	18
2300-0664	2300-3045	2300-2754	2300-3204	3.30	49	18
2300-0665	2300-3046	2300-2755	2300-3205	3.35	49	18
2300-0666	2300-3047	2300-2756	2300-3206	3.40	52	20
...	...	...	...	...	...	...

Сверла повышенной точности классов A1 и A						
Исполнение 1		Исполнение 2		D	L	lr
Правые	Левые	Правые	Левые			
Обозначение	Обозначение	Обозначение	Обозначение			
2300-5417	2300-5418	2300-5817	2300-5818	3.10	49	18
2300-5421	2300-5422	2300-5821	2300-5822	3.15	49	18
2300-5423	2300-5424	2300-5823	2300-5824	3.20	49	18
2300-5425	2300-5426	2300-5825	2300-5826	3.30	49	18
2300-5427	2300-5428	2300-5827	2300-5828	3.35	49	18
2300-5431	2300-5432	2300-5831	2300-5832	3.40	52	20
...	...	...	...	...	...	...

Кодировка:
Точность: T=1 - нормальная; T = 2 - повышенная;
Исполнение: Isp=1 и Isp=2;
Направление вращения: NV=1 - правое; NV=2 - левое

1. Разработать таблицу, включающую название инструмента, ГОСТ, направление вращения (код - название), код постоянных характеристик.
2. Разработать таблицу постоянных характеристик
3. Разработать запрос на реструктуризацию таблицы 2.4. 1, в результате которого должна получиться таблица с одним полем “Обозначение”, который и будет первичным ключом.

Ход работы

Создадим базу данных Restr и добавим в нее таблицу Sverla со следующими полями (Таблица 2.4.2):
Таблица 2.4.2. Поля таблицы Sverla

Название поля	Значение поля	Тип	Первичный ключ
Obozn	Номер сверла	nchar(10)	Да
T	Точность	numeric(1,0)	Нет
Isp	Исполнение	numeric(1,0)	Нет
NV	Направление вращения	nchar(10)	Нет
D	Диаметр	numeric(3,2)	Нет
L	Длина	numeric(2,0)	Нет
Lr	Длина режущей части	numeric(2,0)	Нет
Cpc	Код постоянной характеристики	numeric(1,0)	Нет

Затем создадим таблицу Inf, в которой будут содержаться данные о ГОСТах.

Таблица 2.4.3 . Поля таблицы Inf

Название поля	Значение поля	Тип	Первичный ключ
nomer	Номер ГОСТа в таблице	smallint	Да
name	Название ГОСТа	nvarchar(50)	Нет

С помощью команды CREATE TABLE создадим таблицу sv1, содержащую следующие столбцы (Таблица 4):

Таблица 2.4.4. Поля таблицы sv1

Название поля	Значение поля	Тип	Первичный ключ
name	Название ГОСТа	nvarchar(50)	Нет
T	Точность	numeric(1,0)	Нет
Isp	Исполнение	numeric(1,0)	Нет
NV	Направление вращения	nchar(10)	Нет
Obozn	Номер сверла	nchar(10)	Да
D	Диаметр	numeric(3,2)	Нет
L	Длина	numeric(2,0)	Нет
Lr	Длина режущей части	numeric(2,0)	Нет

Код скрипта создания представлен в листинге 2.4.1.
Затем с помощью команды INSERT INTO в созданную таблицу занесем информацию из таблиц Sverl и Inf. Соответствующий скрипт отражен в

листинге 2.4.2, а результат его работы — в таблице 4.
Листинг 2.4.1. Создание таблицы sv1

```
CREATE TABLE [dbo].[sv1](
      [name] [nvarchar](50) NOT NULL,
      [Obozn] [nchar](10) NOT NULL,
      [T] [numeric](1, 0) NOT NULL,
      [Isp] [numeric](1, 0) NOT NULL,
      [NV] [nchar](10) NOT NULL,
      [D] [numeric](3, 2) NOT NULL,
      [L] [numeric](2, 0) NOT NULL,
      [Lr] [numeric](2, 0) NOT NULL
) ON [PRIMARY]

GO
```

