Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Институт информационных технологий

Кафедра информационных систем и технологий

О. Ю. Кунцевич

БАЗЫ ДАННЫХ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве пособия для специальности 1-40 01 01 «Программное обеспечение информационных технологий»

Минск БГУИР 2023

УДК 004.65(076.5)
ББК 32.973.26-018.2я73
К91

Р е ц е н з е н т ы:

кафедра последипломного образования учреждения образования
«Белорусская государственная академия связи»
(протокол №4 от 30.12.2021);

доцент кафедры экономической информатики учреждения образования
«Белорусский государственный экономический университет»
кандидат технических наук, доцент В. А. Новиков

Кунцевич, О. Ю.

К91 Базы данных. Лабораторный практикум : пособие / О. Ю. Кунцевич. – Минск : БГУИР, 2023. – 83 с. : ил.

ISBN 978-985-543-691-2.

Состоит из одиннадцати лабораторных работ по следующим темам учебной программы: «Основы баз данных», «Основные понятия реляционной модели баз данных», «Нормализация и нормальные формы», «Проектирование баз данных», «Использование языка структурированных запросов». Каждая лабораторная работа включает краткие теоретические сведения, примеры выполнения заданий, рекомендуемый порядок выполнения, а также требования к содержанию отчета для ее защиты.

УДК 004.65(076.5)
ББК 32.973.26-018.2я73

ISBN 978-985-543-691-2

© Кунцевич О. Ю., 2023
© УО «Белорусский государственный университет информатики и радиоэлектроники», 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Изучение дисциплины «Базы данных» является необходимым компонентом в образовании студентов, связанных с информационными технологиями. Успешное усвоение данного предмета позволит усовершенствовать обучающимся навыки разработки профессиональных программных средств, отвечающих современному этапу развития компьютерной техники, а следовательно, будет способствовать будущим специалистам быть более конкурентоспособными на рынке труда.

Пособие предусматривает ознакомление с основами баз данных, их проектированием, языком структурированных запросов SQL. Реализация работы с базами данных предлагается с помощью системы управления базами данных (СУБД) SQL Server. Материал пособия содержит базовые сведения по теории баз данных. Для углубленного изучения предмета рекомендуем обратить внимание на литературу из списка использованных источников.

Данное издание может быть использовано студентами всех форм обучения, но в первую очередь предназначено для студентов заочной формы обучения для получения высшего образования, интегрированного со средним специальным образованием. В этом случае часть материала – первые пять лабораторных работ – следует применить для подготовки к занятиям в качестве самостоятельной работы.

Пособие строго следует учебно-программной документации, цели лабораторных работ совпадают с их содержанием, указанным в учебной программе по дисциплине.

ЛАБОРАТОРНАЯ РАБОТА №1
ИЗУЧЕНИЕ ПРИНЦИПОВ РАБОТЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

Цель работы: изучить основные понятия о реляционных базах данных и принципы работы с ними.

Теоретические сведения

Данные окружают нас повсюду: включаем радио, слышим разговор людей в автобусе по пути на учебу, включаем компьютер и загружаем домашнюю страницу и т. д. Все они имеют для нас значение. Те данные, которые нужны нам для реализации идей, выполнения процессов, обрабатываются, осмысливаются и анализируются для дальнейших действий. Эти данные должны каким-то образом фиксироваться, а для последующей работы с ними – систематизироваться.

Такие данные целесообразно организовывать с помощью определенных структур, которые называют *базами данных*.

Эти структуры должны соответствовать определенным требованиям, в частности, обладать полнотой и непротиворечивостью данных (целостность базы данных), возможностью осуществления поиска данных по запросам пользователя и др. [1–3].

База данных – совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между соответствующими сущностями и поддерживающей одну или более областей применения. Любая база данных должна максимально соответствовать следующим требованиям: адекватность предметной области; удобство использования; производительность; защищенность данных [2, с. 7–18].

Для хранения и обработки данных с помощью компьютера применяют специальные программы – системы управления базами данных (СУБД).

В данном пособии рассматриваются такие базы данных, которые хранят информацию в специально структурированных объектах, имеющих вид двумерных таблиц, спроектированных и связанных между собой по определенным правилам. Такие базы данных называют *реляционными*.

Наиболее распространенными являются следующие СУБД:

– MS SQL Server;
– My SQL;
– PostgreSQL;
– Oracle;
– SQLite и др.

В лабораторных работах №4–11 будем рассматривать работу с реляционными базами данных с помощью СУБД MS SQL Server (вся документация по выбору версии СУБД, установке и работе с ней есть на официальном сайте компании Microsoft [3], для работы на занятиях можно использовать любую реляционную СУБД).

Для работы с реляционной базой данных информацию необходимо представить в виде двумерной таблицы, которая позволит эффективно провести анализ данных, осуществить поиск информации, соответствующей условиям отбора.

А что такое таблица? В Большом Советском словаре читаем: «Таблица – перечень сведений, числовых данных, приведенных в определенную систему и разнесенных по графам; сводка, ведомость» [4, с. 1300].

Рассмотрим пример, для чего возьмем фрагмент поэмы-сказки А. С. Пушкина «Руслан и Людмила»: «У лукоморья дуб зеленый …». Попробуем оформить этот отрывок произведения в виде таблицы (таблица 1).

Таблица 1 – Систематизация данных, полученных из фрагмента произведения А. С. Пушкина, и представление их в виде таблицы

№ п/п	Персонаж	Описание (характеристика)	Место событий
1	Дуб зеленый	Находится у лукоморья	Русь
2	Кот ученый	Ходит по цепи, песнь заводит, сказку говорит	Русь
3	Леший	Бродит	Русь
4	Русалка	На ветвях сидит	Русь
5	Избушка	Стоит на курьих ножках, без окон, без дверей	Русь
6	Тридцать витязей	Прекрасные	Русь
7	Дядька морской	Выходит из вод вместе с витязями	Русь
8	Королевич	Пленяет царя	Русь
9	Колдун	Несет богатыря через леса, через моря	Русь
10	Богатырь	–	Русь
11	Царевна	Тужит в темнице	Русь
12	Бурый волк	Служит царевне	Русь
13	Баба Яга	Летает в ступе	Русь
14	Кащей	Чахнет над златом	Русь
15	Автор	Был в сказочном лукоморье и там пил мед	Русь

Получилась двумерная таблица со следующими характеристиками: строка заголовка, пятнадцать строк с данными, четыре столбца. Можно заметить, что последний столбец включает повторяющиеся значения – «Русь». Далее в пособии будет рассмотрено, каким образом оптимизировать такие структуры, на данном этапе этот столбец необходим, чтобы не потерять информацию.

Что может дать такая систематизация? Даже на данном этапе возможно провести некоторый анализ данных:

– подсчитать количество персонажей;

– найти персонаж по его характеристике и наоборот;

– продолжить систематизацию персонажей, например, по категориям: «положительный – отрицательный» или «занят делом – бездельник», «женского пола – мужского пола» и др.;

– отсортировать по алфавиту или в обратном порядке и др.

Таким образом, представление данных в структурированном виде – таблице – многократно увеличивает возможности обработки информации, ускоряет этот процесс.

Работа с реляционными базами данных опирается на конкретные математические понятия, свойства, операции, которые описаны в дискретной математике, теории множеств. Но здесь имеются свои особенности, которые реализуются в реляционной алгебре.

К таким стандартным операциям над множествами относят: объединение, пересечение, разность, декартово произведение. В реляционной алгебре их обычно называют ***теоретико-множественными*** [5, с. 56].

На рисунке 1 изображены множества A и B, состоящие из элементов, некоторые из которых могут совпадать, а также результаты их пересечения, объединения и разности A – B.

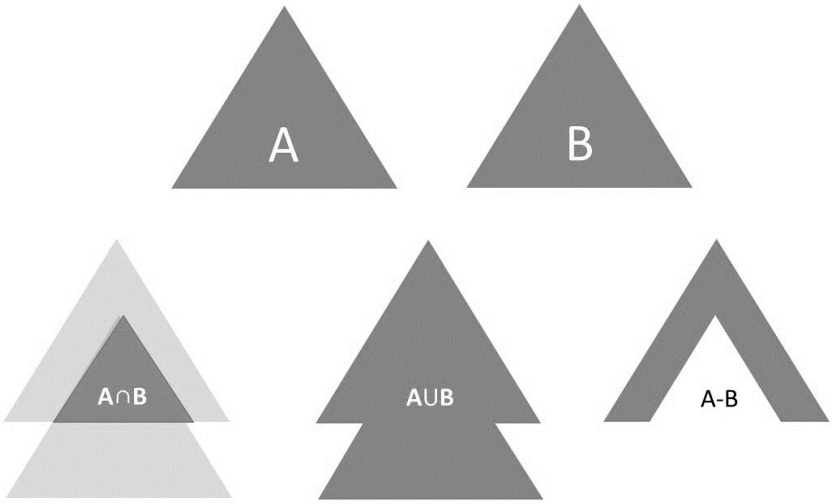


Рисунок 1 – Операции пересечения, объединения и разности множеств (результат операций выделен темным цветом)

Теория реляционных баз данных дополнена одним из ее основоположников – Эдгаром Коддом – еще четырьмя операциями, которые называют ***специальными***: проекция, ограничение (выборка), соединение, деление [5, с. 56].

!!! ЗАМЕЧАНИЕ. *Если строго следовать терминологии, то необходимо заметить, что теоретико-множественные операции – классические, базовые. Они могут реализовываться над произвольными множествами. В своей реляционной интерпретации они имеют особенности применения от классической версии в математике. Специальные операции Э. Кодда относятся непосредственно к реляционной модели данных* .

Рассмотрим наиболее часто применяемые операции и покажем, как они реализуются в теории реляционных баз данных, в которых главными объектами являются отношения, представленные в виде двумерных таблиц.

Следующие четыре операции представлены в их математической версии. В основу положены классические определения этих операций в математике [8, с. 9–11; 9, с. 21–25].

Пересечение (A ∩ B). Результат пересечения множеств A, B – множество всех элементов, которые принадлежат и множеству A, и множеству B одновременно, т. е. элементы, которые являются общими для обоих множеств (рисунок 2).

Col_1
Один
Два
Три
Четыре
Пять

∩

Col_1
Один
Два
Три
Чотири
П'ять

=

Col_1
Один
Два
Три

Рисунок 2 – Результат операции пересечения двух множеств (в первом указаны цифры от 1 до 5 на русском языке, во втором – на украинском)

Объединение (A ∪ B). Результат объединения множеств A, B – множество всех элементов, которые принадлежат или множеству A, или множеству B (хотя бы одному из этих множеств). Следует отметить, что результат будет содержать все строки таблицы A и все строки таблицы B. Повторяющиеся строки в результате не должны дублироваться (рисунок 3).

Col_1	Col_2
Зима	Зіма
Весна	Вясна
Лето	Лета
Осень	Восень

∪

Col_1	Col_2
Зима	Зима
Весна	Весна
Лето	Літо
Осень	Осінь

=

Col_1	Col_2
Зима	Зіма
Весна	Вясна
Лето	Лета
Осень	Восень
Зима	Зима
Весна	Весна
Лето	Літо
Осень	Осінь

Рисунок 3 – Результат объединения двух множеств (в первом указаны названия времен года на русском и белорусском языках, во втором – на русском и украинском языках)

Разность (A – B). Результат разности множеств A, B – множество всех элементов, которые принадлежат множеству A и не принадлежат множеству B. Отметим, что результат операции (A – B) далеко не всегда совпадает с разностью (B – A). На рисунке 4 изображен результат разности двух множеств.