Листинг 2.4.2. Заполнение таблицы sv1

```
INSERT INTO [STUDENT-Restr].[dbo].[sv1]
      SELECT [name], [Obozn], [T], [Isp], [NV], [D], [L], [Lr]
      FROM [STUDENT-Restr].[dbo].[Inf], [STUDENT-
Restr].[dbo].[Sverla]
GO
            [L] FROM [STUDENT-Restr].[dbo].[Sverla],
            [Lr] FROM [STUDENT-Restr].[dbo].[Sverla]
GO
```

Таблица 2.4.5. Результат заполнения таблицы sv1

Name	Obozn	T	Isp	NV	D	L	Lr
ГОСТ 5341-73 Сверло с цилиндрическим хвостовиком	2300-0661	1	1	1	3.10	49	18
ГОСТ 5341-73 Сверло с цилиндрическим хвостовиком	2300-0662	1	1	1	3.15	49	18
ГОСТ 5341-73 Сверло с цилиндрическим хвостовиком	2300-0663	1	1	1	3.20	49	18
ГОСТ 5341-73 Сверло с цилиндрическим хвостовиком	2300-5827	2	2	1	3.35	49	18
ГОСТ 5341-73 Сверло с цилиндрическим хвостовиком	2300-5828	2	2	2	3.35	49	18
……	……	…	…	…	….	….	…
ГОСТ 5341-73 Сверло с коническим хвостовиком	2300-0661	1	1	1	3.10	49	18
ГОСТ 5341-73 Сверло с коническим хвостовиком	2300-0662	1	1	1	3.15	49	18
ГОСТ 5341-73 Сверло с коническим хвостовиком	2300-0663	1	1	1	3.20	49	18
……	……	…	….	…..	…..	…..	….
ГОСТ 5341-73 Сверло с коническим хвостовиком	2300-5422	2	1	2	3.15	49	18

Выводы по работе

Было произведено знакомство с СУБД Microsoft SQL. Путем использования данных таблиц и примера из задания к работе, были сгенерированы различные запросы, хранимые процедуры и отчет. Таким образом, получены базовые навыки реструктуризации данных в СУБД MS SQL и общее представление о логике работы.

Приложение.
Синтаксис SQL

В этой статье описывается синтаксис SQL запросов **[1]**.

В следующем операторе задана минимальная структура и синтаксис, необходимый для SELECT.

```
SELECT [DISTINCT | ALL] {* | список выбора}

FROM {table name [alias] | имя представления}
```

Ключевые слова SQL

Ключевые слова **(SELECT, GRANT, DELETE или CREATE)** прописаны в синтаксисе SQL и имеют в этом языке предопределенное значение. Можно использовать ключевые слова в верхнем или нижнем регистре. Следующие три запроса равнозначны:

```
SELECT * FROM EMPLOYEES;
Select * FROM EMPLOYEES;
select * FROM EMPLOYEES;
```

В некоторых случаях ключевые слова могут быть сокращены. Например, ключевое слово DESCRIBE может быть использовано либо в форме DESC, либо DESCRIBE. Если мы выполним следующие запросы, то в обоих случаях получим структуру таблицы сотрудников.

DESCRIBE EMP

Правила	Платформа	Описание
Идентификатор должен содержать до	SQL2003	128 символов.
	DB2	128 символов, в зависимости от платформы.
	MySQL	64 символа.
	Oracle	30 байт; имена базы данных до 8 байт.
	PostgreSQL	31 символ.
Идентификатор может содержать	SQL2003	Любые цифры, символы и нижнее подчеркивание.
	DB2	Любые цифры, символы в верхнем регистре или символ нижнего подчеркивания.
	MySQL	Любые цифры или символы.