а

Col_1
Один
Два
Три
Четыре
Пять

−

Col_1
Один
Два
Три
Чотири
П'ять

=

Col_1
Четыре
Пять

б

Col_1
Один
Два
Три
Чотири
П'ять

−

Col_1
Один
Два
Три
Четыре
Пять

=

Col_1
Чотири
П'ять

Рисунок 4 – Результат разности множеств A – B (*а*) и B – A (*б*) (множества содержат цифры от 1 до 5 на русском и украинском языках)

Декартово произведение (A × B). Результат этой операции – множество упорядоченных пар, в каждой из которых первый элемент принадлежит множеству A, второй – множеству B. На рисунке 5 приведен пример выполнения этой операции.

A.Col_1	A.Col_2
Зима	Зіма
Весна	Вясна
Лето	Лета
Осень	Восень

×

B.Col_1	B.Col_2
Зима	Зима
Весна	Весна
Лето	Літо
Осень	Осінь

=

A.Col_1	A.Col_2	B.Col_1	B.Col_2
Зима	Зіма	Зима	Зима
Зима	Зіма	Весна	Весна
Зима	Зіма	Лето	Літо
Зима	Зіма	Осень	Осінь
Весна	Вясна	Зима	Зима
Весна	Вясна	Весна	Весна
Весна	Вясна	Лето	Літо
Весна	Вясна	Осень	Осінь
Лето	Лета	Зима	Зима
Лето	Лета	Весна	Весна
Лето	Лета	Лето	Літо
Лето	Лета	Осень	Осінь
Осень	Восень	Зима	Зима
Осень	Восень	Весна	Весна
Осень	Восень	Лето	Літо
Осень	Восень	Осень	Осінь

Рисунок 5 – Результат выполнения операции декартова произведения двух множеств (в первом указаны названия времен года на русском и белорусском языках, во втором – на русском и украинском языках)

Для рассмотренных выше операций элементом множества – таблицы – является строка (запись) таблицы.

Понятия «отношение», «кортеж» и «атрибут» будут рассмотрены в лабораторной работе №2. Однако сейчас для понимания следующих двух определений упростим их и сведем к аналогии: отношение – таблица, кортеж – строка таблицы, атрибут – ее столбец.

Ограничение (выборка) – операция, отбирающая кортежи отношения в соответствии с условием (которое задается, в частности, после слова Where).

Выполним операцию выборки над множеством, представленным как результат на рисунке 5:

(A × B) **Where** (A.Col_1 = B.Col_1)

Результат выполнения изображен на рисунке 6.

A.Col_1	A.Col_2	B.Col_1	B.Col_2
Зима	Зіма	Зима	Зима
Весна	Вясна	Весна	Весна
Лето	Лета	Лето	Літо
Осень	Восень	Осень	Осінь

Рисунок 6 – Результат выполнения операции выборки

Проекция (Projection) – операция, которая проводится над одним отношением (таблицей) R. Ее результат – новое отношение (таблица), содержащее вертикальное подмножество отношения R, создаваемое посредством извлечения значений указанных атрибутов и исключения из результата строк-дубликатов (кортежей-дубликатов) [6]. При отсутствии иных условий сохранятся все соответствующие этим атрибутам (столбцам таблицы) данные.

Выполним операцию проекции над множеством, представленным в результате операции выборки на рисунке 6:

(A × B **Where** A.Col_1 = B.Col_1) **Projection** {A.Col_1, A.Col_2, B.Col_2**}**

Результат выполнения изображен на рисунке 7.

A.Col_1	A.Col_2	B.Col_1	B.Col_2
Зима	Зіма	Зима	Зима
Весна	Вясна	Весна	Весна
Лето	Лета	Лето	Літо
Осень	Восень	Осень	Осінь

Projection

=

A.Col_1	A.Col_2	B.Col_2
Зима	Зіма	Зима
Весна	Вясна	Весна
Лето	Лета	Літо
Осень	Восень	Осінь

Рисунок 7 – Результат выполнения операции проекции

Соединение – операция, которая является обратной к операции проекции. При соединении двух отношений по некоторому условию образуется результирующее отношение, кортежи которого производятся путем

объединения кортежей первого и второго отношений и удовлетворяют этому условию [5, с. 57].

Деление – отношение, которое содержит набор кортежей отношения R, определенных на множестве атрибутов C, которые соответствуют комбинации всех кортежей отношения S [6].

Порядок выполнения лабораторной работы

1. Изучите теоретические сведения, приведенные в пособии.
2. Изучите требования, предъявляемые к отношениям, для применения к ним реляционных операций по источнику [7, с. 249–255].
3. Выполните задания в соответствии со своим вариантом (приложение А). При необходимости доработайте отношения (таблицы) для возможности выполнения необходимых реляционных операций.
4. Представьте результат выполнения операций над отношениями (скобки устанавливают порядок операций (см. приложение А)).

Содержание отчета

1. Цель работы.
2. Краткие теоретические сведения.
3. Графические (табличные) результаты выполнения заданий с текстовым пояснением (при необходимости).
4. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №2
ФОРМИРОВАНИЕ НАБОРА ОТНОШЕНИЙ И ИХ АТРИБУТОВ

Цель работы: сформировать у студентов теоретические знания о понятии «отношение», его компонентах, а также практические навыки в области отражения объектов реального мира с помощью отношений реляционных баз данных.

Теоретические сведения

Реляционная модель данных основана на таком понятии реляционной алгебры, как отношение. Физическим представлением отношения в реляционной базе данных является таблица, состоящая из строк и столбцов, а также имеющая «шапку» – названия столбцов.

Сущность – любой различимый объект, который может быть представлен в базе данных; то, о чем необходимо хранить информацию [7, с. 49–55].

Экземпляр сущности – конкретный объект, характеризующийся набором значений атрибутов сущности [1, с. 10]

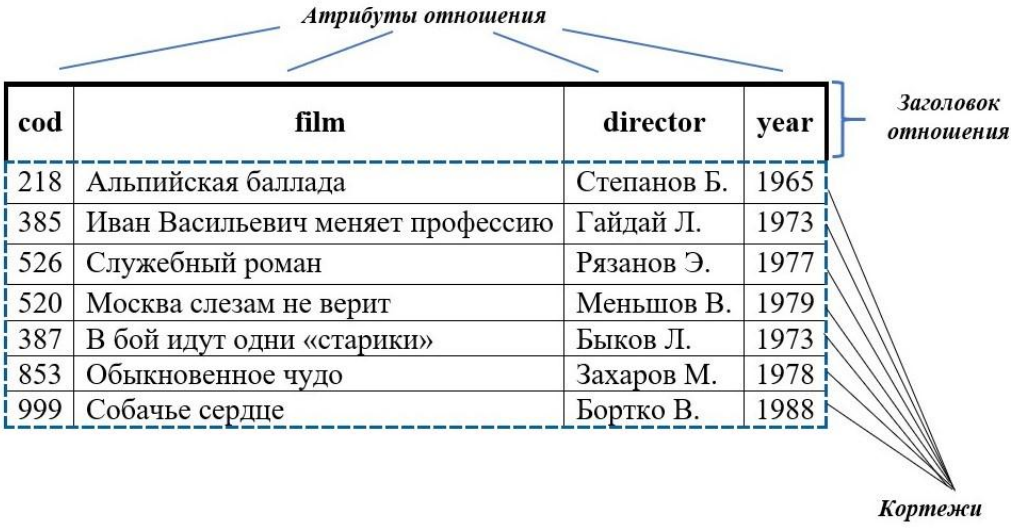
Под отношением будем понимать множество кортежей (записей, строк таблицы), обладающих одинаковым набором атрибутов (свойств, полей, столбцов таблицы) [2, с. 22].

Итак, любое отношение состоит из атрибутов и кортежей, как таблица состоит из столбцов и строк.

Атрибутом называют именованное свойство сущности (отношения), *кортежем* – часть отношения, представляющую собой уникальную взаимосвязанную комбинацию значений, каждое из которых соответствует своему атрибуту [2, с. 23].

Рассмотрим компоненты отношения на примере таблицы (отношения) «Советские фильмы» (рисунок 8).

Soviet_movies



cod	film	director	year
218	Альпийская баллада	Степанов Б.	1965
385	Иван Васильевич меняет профессию	Гайдай Л.	1973
526	Служебный роман	Рязанов Э.	1977
520	Москва слезам не верит	Меньшов В.	1979
387	В бой идут одни «старики»	Быков Л.	1973
853	Обыкновенное чудо	Захаров М.	1978
999	Собачье сердце	Бортко В.	1988

Рисунок 8 – Компоненты отношения (на примере отношения *Soviet_movies*)

Заголовок отношения включает названия атрибутов и вместе с их характеристиками (ограничения, типы данных и др.) формирует схему отношения.

!!! ЗАМЕЧАНИЕ. *Для дальнейшей работы сформулируем некоторые «свои» правила именования атрибутов и отношений:*

– *имена отношений будем записывать с первой прописной буквы, если они состоят из нескольких слов – разделять слова знаком подчеркивания (например, Soviet_movies);*

– *имена атрибутов будем записывать строчными буквами;*

– *имена отношений и атрибутов будем записывать латинскими (английскими) буквами.*

Важными для работы с реляционными базами данных являются понятия «*тип данных*» и «*домен*». Понимание *типа данных* строится на его свойствах. Тип данных определяет:

– множество значений данного типа данных;

– набор операций, применимых к значениям типа данных;

– способ внешнего представления значений типа данных (литералов).

В современных реляционных базах данных допускается работа:

– с числовыми данными;

– символьными данными;

– специализированными числовыми данными (например, денежными);

– специальными типами данных (например, дата/время);

– пользовательскими типами данных.

Домен данных представляет собой набор всех возможных значений атрибута отношения [2, с. 23].

Примерами этих понятий в отношении, изображенном на рисунке 8, являются:

– домен – набор номеров ячеек (коды ячеек), где хранятся пленки с кинофильмами;

– тип данных этих значений (номеров) – числовой.

!!! ЗАМЕЧАНИЕ. *В данном примере есть еще один атрибут, значения которого имеют числовой тип данных, – год. Однако сравнить два числовых значения – год создания фильма и номер (код) ячейки – невозможно по смыслу (семантике) этих значений. Почему? Потому что они относятся к разным доменам.*

Порядок выполнения лабораторной работы

1. Охарактеризуйте предметную область «Организация, в которой я работаю» (опишите структуру, виды работ, исполнителей и др.).

2. По источнику 6 (тема 4 «Реляционная модель данных») рассмотрите свойства отношений. Ответьте на вопрос: любая ли таблица является отношением?

3. Сформируйте одно отношение (таблицу) A, соответствующее полученному описанию предметной области. Дайте ему имя. Отношение должно отвечать следующим характеристикам:

– количество атрибутов отношения – не менее 10;

– количество кортежей отношения – 7;

– наличие следующих типов данных: числовой, символьный, дата/время;

– набор значений конкретного атрибута – отдельный домен;

– наличие разных доменов (не менее трех) с одинаковым типом данных.

4. Заполните отношение (таблицу) данными.

5. Укажите для значений атрибутов самые очевидные ограничения (уникальность, отсутствие или возможность NULL-значений и др.).

6. Оформите отчет о выполненной работе.

Содержание отчета

1. Цель работы.

2. Краткие теоретические сведения.

3. Графические (табличные) результаты выполнения заданий с текстовым пояснением.

4. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №3
ФОРМИРОВАНИЕ НАБОРА КЛЮЧЕЙ

Цель работы: сформировать понятие ключа, изучить виды ключей, закрепить полученные знания на практике.

Теоретические сведения

Потенциальный ключ – атрибут (или совокупность атрибутов), который мог бы стать первичным ключом.

Первичный ключ (Primary Key, PK) – атрибут (или совокупность атрибутов), который выбран из всех потенциальных ключей в качестве уникального идентификатора кортежа отношения (строки таблицы).

Те потенциальные ключи, которые не стали первичным ключом, называются *альтернативными ключами*.

Так, в качестве потенциальных ключей отношения Soviet_movies (см. рисунок 8) могут быть выбраны атрибуты cod и film (определим, что названия фильмов в этой базе данных должны быть уникальными). При продолжении таблицы будут указаны другие фильмы – их названия не повторятся, как и номера ячеек (коды ячеек), где хранятся пленки с кинофильмами (одна ячейка – одна пленка). Но могут повториться режиссеры и годы выпуска добавленных картин. Выберем в качестве первичного ключа поле cod.

Из определения первичного ключа следуют его основные характеристики [2]:

– значения первичного ключа уникальны и не могут повторяться;

– значения первичного ключа не могут быть пустыми (NULL-значениями);

– первичный ключ должен быть несократимым;

– первичный ключ должен быть минимальным.

Пусть на киностудии прошла модернизация хранилищ и теперь номер ячейки, в которой лежит кинопленка, определяется следующей комбинацией: номер архива – номер полки – код ячейки. Дополним отношение Soviet_movies соответствующими атрибутами (таблица 2).

Таблица 2 – Отношение Soviet_movies

archive	shelf	cod	film	director	year
1	782	10	Альпийская баллада	Степанов Б.	1965
1	782	15	Иван Васильевич меняет профессию	Гайдай Л.	1973
2	782	10	Служебный роман	Рязанов Э.	1977
3	310	4	Москва слезам не верит	Меньшов В.	1979
1	513	10	В бой идут одни «старики»	Быков Л.	1973
4	205	43	Обыкновенное чудо	Захаров М.	1978
2	205	43	Собачье сердце	Бортко В.	1988

Теперь только по коду ячейки невозможно определить, пленка с каким фильмом в ней находится. Для однозначного установления соответствующего события необходимо знать и номер архива, и номер полки в нем, и непосредственно код ячейки на этой полке в этом архиве. То есть идентификация записи происходит по значению не одного атрибута, а комбинации значений трех атрибутов, которая является уникальной в рамках данного отношения.

Следовательно, в данном случае первичный ключ состоит из нескольких атрибутов. Такой ключ называется *составным*. Соответственно ключ, состоящий из одного атрибута, называют *простым*.

Возможно ли в комбинации составного ключа убрать один из атрибутов? Нет, поскольку все другие комбинации «номер архива – номер полки», «номер архива – код ячейки», «номер полки – код ячейки» однозначно не идентифицируют фильм. Таким образом, мы определили третью характеристику первичного ключа – первичный ключ должен быть несократимым.

Конечно, после следующей модернизации архивов можно к «адресу» ячейки хранения пленок добавить номер комнаты, номер этажа и т. д. То есть первичный ключ будет разрастаться «вширь». При ручном заполнении всех значений, входящих в составной ключ, даже при соблюдении указанных характеристик первичного ключа, могут произойти ошибки. Составной ключ усложнит поиск в базе нужных данных (например, таблица заполнится на 50 000 строк), сортировку и др.

Для разрешения этой ситуации целесообразно создать еще одно поле-идентификатор – счетчик – (назовем его id), которое будет однозначно определять комбинацию полей выделенного ранее первичного ключа (таблица 3), а следовательно, – и записей таблицы.

Таблица 3 – Отношение Soviet_movies с полем id

id	archive	shelf	cod	film	director	year
1	1	782	10	Альпийская баллада	Степанов Б.	1965
2	1	782	15	Иван Васильевич меняет профессию	Гайдай Л.	1973
3	2	782	10	Служебный роман	Рязанов Э.	1977
4	3	310	4	Москва слезам не верит	Меньшов В.	1979
5	1	513	10	В бой идут одни «старики»	Быков Л.	1973
6	4	205	43	Обыкновенное чудо	Захаров М.	1978
7	2	205	43	Собачье сердце	Бортко В.	1988

Очевидно, что новое поле также попадает в множество потенциальных ключей. И здесь мы приходим еще к четвертой характеристике первичного ключа – первичный ключ должен быть минимальным.

Итак, в качестве потенциальных ключей у нас есть:

– составной ключ, который получен из описания предметной области: **archive – shelf – cod**;

– специально внесенный для упрощения работы с данными таблицы простой ключ: **id**.

Такие ключи соответственно имеют названия: *естественный* и *искусственный*.

Какой из них в нашем примере соответствует всем выделенным характеристикам первичного ключа? Простой искусственный ключ id.

Допустим, что возникла необходимость дополнить таблицу Soviet_movies следующими данными:

– Бриллиантовая рука – Гайдай Л. – 1968;

– Спортлото-82 – Гайдай Л. – 1982.

Также необходимо расширить информацию о режиссерах: добавить место и год их рождения. Тогда таблица примет следующий вид (таблица 4).

Таблица 4 – Отношение Soviet_movies после добавления данных

id	archive	shelf	cod	film	director	place_of_birth	year_of_birth	year
1	1	782	10	Альпийская баллада	Степанов Б.	Петропавловск	1927	1965
2	1	782	15	Иван Васильевич меняет профессию	Гайдай Л.	Свободный	1923	1973
3	2	782	10	Служебный роман	Рязанов Э.	Самара	1927	1977
4	3	310	4	Москва слезам не верит	Меньшов В.	Баку	1939	1979
5	1	513	10	В бой идут одни «старики»	Быков Л.	Знаменка	1928	1973
6	4	205	43	Обыкновенное чудо	Захаров М.	Москва	1933	1978
7	2	205	43	Собачье сердце	Бортко В.	Москва	1946	1988
8	3	320	4	Бриллиантовая рука	Гайдай Л.	Свободный	1923	1968
9	1	553	17	Спортлото-82	Гайдай Л.	Свободный	1923	1982

Информация о режиссерах в данном случае будет повторяться (дублироваться) неоднократно, что при больших количествах данных будет значительно перегружать таблицу, способствовать избыточности информации в ней.

А стоит ли вообще перегружать таблицу «адресом» ячейки хранения фильма в архиве или дублирующейся информацией о режиссерах? Здесь целесообразно сформировать соответствующие отдельные таблицы, которые можно будет дополнять. То есть таблицу Soviet_movies следует разбить на три: Soviet_movies_inf (таблица 5), Movies_adress (таблица 6), Directors (таблица 7).

Таблица 5 – Soviet_movies_inf

id	id_m_a	film	id_d	year
1	1	Альпийская баллада	1	1965
2	2	Иван Васильевич меняет профессию	2	1973
3	3	Служебный роман	3	1977
4	4	Москва слезам не верит	4	1979
5	5	В бой идут одни «старики»	5	1973
6	6	Обыкновенное чудо	6	1978
7	7	Собачье сердце	7	1988
8	8	Бриллиантовая рука	2	1968
9	9	Спортлото-82	2	1982

Таблица 6 – Movies_adress

id_m_a	archive	shelf	cod
1	1	782	10
2	1	782	15
3	2	782	10
4	3	310	4
5	1	513	10
6	4	205	43
7	2	205	43
8	3	320	4
9	1	553	17

Таблица 7 – Directors

id_d	name	place_of_birth	year_of_birth
1	Степанов Б.	Петропавловск	1927
2	Гайдай Л.	Свободный	1923
3	Рязанов Э.	Самара	1927
4	Меньшов В.	Баку	1939
5	Быков Л.	Знаменка	1928
6	Захаров М.	Москва	1933
7	Бортко В.	Москва	1946

Каким образом теперь возможно определить режиссера, снявшего фильм? Для этого нами в таблицу Directors (см. таблицу 7) был добавлен искусственный простой первичный ключ – поле **id_d**, и его значения внесены для соответствующих записей в таблицу Soviet_movies_inf (см. таблицу 5).

Определить «адрес» ячейки, в которой хранится оригинал пленки, можно по номеру **id_m_a**. Определить, кто из режиссеров снял определенный фильм, можно по номеру **id_d**.

Обратите внимание:

– значения **id_m_a** в таблицах Movies_adress (см. таблицу 6) и Soviet_movies_inf (см. таблицу 5) полностью совпадают без дублирования;

– значения **id_d** из таблицы Directors (см. таблицу 7) и Soviet_movies_inf (см. таблицу 5) совпадают и могут повторяться.

Это замечание пригодится при выполнении следующей лабораторной работы. Сейчас отметим, что поля **id_m_a** и **id_d** являются первичными ключами в одних таблицах и организуют связь с таблицей Soviet_movies_inf (см. таблицу 5). При этом в таблице Soviet_movies_inf содержатся значения первичных ключей из других таблиц, а поля **id_m_a** и **id_d** являются внешними ключами.

Внешний ключ (Foreign Key, FK) – атрибут (или группа атрибутов) отношения, содержащий в себе копии значений первичного ключа другого отношения [2, с. 47].

Порядок выполнения лабораторной работы

1. Изучите достоинства и недостатки естественных и искусственных ключей, понятие и примеры рекурсивного ключа [2, 6, 7].

2. Доработайте отношение A (см. порядок выполнения лабораторной работы №2, задание 2) таким образом, чтобы в нем было не менее трех потенциальных ключей.

3. Выделите из полученного отношения 3–5 отдельных отношений – его проекции B, C, D, …, дайте им имена. При необходимости дополните отношения атрибутами.

4. Доработайте эти отношения-проекции таким образом, чтобы в каждом из них был первичный ключ, при необходимости – внешние ключи. Для разных отношений подберите первичные ключи:

– естественный составной ключ;

– естественный простой ключ;

– искусственный ключ.

5. Изобразите схематически структуру базы данных, включающую полученные отношения с указанием их атрибутов (заголовки отношений) и свойств атрибутов (PK, FK, уникальность значений и др.).

Содержание отчета

1. Цель работы.

2. Краткие теоретические сведения.

3. Графические (табличные) результаты выполнения заданий с текстовым пояснением.

4. Схематическое изображение структуры базы данных.

5. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №4
УСТАНОВКА СВЯЗЕЙ, ОБЕСПЕЧЕНИЕ ССЫЛОЧНОЙ ЦЕЛОСТНОСТИ И КОНСИСТЕНТНОСТИ

Цель работы: ознакомиться с основами теории ссылочной целостности (связь, свойства, ссылочная целостность, консистентность) и приобрести практические навыки их реализации в базах данных.