	Oracle	Любые цифры, символы и нижнее подчеркивание (_), знак фунта стерлингов (#) или доллара ($).
	PostgreSQL	Любые цифры, символы и нижнее подчеркивание (_).
Первый символ должен быть	SQL2003	Буквой.
	DB2	Буквой.
	MySQL	Буквой или цифрой (но не должен содержать только цифры).
	Oracle	Буквой.
	PostgreSQL	Буквой или нижним подчеркиванием (_).
Идентификатор не может содержать	SQL2003	Специальные символы или пробелы.
	DB2	Специальные символы или пробелы.
	MySQL	Точку (.), слэш (/) или ASCII(0) и ASCII(255). Кавычки (') и двойные кавычки (") допускаются только в ссылающихся идентификаторах.
	Oracle	Пробелы, двойные кавычки (") или специальные символы.
	PostgreSQL	Двойные кавычки (").
В синтаксисе SQL запросов символ идентификатора	SQL2003	Двойные кавычки (").
	DB2	Двойные кавычки (").
	MySQL	Кавычки (') или двойные кавычки (") в режиме совместимости с ANSI.
	Oracle	Двойные кавычки (").
	PostgreSQL	Двойные кавычки (").
Идентификатор может быть зарезервирован	SQL2003	Нет, кроме ссылающихся идентификаторов.
	DB2	Да.
	MySQL	Нет, кроме ссылающихся идентификаторов.
	Oracle	Нет, кроме ссылающихся идентификаторов.

	PostgreSQL	Нет, кроме ссылающихся идентификаторов.
Адресация к схеме	SQL2003	Каталог.схема.объект.
	DB2	Схема.объект.
	MySQL	База_данных.объект.
	Oracle	Схема.объект.
	PostgreSQL	База_данных.схема.объект.
Идентификатор должен быть уникальным	SQL2003	Да.
	DB2	Да.
	MySQL	Да.
	Oracle	Да.
	PostgreSQL	Да.

Конвенции имен

Стандарт SQL не содержит никаких точных указаний по наименованиям, поэтому нужно следовать следующим основным принципам (*в том числе и в синтаксисе SQL запросов UPDATE*):

- Выбирайте имя, которое содержит смысл и имеет описательный характер. Например, таблица сотрудников не должна называться emp, а столбец имени сотрудника должен называться first_name, а не fname, хотя и «*emp*», и «*fname*» это допустимые идентификаторы;
- Используйте для всех объектов в базе данных SQL либо заглавные буквы, либо строчные, поскольку некоторые серверы баз данных чувствительны к регистру.

Литералы SQL

Термин литералы относится к фиксированным значениям данных. SQL распознает четыре типа литералов: числовые значения, строки символов, дата или время, логическое значение. Например, 100, -120, 544, 03, -458, 25, 3e2, 5E-2 являются действительными числовыми литералами. 'США', '2000', 'SQL Синтаксис', '1 января 1981' являются действительными строками символов (*должны быть заключены в одинарные кавычки (' ')*). Логические литералы и литералы даты/времени выглядят следующим образом: TRUE и '*JAN-28-1976 21:12:40:00*'.

Операторы

Операторы работают с отдельными элементами данных и возвращают результат. Операторы используются в различных операциях SQL, таких как SELECT, INSERT, UPDATE или DELETE. А также при создании различных объектов базы данных, таких как функции, представления, триггеры и хранимые процедуры. MS SQL синтаксис запросов поддерживает различные типы операторов, хотя не все СУБД поддерживают все операторы.

Смотрите таблицу ниже:

Операторы	Работают во
Арифметические операторы	Всех базах данных.
Операторы присвоения	Всех базах данных.
Побитовые операторы	Microsoft SQL Server.
Операторы сравнения	Всех базах данных.
Логические операторы	DB2, Oracle, SQL Server и PostgreSQL.
Унарные операторы	DB2, Oracle и SQL Server.

Приоритетность операторов

Приоритетность - это порядок, в котором база данных оценивает различные операторы в одном выражении. В синтаксисе SQL запросов при выполнении выражения, содержащего несколько операторов (например, +, -, /), сначала выполняются операторы с высшей приоритетностью, а затем с более низкой. При оценке операторов с одинаковой приоритетностью операторы выполняются в порядке их расстановки в выражении слева направо.
Если в выражении есть круглые скобки, то операторы в них вычисляется в первую очередь, а остальные части выражения, которые находятся вне скобок, вычисляются после этого. В следующей таблице перечислены уровни приоритетности операторов SQL от высокого к низкому.