Теоретические сведения

Связь (relationship) – это способ указания того факта, что отношения находятся в логическом взаимодействии друг с другом. Наличие связи налагает на объединенные этой связью отношения ряд ограничений, призванных гарантировать те или иные виды целостности базы данных [10].

Родительская сущность (родительское отношение, родительская таблица) – сущность, которая содержит первичный ключ, который в свою очередь может мигрировать в связанную сущность.

Дочерняя сущность – сущность, в которую мигрировал первичный ключ из родительской сущности. В дочерней сущности мигрировавший ключ становится внешним ключом.

Итак, для двух связанных таблиц родительской будет являться та, которая содержит первичный ключ, дочерней – та, которая содержит внешний ключ.

Вернемся к таблицам Soviet_movies_inf, Movies_adress, Directors. На рисунке 9 показана диаграмма базы данных (графическое изображение структуры БД), состоящая из этих таблиц.

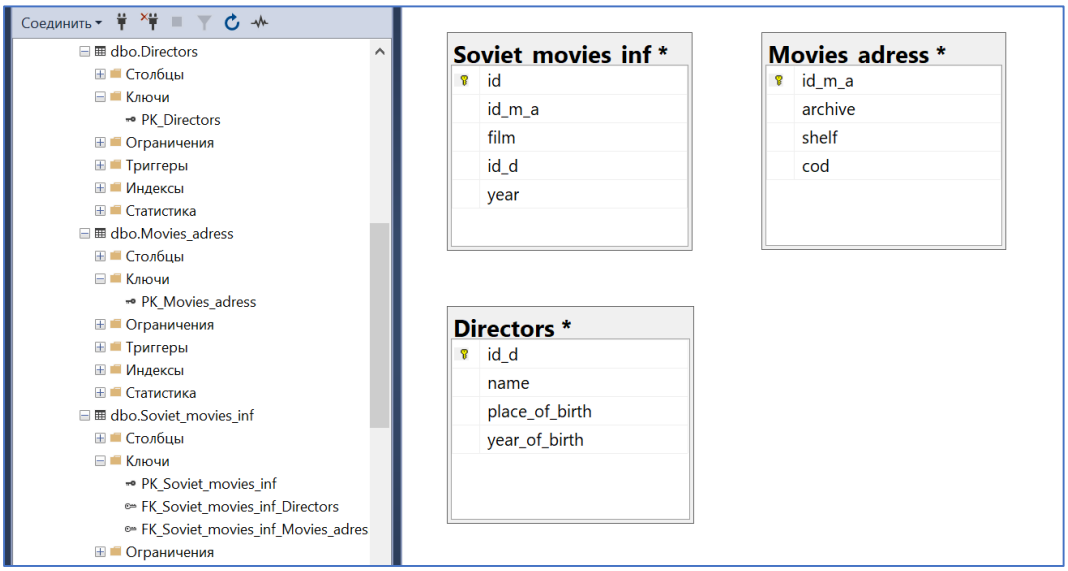


Рисунок 9 – Диаграмма базы данных Soviet_movies (только таблицы)

Таблица Movies_adress содержит первичный ключ, который мигрировал в таблицу Soviet_movies_inf.

Таблица Directors содержит первичный ключ, который мигрировал в таблицу Soviet_movies_inf.

Поэтому можно утверждать, что таблицы Movies_adress и Directors являются родительскими по отношению к таблице Soviet_movies_inf. В свою очередь таблица Soviet_movies_inf является дочерней по отношению к таблицам Movies_adress и Directors.

Значит, эти три отношения взаимодействуют друг с другом, т. е. связаны между собой.

Выделяют три основных типа связей [1, с. 11]:

1. «Один к одному (1:1)» – каждый экземпляр сущности A может быть связан не более чем с одним экземпляром сущности B.

2. «Один ко многим (1:N)» – каждый экземпляр сущности A может быть связан более чем с одним экземпляром сущности B, а каждый экземпляр сущности B может быть связан не более чем с одним экземпляром сущности A.

3. «Многие ко многим (M:N)» – каждый экземпляр сущности A может быть связан с несколькими экземплярами сущности B, а каждый экземпляр сущности B может быть связан с несколькими экземплярами сущности A.

!!! ЗАМЕЧАНИЕ. *В зависимости от рассматриваемых объектов (сущности, отношения, таблицы) применяется и соответствующая им терминология – экземпляр сущности, кортеж отношения, строка таблицы.*

В лабораторной работе №2 было отмечено следующее:

– значения **id_m_a** в таблицах Movies_adress и Soviet_movies_inf полностью совпадают без дублирования;

– значения **id_d** из таблицы Directors и Soviet_movies_inf совпадают и могут повторяться.

Следовательно:

– Movies_adress и Soviet_movies_inf связаны, как «один к одному»;

– Directors и Soviet_movies_inf связаны, как «один ко многим».

Установим эти типы связей на диаграмме базы данных (рисунок 10).

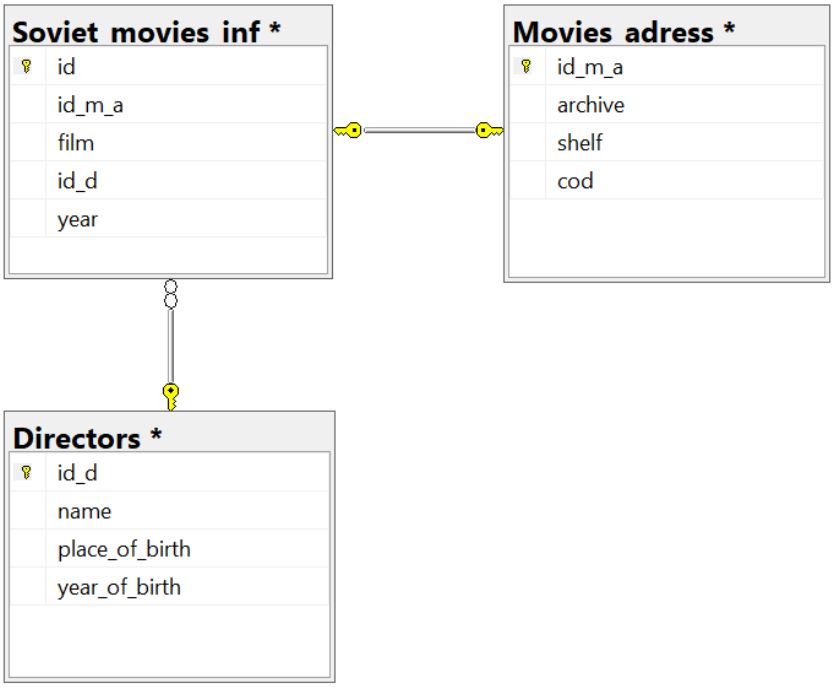


Рисунок 10 – Диаграмма базы данных Soviet_movies (с изображением таблиц и связей между ними)

Добавим в базу данных таблицу Studio (киностудия) и попробуем определить, кто из режиссеров снимал свои фильмы на какой киностудии.

Изучив информацию, выясним, что каждый из указанных режиссеров работал не на одной (в том числе и в качестве актера), а на разных киностудиях. И наоборот, на каждой из киностудий снимались фильмы множеством режиссеров (для большей наглядности в таблицах оставим только соответствующие поля) (рисунок 11).

Directors

id_d	name
1	Степанов Б.
2	*Гайдай Л.*
3	*Рязанов Э.*
4	*Меньшов В.*
5	**Быков Л.**
6	Захаров М.
7	Бортко В.

Studio

id_s	title
1	Беларусьфильм
2	*Мосфильм*
3	Одесская киностудия
4	**Киностудия им. М. Горького**
5	**Ленфильм**
6	**Киностудия имени А. Довженко**

Рисунок 11 – Пример связи «многие ко многим»

Таким образом, таблицы Studio и Directors связаны как «многие ко многим». Для реализации такой связи в реляционной модели добавляется промежуточная таблица, состоящая из первичных ключей двух связанных таблиц. То есть

связь «многие ко многим» преобразуется в комбинацию связей «один ко многим». К этому моменту мы еще вернемся, когда будем проектировать базу данных в следующих работах.

Кроме основных, есть еще дополнительные типы связей, которые являются специфическими для конкретной задачи и/или предметной области, например, «1 к 7», «1 к 0…8», «0…2 к 4» и т. д.

Ссылочная целостность (referential integrity) – свойство реляционной базы данных, состоящее в неукоснительном соблюдении правила: если внешний ключ дочернего отношения содержит некоторое значение, это значение обязательно должно присутствовать в первичном ключе родительского отношения [2, с. 72].

Вернемся к рисунку 10. Рассмотрим связь между таблицами Directors и Soviet_movies_inf. Как было показано выше, – это связь «один ко многим». Родительской таблицей является Directors, т. к. в ней находится первичный ключ (значения уникальны), который мигрирует в дочернюю таблицу Soviet_movies_inf. В последней таблице мигрировавший ключ становится внешним ключом (значения должны входить во множество значений первичного ключа, могут повторяться *несколько* раз).

Подумаем, как повлияют на записи в связанных между собой таблицах следующие действия, если произойдет:

– удаление записи из родительской таблицы Directors;
– удаление значения FK из дочерней таблицы Soviet_movies_inf;
– вставка новой записи в родительскую таблицу Directors;
– вставка нового значения FK (содержащегося и отсутствующего в таблице Directors) в дочернюю таблицу Soviet_movies_inf;
– изменение значения PK в родительской таблице Directors;
– изменение значения FK (содержащегося и отсутствующего в таблице Directors) в дочерней таблице Soviet_movies_inf?

Для ответа на эти вопросы существуют правила соблюдения условий ссылочной целостности (таблица 8).

Таблица 8 – Случаи нарушения (–) ссылочной целостности и ее соблюдения (+)

Операция для значений ключа	**В родительской таблице (для PK)**	**В дочерней таблице (для FK)**
Вставка (нового значения)	+	–
Изменение	–	–
Удаление	–	+

В реляционных СУБД существуют механизмы поддержания действий по обеспечению ссылочной целостности. Они работают при удалении или обновлении значений первичного ключа в родительской таблице. Так, в СУБД SQL Server:

– ***Cascade***. Удаление (изменение) значения PK в родительской таблице приведет к удалению (изменению) значений FK в дочерних таблицах;

– ***No action***. Не позволяет удалять (изменять) в родительской таблице значение PK, если существует один или несколько ссылающихся на него значений FK в дочерних таблицах;

– ***Set null***. Удаление (изменение) значения PK в родительской таблице приведет к установке в соответствующее значение FK дочерней таблицы значения Null;

– ***Set default***. При удалении (изменении) значения PK в родительской таблице устанавливается значение по умолчанию во внешний ключ дочерней таблицы.

!!! ЗАМЕЧАНИЕ. *Каскадные операции запускаются только при затрагивании значений первичного ключа в родительской таблице.*

!!! ЗАМЕЧАНИЕ. *Под словами «удаление значения первичного ключа» понимается удаление соответствующей записи (строки) в таблице. Под словами «изменение значения первичного ключа» понимается изменение именно значения ключа.*

Рассмотрим пример описания механизмов поддержания ссылочной целостности в базе данных Soviet_movies (таблица 9).