Приоритетность операторов
() (выполняются в первую очередь).
+, -, ~ (унарные операторы).
*, /, % (математические операторы).
+, - (арифметические операторы).
=, >, <, >=, <=, <>, !=, !>, !< (операторы сравнения).
I^ (Побитовый OR), & (Побитовый AND), | (Побитовый OR).

NOT.
AND.
ALL, ANY, BETWEEN, IN, LIKE, OR, SOME.
= (присвоение переменных).

Приоритетность операторов

Следующие выражения в запросе MySQL возвращают разные результаты:

- SELECT 12 * 2 + 24;

12 * 2 + 24
48

- SELECT 12 * (2 + 24)

Результат

12 * (2 + 24)
312

Комментарии SQL

Комментарии в синтаксисе SQL запросов - это необязательный текст, который описывает, что делает программа и почему код был изменен. Компилятор всегда игнорирует комментарии. Комментарий вводится через двойное тире и пробел:
-- Это комментарий SQL

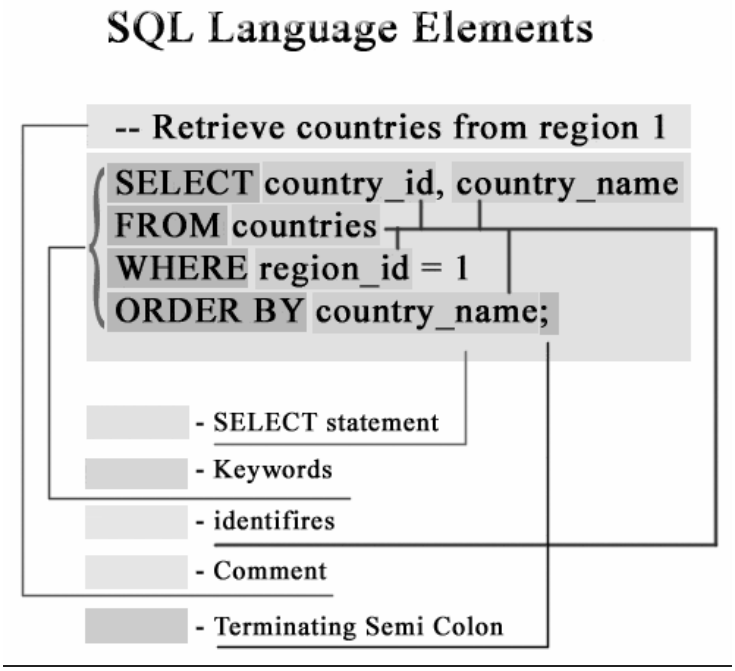
В качестве альтернативы, можно использовать блок комментариев С-стиля:

/ * Это первая строка комментария

Это вторая строка комментария * /.

Пробелы

Пробелы, как правило, игнорируются в операторах SQL, что позволяет проще форматировать код для удобства чтения. На следующей диаграмме приведены элементы синтаксиса SQL запросов, которые составляют одиночный оператор:



Список ключевых слов SQL:

ABSOLUTE	ACTION	ADD	ADMIN
AFTER	AGGREGATE	ALIAS	ALL
ALLOCATE	ALTER	AND	ANY
ARE	ARRAY	AS	ASC
ASSERTION	ASSERTION	AT	ATOMIC
AUTHORIZATION	BEFORE	BEGIN	BIGINT
BINARY	BIT	BLOB	BOOLEAN
BOTH	BREADTH	BY	CALL
CASCADE	CASCADED	CASE	CAST
CATALOG	CHAR	CHARACTER	CHECK
CLASS	CLOB	CLOSE	COLLATE
COLLATION	COLLECT	COLUMN	COMMIT
COMPLETION	CONDITION	CONNECT	CONNECTION
CONSTRAINT	CONSTRAINTS	CONSTRUCTOR	CONTAINS
CONTINUE	CORRESPONDING	CREATE	CROSS
CUBE	CURRENT	CURRENT_DATE	CURRENT_PATH
CURRENT_ROLE	CURRENT_TIME	CURRENT_TIMESTAMP	CURRENT_USER
CURSOR	CYCLE	DATA	DATALINK