Таблица 9 – Описание ссылочной целостности в базе данных Soviet_movies

Родительская таблица (с PK)	**Дочерняя таблица, внешний ключ (FK)**	**Механиз поддержания ссылочной целостности и его описание**	
		при операции изменения	**при операции удаления**
Directors	Soviet_movies_inf, id_d (FK)	Cascade При изменении значения первичного ключа в родительской таблице Directors связанные значения внешнего ключа в дочерней таблице Soviet_movies_inf будут также соответственно изменены. Например, изменение номера режиссера (PK) в таблице Directors приведет к тому, что во все данные (FK) о снятых им фильмах также будет внесена соответствующая правка	No action При попытке удаления значений первичного ключа в таблице Directors произойдет откат этой операции. Например, запрет на удаление режиссера, если в базе есть фильмы, снятые им

Консистентность базы данных (database consistency) – свойство реляционной базы данных, состоящее в неукоснительном соблюдении в любой момент времени всех ограничений, заданных неявно реляционной моделью или явно конкретной схемой базы данных [2, с. 72].

Порядок выполнения лабораторной работы

1. Доработайте последний вариант базы данных (см. лабораторную работу №3) таким образом, чтобы в ней были реализованы все основные типы связей: «один ко многим» (несколько раз), «один к одному» и «многие ко многим» (не менее одного раза).

2. Модернизируйте полученную реляционную базу данных под правила ссылочной целостности.

3. Не генерируя код базы данных, продумайте все механизмы поддержания ссылочной целостности: Cascade, No action,* Set default, Set null. Опишите их действия в соответствии с таблицей 9.

4. С помощью конструктора любого средства проектирования (например, СУБД) постройте диаграмму базы данных, реализуйте установленные механизмы поддержания ссылочной целостности.

5. Заполните таблицы с помощью конструктора не менее чем на пять строк. Проверьте работу механизмов ссылочной целостности.

6. Оформите отчет.

Содержание отчета

1. Цель работы.

2. Краткие теоретические сведения.

3. Диаграмма базы данных с реализацией всех типов связей и механизмов поддержания ссылочной целостности.

4. Описание действия механизмов ссылочной целостности в базе данных.

5. Скриншоты изменений в дочерней таблице при реализации механизмов ссылочной целости для операций в родительской таблице.

6. Выводы по работе.

---

* Механизмы поддержания действий по обеспечению ссылочной целостности No action и Restrict являются схожими. В некоторых СУБД они отождествляются, в некоторых – есть отличия. В СУБД SQL Server они по сути совпадают и используется только No action.

ЛАБОРАТОРНАЯ РАБОТА №5
НОРМАЛИЗАЦИЯ БАЗЫ ДАННЫХ

Цель работы: изучить нормальные формы и привести существующую базу данных к третьей нормальной форме.

Теоретические сведения

Аномалии операций с данными

Аномалии операций с данными (data operation anomalies) – некорректное выполнение операций с данными или возникновение побочных эффектов операций с данными, ставшее результатом нарушения требования адекватности базы данных предметной области [2, с. 161].

Выделяют аномалии вставки, изменения и удаления. Рассмотрим на примерах каждую из них.

Аномалии вставки возникают в том случае, когда при добавлении информации в значение поля нам необходимо достраивать всю запись данными либо устанавливать вместо них значения NULL.

Добавляя в таблицу нового режиссера (name), нам необходимо заполнить поля с его местом и годом рождения либо написать там NULL. Очевидно, что запись, содержащая место и/или год рождения без указания режиссера, вообще теряет смысл, т. е. значение NULL в поле name неприемлемо (таблица 10).

Таблица 10 – Пример аномалии вставки в отношении Directors

id_d	name	place_of_birth	year_of_birth
1	Степанов Б.	Петропавловск	1927
2	Гайдай Л.	Свободный	1923
…	…	…	…
8	Тарковский А.	?	?
9	???????	?	1920

Аномалии удаления возникают в том случае, когда при ликвидации одних данных исчезают другие, которые изначально не предполагалось удалять.

Предположим, что все архивы начинают оцифровывать, информацию переносить на сервер, а записи на пленках удаляют из ячеек архивов. Например, оцифровали фильмы (и перенесли на сервер), созданные режиссером Э. Рязановым. Следовательно, из базы данных исчезнет информация (например, год выпуска) о всех картинах, снятых этим режиссером, а также информация о том, что в данном архиве есть полка под определенным номером, на которой расположена ячейка с определенным кодом (таблица 11).

Таблица 11 – Пример аномалии удаления в отношении Soviet_movies

archive	shelf	cod	film	director	year
1	782	10	Альпийская баллада	Степанов Б.	1965
1	782	15	Иван Васильевич меняет профессию	Гайдай Л.	1973
~~2~~	~~782~~	~~10~~	~~Служебный роман~~	~~Рязанов Э.~~	~~1977~~
3	310	4	Москва слезам не верит	Меньшов В.	1979
1	513	10	В бой идут одни «старики»	Быков Л.	1973
4	205	43	Обыкновенное чудо	Захаров М.	1978
2	205	43	Собачье сердце	Бортко В.	1988
~~2~~	~~782~~	~~72~~	~~Гараж~~	~~Рязанов Э.~~	~~1979~~

Аномалии изменения возникают в том случае, когда при обновлении значения некоторого атрибута в одной записи появляется необходимость его обновления и в других записях.

Например, архивы теперь не нумеруются, а обозначаются буквами латинского алфавита. То есть архив 1 становится архивом A, архив 2 – архивом B и т. д. Теперь нам необходимо обновить названия архивов во всех записях, не пропустив и не перепутав их новые заглавия (таблица 12):

Таблица 12 – Пример аномалии изменения в отношении Soviet_movies

archive	shelf	cod	film	director	year
A	782	10	Альпийская баллада	Степанов Б.	1965
1	782	15	Иван Васильевич меняет профессию	Гайдай Л.	1973
B	782	10	Служебный роман	Рязанов Э.	1977
C	310	4	Москва слезам не верит	Меньшов В.	1979
1	513	10	В бой идут одни «старики»	Быков Л.	1973
…	…	…	…	…	…

Получение отношений, не подверженных аномалиям вставки, удаления и изменения данных, реализуется, в частности, в процессе их нормализации.

Теория зависимостей

Заметим, что в теории баз данных можно встретить такие понятия, как «переменная отношения» и «отношение». Это различные понятия, которые часто отождествляют. Разница в том, что первое – это именно таблица в СУБД, второе – данные, хранящиеся в ней.

Функциональная зависимость. В значении переменной отношения r атрибут Y функционально зависит от атрибута X в том и только в том случае, если каждому значению X соответствует в точности одно значение Y. В этом случае говорят также, что атрибут X функционально определяет атрибут Y. Обозначается такая зависимость следующим образом: r.X→ r.Y [5, с. 111].

Функциональную зависимость Y от X будем обозначать как X→Y. X и Y могут представлять собой как единичные атрибуты, так и составные, полученные из нескольких атрибутов одного отношения.

Рассмотрим некоторые виды функциональных зависимостей, которые будут использоваться далее. Следующие два определения основаны на терминологии из источника [10].

Полная функциональная зависимость – функциональная зависимость X → Y, при которой Y не зависит функционально от любого подмножества X.

Например, в отношении Soviet_movies название фильма можно определить по полному адресу ячейки, в которой лежит пленка; если исключить, например, из адреса ячейки номер архива, то сказать с уверенностью, какой фильм содержится в ячейке хранения, не получится:

$$\{archive, shelf, cod\} \rightarrow film$$

Частичная функциональная зависимость – функциональная зависимость X → Y, при которой Y зависит функционально от некоторого подмножества X.

Например, в отношении Soviet_movies для определения года выпуска фильма можно знать название фильма и режиссера. Если определить, что фильмы должны сниматься под уникальным названием, то год выпуска не зависит от имени режиссера, достаточно знать только название картины:

$$\{film, \sout{director}\} \rightarrow year$$

Транзитивная функциональная зависимость – зависимость X→Y, при которой существует такой атрибут Z, что имеются функциональные зависимости X→Z и Z→Y и отсутствует функциональная зависимость Y → X [5, с. 114].

Например, в отношении Soviet_movies название фильма в ячейке можно узнать по адресу ячейки, а год выпуска фильма – по его названию:

$$\{archive, shelf, cod\} \rightarrow film \rightarrow year$$

Выделяют также многозначную зависимость, избыточные полную и частичную зависимости и др.

Нормальные формы

Вспомним одну из специальных реляционных операций – проекция. Рассмотрим ее следующим образом: допустим, что отношение R состоит из n атрибутов и k кортежей; если из данного отношения составить новое, взяв только m атрибутов (m < n) и все кортежи без дубликатов, то полученное отношение R_1 будет являться ***проекцией*** исходного отношения.

Нормализация – процесс декомпозиции переменной отношения R на набор проекций R_1, R_2, ..., R_n, таких, что:

а) объединение проекций R_1, R_2, ..., R_n позволяет гарантированно получить исходную переменную отношения R;

б) каждая из проекций R_1, R_2, ..., R_n является необходимой для выполнения условия «а»;

в) как минимум одна из проекций R_1, R_2, ..., R_n находится в более высокой нормальной форме, чем исходная переменная отношения R [2, с. 211].

Нормализация позволяет устранить дублирование данных в таблице, исключить аномалии вставки, удаления, изменения данных, борется с избыточностью данных.

Нормализация может рассматриваться как процесс поэтапного выполнения определенных условий до тех пор, пока дальнейшая декомпозиция получаемых на каждом этапе отношений станет невозможной.

Для реализации процесса нормализации отношения приводят к нормальным формам. В общей теории баз данных выделяют шесть нормальных форм, нормальную форму Бойса – Кодда и доменно-ключевую нормальную форму. Однако часто достаточно нормализовать отношения базы данных до третьей нормальной формы.

На основании источников [2, 5, 7] сформулируем определения нормальных форм.

Первая нормальная форма (1НФ) предполагает приведение переменной отношения к виду, в котором все атрибуты отношения будут атомарны. Атомарность может определяться особенностями предметной области, через исключение многозначности данных, составных данных. Здесь следует исключить дублирование кортежей отношения (повторения строк (записей) в таблице).

Вторая нормальная форма (2НФ). Переменная отношение находится во 2НФ, если оно находится в 1НФ и не содержит частичных функциональных зависимостей неключевых* атрибутов от первичного ключа: неключевые атрибуты должны функционально полно зависеть от первичного ключа.

Очевидно, что 2НФ в этом случае будет достигнута, если первичный ключ – простой. Если он составной, то целесообразно добавить в отношение искусственный первичный ключ – счетчик (идентификатор (id)), а составной первичный ключ (который становится альтернативным ключом) вынести в отдельное отношение. Если же добавление id не предполагается по каким-то причинам, то следует добиться соответствующей полной функциональной зависимости.

Третья нормальная форма (3НФ). Переменная отношения находится в 3НФ, если оно находится во 2НФ и не содержит транзитивных функциональных зависимостей неключевых атрибутов от первичного ключа.

Например, нарушением 3НФ будет наличие вычисляемого столбца в таблице.