DATE	DAY	DEALLOCATE	DEC
DECIMAL	DECLARE	DEFAULT	DEFERRABLE
DELETE	DEPTH	DEREF	DESC
DESCRIPTOR	DESTRUCTOR	DIAGNOSTICS	DICTIONARY
DISCONNECT	DO	DOMAIN	DOUBLE
DROP	ELEMENT	END-EXEC	EQUALS
ESCAPE	EXCEPT	EXCEPTION	EXECUTE
EXIT	EXPAND	EXPANDING	FALSE
FIRST	FLOAT	FOR	FOREIGN
FREE	FROM	FUNCTION	FUSION
GENERAL	GET	GLOBAL	GOTO
GROUP	GROUPING	HANDLER	HASH
HOUR	IDENTITY	IF	IGNORE
IMMEDIATE	IN	INDICATOR	INITIALIZE
INITIALLY	INNER	INOUT	INPUT
INSERT	INT	INTEGER	INTERSECT
INTERSECTION	INTERVAL	INTO	IS
ISOLATION	ITERATE	JOIN	KEY
LANGUAGE	LARGE	LAST	LATERAL
LEADING	LEAVE	LEFT	LESS
LEVEL	LIKE	LIMIT	LOCAL
LOCALTIME	LOCALTIMESTAMP	LOCATOR	LOOP
MATCH	MEMBER	MEETS	MERGE
MINUTE	MODIFIES	MODIFY	MODULE
MONTH	MULTISET	NAMES	NATIONAL
NATURAL	NCHAR	NCLOB	NEW
NEXT	NO	NONE	NORMALIZE
NOT	NULL	NUMERIC	OBJECT
OF	OFF	OLD	ON
ONLY	OPEN	OPERATION	OPTION
OR	ORDER	ORDINALITY	OUT
OUTER	OUTPUT	PAD	PARAMETER

PARAMETERS	PARTIAL	PATH	PERIOD
POSTFIX	PRECEDES	PRECISION	PREFIX
PREORDER	PREPARE	PRESERVE	PRIMARY
PRIOR	PRIVILEGES	PROCEDURE	PUBLIC
READ	READS	REAL	RECURSIVE
REDO	REF	REFERENCES	REFERENCING
RELATIVE	REPEAT	RESIGNAL	RESTRICT
RESULT	RETURN	RETURNS	REVOKE
RIGHT	ROLE	ROLLBACK	ROLLUP
ROUTINE	ROW	ROWS	SAVEPOINT
SCHEMA	SCROLL	SEARCH	SECOND
SECTION	SELECT	SEQUENCE	SESSION
SESSION_USER	SET	SETS	SIGNAL
SIZE	SMALLINT	SPECIFIC	SPECIFICTYPE
SQL	SQLEXCEPTION	SQLSTATE	SQLWARNING
START	STATE	STATIC	STRUCTURE
SUBMULTISET	SUCCEEDS	SUM	SYSTEM_USER
TABLE	TABLESAMPLE	TEMPORARY	TERMINATE
THAN	THEN	TIME	TIMESTAMP
TIMEZONE_HOUR	TIMEZONE_MINUTE	TO	TRAILING
TRANSACTION	TRANSLATION	TREAT	TRIGGER
TRUE	UESCAPE	UNDER	UNDO
UNION	UNIQUE	UNKNOWN	UNTIL
UPDATE	USAGE	USER	USING
VALUE	VALUES	VARCHAR	VARIABLE
VARYING	VIEW	WHEN	WHENEVER
WHERE	WHILE	WITH	WRITE
YEAR	ZONE		