Нормальная форма Бойса – Кодда (НФБК) (частная форма 3НФ). Рассмотрим ситуацию, когда отношение содержит более одного потенциального ключа, два и более потенциальных ключа являются составными и имеют хотя бы один общий атрибут. Переменная отношения находится в нормальной форме Бойса – Кодда (НФБК) тогда и только тогда, когда она находится в 3НФ и не содержит пересекающихся потенциальных ключей [2, с. 259].

---

* Неключевой атрибут – атрибут отношения, который не входит в состав ни одного потенциального (возможного) ключа этого отношения [2, с. 23; 5, с. 129]. К. Дейт определяет 2НФ и 3НФ при условии наличия в отношении только одного потенциального ключа, который и является первичным ключом отношения, а также неключевой атрибут, как атрибут, который не входит в состав первичного ключа рассматриваемой переменной отношения [7, с. 467–474].

Порядок выполнения лабораторной работы

1. Определите вариант для выполнения заданий (приложение Б) и выполните нормализацию до 3НФ соответствующего отношения (возможна нормализация и до высших нормальных форм – по желанию).

2. Изобразите графически структуру (диаграмму) полученной базы данных в любом редакторе (используйте правила наименования таблиц и их полей, сформулированные в лабораторной работе №2).

3. Оформите отчет.

Содержание отчета

1. Цель работы.
2. Краткие теоретические сведения.
3. Выполненные задания лабораторной работы.
4. Поэтапное описание нормализации исходной таблицы до 3НФ.
5. Графическое изображение структуры (диаграмма) итоговой базы данных.
6. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №6
ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ НА ИНФОЛОГИЧЕСКОМ УРОВНЕ

Цель работы: ознакомиться с задачами этапа концептуального проектирования баз данных, разработать инфологическую модель с учетом семантических ограничений предметной области.

Теоретические сведения

Инфологический (концептуальный) уровень является первым в процессе проектирования баз данных. Здесь происходит анализ предметной области, выделяются сущности, их свойства (атрибуты), а также устанавливаются связи между сущностями. То есть проводится описание (и даже структурирование) объектов и их характеристик, относящихся к предметной области, которые потом будут помещены в базу данных.

Основными составными элементами инфологической модели являются сущности, связи между ними и атрибуты сущностей.

Понятия «сущность», «отношение», «таблица» можно считать эквивалентными в рамках описываемых терминов и рассматриваемых объектов баз данных. Не переходя к конкретным определениям, можно упрощенно сказать, что сущность в инфологической модели – это своего рода прообраз таблицы в реляционной базе данных. Аналогично можно сказать, что атрибут – это прообраз столбца таблицы.

В инфологической модели также целесообразно выделять те атрибуты (или их совокупность), по значениям которых можно однозначно найти требуемый экземпляр сущности, т. е. первичные ключи.

Инфологическая модель может быть описана в текстовой форме (дается список сущностей, перечисляются их атрибуты, приводятся некоторые комментарии, формируются ограничения и др.) или в графической форме (ER-диаграмма, UML-диаграмма и др.).

Каждая из этих форм имеет преимущества и недостатки, но все же предпочтительней является вторая – графическая форма.

Например, спроектируем инфологическую модель взаимодействия врачей и пациентов в поликлинике. Пусть определены некоторые требования, подлежащие проектированию:

– за каждым поступившим в отделение пациентом закрепляется один лечащий врач;
– каждый врач относится к одной специальности;
– врач лечит множество пациентов;
– пациент определяется в палату;
– известны данные о врачах (фамилия, имя, отчество);
– известны данные о пациенте (фамилия, имя, отчество, номер карты, пол, номер палаты).

Инфологическая модель в текстовой форме.

Из описания предметной области можно выделить минимум две сущности: врач и пациент. Определим свойства (атрибуты) каждой из них:

1. Врач:
   а) фамилия;
   б) имя;
   в) отчество;
   г) специальность.
2. Пациент:
   а) фамилия;
   б) имя;
   в) отчество;
   г) номер карты;
   д) пол;
   е) номер палаты.

Внимательно изучив полученный результат, можно обнаружить, что специальности у разных врачей будут повторяться, пол пациента может быть либо мужским, либо женским. Требуется также как-то идентифицировать врачей и пациентов. Таким образом, получим следующий набор сущностей и атрибутов:

1. Врач:
   а) идентификатор;
   б) фамилия;
   в) имя;
   г) отчество;
   д) специальность (FK).
2. Специальность:
   а) идентификатор;
   б) название.
3. Пациент:
   а) идентификатор;
   б) номер карты (уникальный);
   в) фамилия;
   г) имя;
   д) отчество;
   е) номер палаты;
   ж) пол (FK);
   з) лечащий врач (FK).
4. Пол:
   а) идентификатор;
   б) название.

Здесь можно установить, например, правила составления номера карты пациента (дата поступления пациента – первые буквы фамилии, имени, отчества пациента – идентификатор врача – номер палаты – уникальный номер), номера

палаты (отделение – корпус – этаж – номер палаты) и др. То есть описание может постоянно дополняться.

Инфологическая модель на основе ER-диаграммы.

Диаграммы «сущность – связь» (ER-диаграммы) намного информативнее и нагляднее текстового описания. Они в графическом виде отображают связи между двумя сущностями. На основе ER-диаграмм строятся ER-модели предметной области. На них четко выделяются сущности, атрибуты, ключевые атрибуты, а также связи между сущностями (с указанием типов связей). В зависимости от нотации (набора объектов для изображения составляющих модели (типов фигур, стрелок, символов, правил и др.)) сущности, атрибуты, связи могут изображаться по-разному.

Так, например, в нотации П. Чена сущность изображается прямоугольником, атрибут – овалом, связь – ромбом. Также могут быть выделены зависимые, независимые сущности, первичный ключ (подчеркивается одной линией) и др. Изобразим инфологическую модель описанной задачи на данном этапе в нотации П. Чена (рисунок 12).

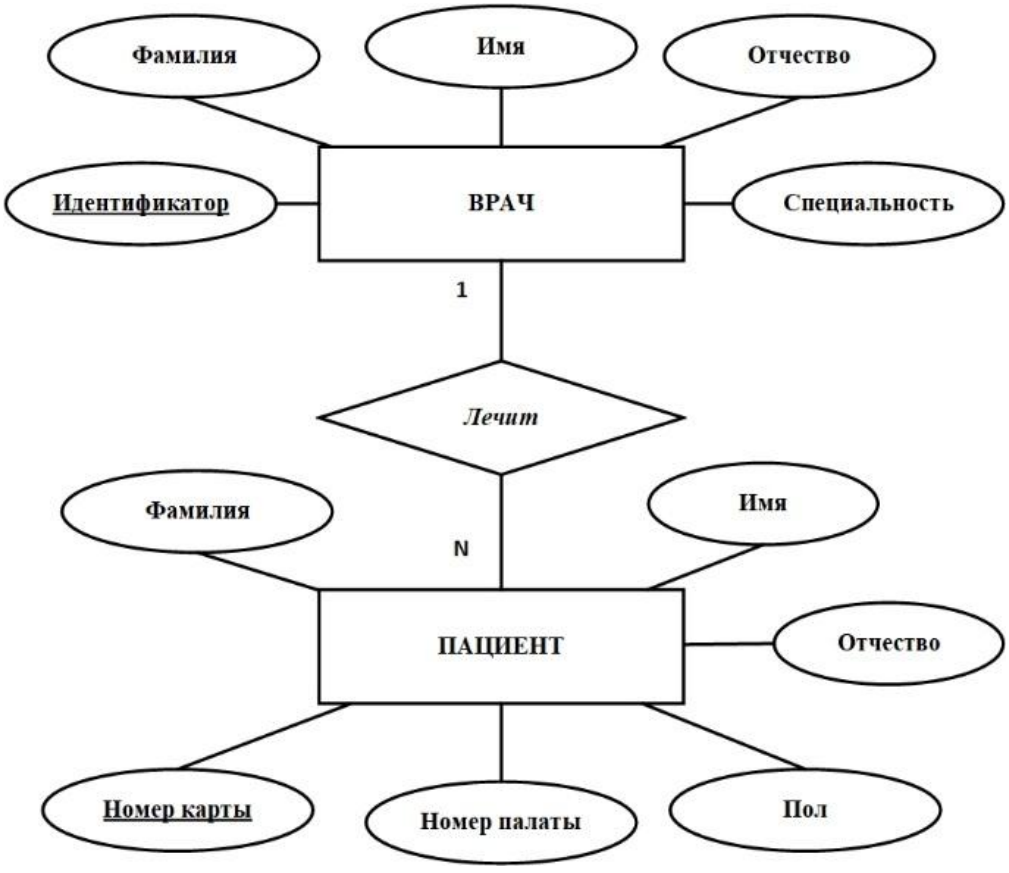


Рисунок 12 – Фрагмент ER-модели в нотации П. Чена

Для построения моделей предметной области могут быть использованы нотация Мартина, IDEF1X и др. Для моделирования данных можно воспользоваться, например, следующими программными продуктами: All Fusion ERWin Data Modeler, Sparx Enterprise Architect, IBM Rational Data Architect и др.

Порядок выполнения лабораторной работы

1. По предложенному варианту заданий (приложение В) проанализируйте соответствующую предметную область. Сформулируйте ее письменное описание, выделите не менее пяти требований к проектированию.

2. Сформируйте набор сущностей (не менее шести).

3. Для каждой сущности (по возможности) определите более трех атрибутов.

4. Сформируйте все типы связей между сущностями.

5. Сформируйте различные виды ключей (опишите, какие атрибуты (или совокупности атрибутов) могли бы стать тем или иным видом ключа): первичный (простой, составной; естественный, искусственный), внешний. Укажите альтернативные ключи.

6. Постройте инфологическую модель (предпочтительно ER-модель в выбранной нотации) предметной области с помощью любого удобного инструмента.

Содержание отчета

1. Цель работы.

2. Краткие теоретические сведения.

3. Перечисление сущностей и их атрибутов.

4. Описание ключей (перечислите атрибуты (совокупности атрибутов), относящиеся к каждому виду ключа).

5. Графическое изображение спроектированной инфологической модели.

6. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №7
ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ НА ДАТАЛОГИЧЕСКОМ УРОВНЕ

Цель работы: преобразовать построенную инфологическую модель базы данных в реляционную.

Теоретические сведения

На данном этапе проектирования базы данных следует преобразовать инфологическую модель в реляционную. Определим некоторые минимальные правила для такого перехода (формирование таблиц из ER-диаграмм) на основе [1, 6]:

1. Сущность ER-диаграммы становится таблицей.

2. Атрибуты сущности ER-диаграммы становятся столбцами таблицы. Из уникальных идентификаторов сущности выбирается один, который становится первичным ключом в таблице. В качестве первичных ключей целесообразно выбрать (или добавить) искусственные, простые, с автоинкрементируемыми («счетчик») значениями (уникальные, натуральные числа).

3. Преобразуются связи (изучите самостоятельно по [1, с. 14–15; 6, пункт 4.2.4]):

а) *типа 1:1 одним из следующих способов:*

– отношения со связью 1:1 объединяются в одну таблицу;

– в одну из таблиц добавляется новое поле – внешний ключ, ссылающийся на соответствующие значения первичного ключа другой таблицы (второй из этих двух);

– строится три таблицы: по одной для каждой сущности и одна связывающая их;

б) *типа 1:N одним из следующих способов:*

– одна таблица становится родительской (в ней находится первичный ключ), вторая таблица становится дочерней (в ней находится соответствующий внешний ключ);

– определенным образом создается промежуточная таблица;

в) *типа M:N следующим образом:* создается промежуточная таблица, состоящая из внешних ключей, ссылающихся на первичные ключи связываемых таблиц; такую таблицу называют «Таблица1_Таблица2».

Далее проводится нормализация отношений (таблиц).

Также на этом этапе:

– определяется тип и конкретная СУБД для дальнейшей работы;

– определяются типы данных для значений столбцов таблиц и возможные ограничения;

– формируются правила работы в конкретной СУБД, в частности:

а) технические требования (версия СУБД, тип сервера и др.);

б) правила именования объектов (таблиц, хранимых процедур, триггеров и др.);

в) правила оформления SQL-кода (написание ключевых слов, комментариев и др.).

Для нашей работы выберем реляционную СУБД SQL Server 2019 или выше, утилиту SQL Server Management Studio 18 или выше (можно использовать соответствующие версии Express) [3].

Для выявленных в инфологической модели связей заполним таблицу (пример приведен в таблице 13).

Таблица 13 – Описание выявленных связей концептуальной модели

№	Название связи	Сущности, участвующие в связи	Назначение
1	1:N	Специальность – Врач	Одной специальности могут соответствовать несколько врачей
2	1:N	Врач – Пациент	Один врач закреплен за несколькими пациентами
3	N:1	Пациент – Пол	Множество пациентов могут быть одного пола

Отметим, что связи 1:N и N:1 – это один и тот же тип связи. С целью показать направление этой связи мы записали ее двумя способами.

Опишем каждую таблицу полученной на данном этапе даталогической модели БД (таблица 14). Имена таблиц и полей сформируем согласно правилам, описанным выше.

Таблица 14 – Описание таблиц и некоторые их характеристики

Имя поля	Тип данных	Ограничения
Таблица Doctor		
id_doctor	Числовой	PK (первичный ключ)
id_ speciality	Числовой	FK (внешний ключ)
surname	Символьный	–
name	Символьный	–
patronymic	Символьный	–
Таблица Speciality		
id_ speciality	Числовой	PK (первичный ключ)
title	Символьный	–
Таблица Patient		
id_ patient	Числовой	PK (первичный ключ)
card_number	Числовой	Уникальное значение (unique)
surname	Символьный	–
name	Символьный	–
patronymic	Символьный	–
room_number	Числовой	Не может быть больше 500
id_pol	Числовой	FK (внешний ключ)
id_doctor	Числовой	FK (внешний ключ)
Таблица Pol		
id_pol	Числовой	PK (первичный ключ)
title	Символьный	«женский» или «мужской»

Можно заметить, что в нескольких таблицах есть поля с одинаковыми названиями, например, в таблицах Doctor и Patient есть поля surname, name, patronymic. Как же в дальнейшем их различать, в частности, при создании запросов? Здесь возможно несколько вариантов:

– определить на данном этапе правило формирования имени поля через указание таблицы, например, doctor_surname или surname_doctor;

– в многотабличных запросах придется всегда формировать полное имя поля, указывая вначале имя таблицы, например, Doctor.surname.

Порядок выполнения лабораторной работы

1. Изучите теоретическую часть.
2. Опишите выявленные связи концептуальной модели.
3. Примените правила преобразования инфологической модели (для ER-моделей) в реляционную.
4. Проведите нормализацию отношений (таблиц) до 3НФ (НФБК), при желании – до высших нормальных форм.
5. Выполните остальные рекомендации к данному этапу проектирования баз данных.
6. Опишите полученные таблицы базы данных, их характеристики.
7. С помощью любого средства проектирования постройте модель базы данных на даталогическом уровне (примеры создания модели базы данных с помощью ERwin приведены в [1, 10], с помощью Sparx Enterprise Architect в [2, 10]; также возможно применить средства выбранной для работы СУБД).

Содержание отчета

1. Цель работы.
2. Краткие теоретические сведения.
3. Описание выявленных связей концептуальной модели.
4. Описание полученной даталогической модели.
5. Графическое изображение спроектированной модели базы данных на даталогическом уровне.
6. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №8
ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ НА ФИЗИЧЕСКОМ УРОВНЕ

Цель работы: разработать физическую модель базы данных на основе даталогической.

Теоретические сведения

Физический уровень моделирования продолжает детализацию и позволяет максимально учесть технические особенности работы конкретной СУБД и ее возможности по организации и управлению объектами разрабатываемой базы данных и данными в ней.

Современные средства проектирования баз данных позволяют автоматически сгенерировать SQL-код для создания базы данных. Однако в нашей работе база данных является не такой объемной, и мы считаем целесообразным создать ее «вручную», не привязываясь к интерфейсу СУБД, – написать самостоятельно код для ее создания, изменения и др.

Итак, на данном этапе проектирования базы данных мы уже определили тип и конкретную СУБД для дальнейшей работы. Поэтому целесообразно, в частности, определить права доступа, кодировки, методы доступа, индексы, настройки СУБД [2, с. 314].

Так, в частности, индексы нужны для ускорения поиска информации в базе данных. Часто индексы в базе данных характеризуют как аналогию пунктов содержания в книге, по которым можно быстро перейти к нужной главе, параграфу, пункту. Индексы создаются для полей, соответствующих определенным правилам. Так, например, индексы создаются для полей, указанных в условиях запроса. Автоматически создается индекс (кластеризованный) для первичного ключа таблицы (если иное поле не было определено для него ранее).

Также следует определиться с точными типами данных для значений атрибутов. Напомним, что мы работаем с помощью СУБД SQL Server 2019. Большинство типов данных этой СУБД стандартно.

Следует учитывать, что в SQL Server нет логического типа данных. Вместо него можно использовать тип BIT. Значение поля BIT равно «1», или «0», или NULL (если определена такая возможность).

Еще одной особенностью данной СУБД является работа с типами данных: nchar, nvarchar, ntext (см. подробнее в [3]).

Точные типы данных, которые мы добавляем на физическом уровне проектирования, указаны в таблице 15.

Таблица 15 – Описание таблиц с указанием типов данных для конкретной СУБД

Имя поля	Тип данных	Not Null	Ограничения
Таблица Doctor			
id_doctor	int	+	PK (первичный ключ), автоинкрементируемый
id_ speciality	int	–	FK (внешний ключ)*
surname	nvarchar(50)	+	–
name	nvarchar(50)	+	–
patronymic	nvarchar(50)	–	–
Таблица Speciality			
id_speciality	int	+	PK (первичный ключ), автоинкрементируемый
title	nvarchar(50)	+	–
Таблица Patient			
id_patient	int	+	PK (первичный ключ), автоинкрементируемый
card_number	int	+	unique
surname	nvarchar(50)	+	–
name	nvarchar(50)	+	–
patronymic	nvarchar(50)	–	–
room_number	int	+	Не может быть больше 500
id_pol	int	+	FK (внешний ключ)
id_doctor	int	+	FK (внешний ключ)
Таблица Pol			
id_pol	int	+	PK (первичный ключ), автоинкрементируемый
title	nvarchar(10)	+	«женский» или «мужской»

Создадим диаграмму базы данных с помощью SQL Server Management Studio. Все свойства полей можно увидеть в окне «Свойства», выделив соответствующий столбец таблицы на диаграмме (рисунок 13).

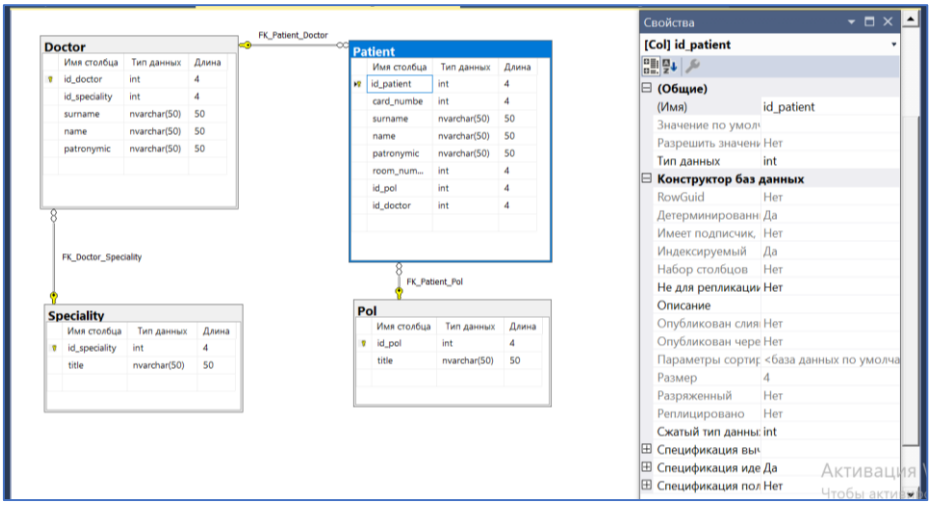


Рисунок 13 – Диаграмма базы данных Poliklinika

*Пусть для некоторых сотрудников из таблицы Doctor не определена специальность, например, для студентов, не окончивших обучение, но пришедших на практику в поликлинику. С точки зрения классической теории реляционной модели данных определение NULL для значений внешнего ключа не одобряется, но и не запрещено. Подробнее читайте в [7, с. 749–751]. В указанном случае мы разрешаем значение NULL с целью рассмотрения в дальнейшем некоторых примеров.

Чтобы воспользоваться модулем создания диаграммы в SQL, Server необходимо первоначально создать сами таблицы с помощью кода SQL либо встроенного конструктора. Выберем первый вариант. Приведем пример кода для создания таблицы Patient и определения некоторых свойств ее полей:

```
Create table Patient
(
        id_patient int identity(1,1) Primary Key,
        card_number int unique not null,
        surname nvarchar(50) not null,
        name nvarchar(50) not null,
        patronymic nvarchar(50) not null,
        room_number int not null,
        id_pol int not null,
        id_doctor int not null
)

Alter table Patient
Add constraint FK_Patient_Doctor
Foreign Key(id_doctor) references Doctor (id_doctor)

Alter table Patient
Add constraint FK_Patient_Pol
Foreign Key(id_pol) references Pol (id_pol)
```

Некоторые комментарии к данному коду:

– свойство IDENTITY(1,1) означает изменение значения поля на единицу, начиная с номера 1 (другими словами – поле «счетчик»);

– свойство UNIQUE означает уникальность значений поля;

– свойство CONSTRAINT означает создание связи между таблицами.

!!! ЗАМЕЧАНИЕ. *Отметим, что свойство поля PRIMARY KEY включает в себя характеристики UNIQUE и NOT NULL, но наоборот – не всегда.*

Приведенный для примера скрипт создания таблицы базы данных возможно оформить также только с помощью конструкции CREATE. Такое описание будет приведено в следующей лабораторной работе.

На физическом уровне проектирования баз данных следует установить индексы. Они связаны с таблицей или представлением и необходимы для оптимизации запросов. Индекс содержит ключи, построенные из одного или нескольких столбцов в таблице или представлении, предоставляя быстрый доступ к строкам данных.