Литература

1. Сайтостроение / Статьи / Базы данных / SQL [Электронный ресурс].– Режим доступа: http://www.internet-technologies.ru/articles/article_2931.html
2. Объединение UNION и UNION ALL в SQL – описание и примеры [Электронный ресурс].– Режим доступа: http://info-comp.ru/obucheniest/340-sql-union-and-union-all.html
3. Агрегатные функции [Электронный ресурс].– Режим доступа: https://msdn.microsoft.com/ru-ru/library/ms173454.aspx
4. Инструкция CREATE TABLE (Transact-SQL) [Электронный ресурс].– Режим доступа: https://msdn.microsoft.com/ru-ru/library/ms174979.aspx
5. ALTER TABLE (Transact-SQL) [Электронный ресурс].– Режим доступа: https://msdn.microsoft.com/ru-ru/library/ms190273.aspx
6. SELECT (Transact-SQL) [Электронный ресурс].– Режим доступа: https://msdn.microsoft.com/ru-ru/library/ms189499.aspx
7. Синтаксис SQL 2003 [Электронный ресурс].– Режим доступа: http://sd-company.su/article/sql/operator_all_any_some
8. SQL. Команда GRANT [Электронный ресурс].– Режим доступа: http://sql-language.ru/grant.html
9. SQL. Команда REVOKE [Электронный ресурс].– Режим доступа: https://sql-language.ru/revoke.html
10. DECLARE CURSOR (Transact-SQL) [Электронный ресурс].– Режим доступа:https://msdn.microsoft.com/ru-ru/library/ms180169.aspx
11. Управление, использование, выборка циклом в курсорах T-SQL MS SQL Server [Электронный ресурс].– Режим доступа: http://snakeproject.ru/rubric/article.php?art=tsql_cursor_2012
12. Курсоры языка Transact-SQL [Электронный ресурс].– Режим доступа: https://technet.microsoft.com/ru-ru/library/ms190028(v=sql.105).aspx
13. Представление [Электронный ресурс].– Режим доступа: http://housecomputer.ru/programs/sql/ch20.html
14. Изменение значений с помощью представлений [Электронный ресурс].– Режим доступа:http://housecomputer.ru/programs/sql/ch21.
15. Астахова, И.Ф. СУБД: язык SQL в примерах и задачах. [Электронный ресурс] / И.Ф. Астахова, В.М. Мельников, А.П. Толстобров, В.В. Фертиков. — Электрон. дан. — М. : Физматлит, 2009. — 168 с. — Режим доступа: http://e.lanbook.com/book/2101 — Загл. с экрана.
16. Зудилова, Т.В. Создание запросов в Microsoft SQL Server 2008. [Электронный ресурс] / Т.В. Зудилова, Г.Ю. Шмелева. — Электрон.дан. — СПб. : НИУ ИТМО, 2013. — 149 с. — Режим доступа: http://e.lanbook.com/book/43576 — Загл. с экрана.

УНИВЕРСИТЕТ ИТМО

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА ТЕХНОЛОГИИ ПРИБОРОСТРОЕНИЯ

Кафедра технологии приборостроения относится к числу ведущих кафедр института со дня его основания в 1931 году. Тогда она называлась кафедрой механической технологии и возглавлялась известным ученым в области разработки инструмента профессором Александром Павловичем Знаменским. Позже она была переименована в кафедру технологии приборостроения.

За время своего существования кафедра выпустила из стен института более тысячи квалифицированных инженеров, более сотни кандидатов и докторов наук. В разные годы ее возглавляли известные ученые и педагоги профессора Николай Павлович Соболев и Сергей Петрович Митрофанов.

Кафедра имеет выдающиеся научные достижения. Заслуженным деятелем науки и техники РСФСР, профессором С.П. Митрофановым были разработаны научные основы группового производства, за что он был удостоен Ленинской премии СССР. Методы группового производства с успехом применяются в промышленности и постоянно развиваются его учениками. Заслуженным деятелем науки и техники РСФСР, Заслуженным изобретателем СССР Юрием Григорьевичем Шнейдером разработаны метод и инструментарий нанесения регулярного микрорельефа на функциональной поверхности.

В настоящее время кафедра осуществляет выпуск бакалавров и магистров по направлениям 12.03.01, 12.04.01 "Приборостроение" и 09.03.01, 09.04.01 "Информатика и вычислительная техника".

А.Н. Филиппов

ПРИМЕНЕНИЕ ЯЗЫКА ЗАПРОСОВ SQL В САПР ТП

Учебное пособие

В авторской редакции

Редакционно-издательский отдел Университета ИТМО

Зав. РИО Н.Ф. Гусарова

Тираж 50 экз.

Отпечатано на ризографе