Кластеризованные индексы сортируют и хранят строки данных в таблицах или представлениях на основе их ключевых значений – столбцов, включенных в определение индекса. Для каждой таблицы может существовать только один кла-

стеризованный индекс. Если он есть, то строки данных в таблице хранятся в порядке сортировки. Если такой индекс не определен заранее, то им становится поле первичного ключа.

Некластеризованный индекс определяется вручную. Обычно его определяют для таких полей, как внешние ключи, часто используемых в предложении select, подзапросах, order by и других случаях.

Для создания индекса используют следующий синтаксис.

```
--Создание кластеризованного индекса
Create clustered index Name_Index
On Name_Table (Name_Column);
Go

--Создание некластеризованного индекса
Create nonclustered index Name_Index
On Name_Table (Name_Column);
Go
```

Больше информации об индексах, их видах и хранении можно найти в [2, 3, 5].

Порядок выполнения лабораторной работы

1. Изучите теоретическую часть.
2. Опишите таблицы своей БД (таблица 15) с учетом выбранной СУБД.
3. Установите выбранную СУБД на персональный компьютер, изучите особенности работы в ней.
4. Изучив материал по источникам [2, 3, 5], определите для работы права доступа, кодировки, методы доступа, настройки.
5. Укажите, для каких столбцов таблиц будут автоматически созданы кластеризованные индексы, для каких – целесообразно создать некластеризованные.
6. С помощью любого средства проектирования постройте модель базы данных на физическом уровне.

Если выбрана СУБД SQL Server и среда разработки Management Studio, то ознакомьтесь с официальной документацией по работе в них [3]. Некоторые элементы соответствующего интерфейса представлены в приложении Г.

Содержание отчета

1. Цель работы.
2. Краткие теоретические сведения.
3. Описание полученной физической модели базы данных: описание таблиц, их характеристики, определение прав доступа, кодировок символов, методов доступа, настроек СУБД, индексов.
4. Графическое изображение спроектированной модели базы данных на физическом уровне.
5. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №9
РЕАЛИЗАЦИЯ ОПЕРАЦИЙ УПРАВЛЕНИЯ СТРУКТУРАМИ БАЗ ДАННЫХ

Цель работы: сформулировать запросы для создания, удаления и модификации структуры базы данных.

Теоретические сведения

Язык структурированных запросов SQL, который является универсальным для работы с базами данных, включает в себя несколько составляющих. Основные из них следующие [6]:

1. DDL (Data Definition Language) – язык определения данных – позволяет создавать и изменять структуру объектов базы данных. Основными командами языка DDL являются: CREATE (создание объекта базы данных), ALTER (изменение объекта базы данных), DROP (удаление объекта базы данных).

2. DML (Data Manipulation Language) – язык манипулирования данными – используется для манипулирования информацией внутри объектов реляционной базы данных посредством следующих команд: INSERT (вставка данных), UPDATE (изменение данных), DELETE (удаление данных). Некоторые источники добавляют и команду создания запросов SELECT, некоторые – выделяют в отдельную составляющую языка SQL.

Также выделяют:

– команды, связанные с работой разных пользователей, наделением их привилегиями, в частности, GRANT (наделить привилегией), REVOKE (отозвать привилегию);

– команды по управлению транзакциями и др.

В данной лабораторной работе мы изучим команды языка DDL и начнем рассматривать DML на примере диалекта, реализующего работу в СУБД SQL SERVER – Transact-SQL.

Команда CREATE

***Создание базы данных*:**

1. Запустите MS SQL Server.

2. Откройте редактор запросов, нажав на панели инструментов Management Studio кнопку *Создать запрос* (Ctrl + N).

3. В открывшемся окне создайте новый запрос: Create Database Name.

4. Запустите запрос на выполнение, обновите базу данных. Созданная БД должна отобразиться в списке баз данных.

Создание таблиц базы данных и связей между ними

Пример создания таблицы был приведен в лабораторной работе №8. Дополним созданный код реализацией правил ссылочной целостности.

Полный скрипт базы данных со всеми рассматриваемыми объектами, заполненными данными таблицами, будет представлен на занятиях.

```
Create database Poliklinika
Use Poliklinika
	Create table Doctor
	(
		id_doctor int identity(1,1) Primary Key,
		id_speciality int null,
		surname nvarchar(50) not null,
		name nvarchar(50) not null,
		patronymic nvarchar(50) not null
	)

	Create table Patient
	(
		id_patient int identity(1,1) Primary Key,
		card_numbe int unique not null,
		surname nvarchar(50) not null,
		name nvarchar(50) NOT null,
		patronymic nvarchar(50) not null,
		room_number int not null,
		id_pol int not null,
		id_doctor int not null,

	Constraint FK_Patient_Doctor

	Foreign Key(id_doctor) references Doctor (id_doctor)

	On delete no action on update cascade
	)
```

!!! ЗАМЕЧАНИЕ. *Создание связи можно осуществить и через команду Alter уже после создания таблицы (см. лабораторную работу №8):*

```
Alter table Patient
Add constraint FK_Patient_Doctor
Foreign Key(id_doctor) references Doctor (id_doctor)
On delete no action on update cascade
```

Команда ALTER

Данная команда позволяет вносить изменения в структуру таблиц, добавлять, удалять столбцы, изменять их свойства и др. Приведем несколько примеров применения команды ALTER.

```
--Добавляем столбец в таблицу
Alter table Doctor
Add experience int;

--Удаляем столбец из таблицы
Alter table Doctor
Drop column experience;
```

```
--Изменяем тип данных столбца
Alter table Doctor
Alter column id_speciality bigint;
```

!!! ЗАМЕЧАНИЕ. *Для переименования таблицы или столбца применяется конструкция sp_rename:*

```
sp_rename 'Doctor', 'Vrach';        -- и обратно:
sp_rename 'Vrach', 'Doctor';
```

Команда DROP

Данная команда удаляет объект из базы данных или структурный элемент из объекта (например, столбец из таблицы):

1. Создадим таблицу Doctor_2, а затем удалим ее:

```
Create table Doctor_2
(
	[id_doctor] [int] identity(1,1) not null,
	[id_speciality] [int] null,
	[surname] [nvarchar](50) null,
	[name] [nvarchar](50) null,
	[patronymic] [nvarchar](50) null
)

Drop table Doctor_2
```

2. Удалим столбец patronymic из таблицы Doctor:

```
Alter Table [dbo].[Doctor]
Drop Column [patronymic]
```

Рассмотрим также команды языка DML, которые работают не с объектами (структурами) базы данных, а с самими данными.

Ввод данных посредством выполнения инструкции INSERT

Ввод данных в таблицу производится следующим образом:

```
Insert into Таблица (Col1, Col2, ….)
Values    (Значение_ Col1, Значение_ Col2, …), // одна строка
                ….
          (Значение_ Col1, Значение_ Col2, …), // n-я строка
```

Для примера рассмотрим заполнение таблицы Doctor:

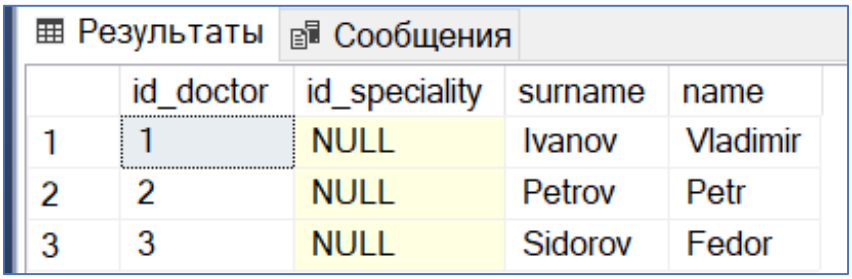
```
Insert into Doctor ([surname],[name])
Values ('Ivanov','Ivan'), ('Petrov','Petr'), ('Sidorov','Fedor')
```

Как видно из примера, заполнены столбцы, для которых установлено свойство NOT NULL. При этом значения столбца id_doctor (PK) формируются автоматически. Значения столбца id_speciality (FK) будут равны NULL.

Обновление (изменение) данных с помощью UPDATE

Обновим данные для врача по фамилии «Ivanov», изменим его имя на «Vladimir» (рисунок 14):

```
Update Doctor
Set [name] = 'Vladimir'
Where [surname] = 'Ivanov'
```



Результаты | Сообщения

	id_doctor	id_speciality	surname	name
1	1	NULL	Ivanov	Vladimir
2	2	NULL	Petrov	Petr
3	3	NULL	Sidorov	Fedor

Рисунок 14 – Обновление данных таблицы Doctor

Без использования условия в строке Where обновятся все значения в соответствующем столбце. Например, после выполнения следующего скрипта имена всех врачей изменятся на имя Ivan:

```
Update Doctor
Set [name] = 'Ivan'
```

Удаление данных с помощью DELETE

Удалим из таблицы Doctor записи для сотрудников по фамилии «Ivanov»:

```
Delete
From [dbo].[Doctor]
Where [surname]='Ivanov'
```

!!! ЗАМЕЧАНИЕ. *При удалении, обновлении и вставке данных следует помнить про правила соблюдения ссылочной целостности и их свойства, установленные при создании таблиц, связей и ограничений.*

Порядок выполнения лабораторной работы

1. Создайте базу данных, соответствующую вашему варианту, выданному на первом занятии. Название базы данных сформируйте следующим образом: *Номер группы_Фамилия_Вариант*, например, *81071_Ivanov_1*.
2. Создайте таблицы и связи между ними.

3. Установите возможность каскадного удаления и обновления данных, при необходимости определите и другие механизмы поддержания действий по обеспечению ссылочной целостности в базе данных.

4. Заполните каждую из таблиц валидными данными не менее чем на пять строк.

5. Убедитесь, что для столбцов – первичных ключей таблиц – автоматически созданы индексы, что эти индексы – кластеризованные.

6. Создайте несколько некластеризованных индексов для других столбцов таблиц базы данных. Аргументируйте выбор столбцов.

7. Реализуйте простые действия с помощью оператора ALTER TABLE по изменению структуры таблицы (не менее трех) и свойств столбцов.

8. Создайте таблицу-копию одной из таблиц базы данных. С помощью команды DROP удалите ее.

9. Создайте запрос на изменение данных одной из таблиц с применением и без применения оператора Where.

10. Оформите отчет по лабораторной работе (включите все скрипты и диаграмму БД).

Содержание отчета

1. Цель работы.
2. Краткие теоретические сведения.
3. Все скрипты и скриншоты результатов их выполнения для базы данных.
4. Обновленная диаграмма базы данных с указанием свойств столбцов таблиц.
5. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №10
РЕАЛИЗАЦИЯ ОПЕРАЦИЙ УПРАВЛЕНИЯ ДАННЫМИ

Цель работы: познакомиться с оператором SELECT языка SQL, создать запросы на выборку данных на языке SQL и реализовать их в СУБД.

Теоретические сведения

Создание запросов

Синтаксис классического однотабличного запроса *с условием* на T-SQL выглядит следующим образом:

```
Select Col_1, Col_2, ….
From Table_Name
Where Условие выборки
```

В предложении **Select** перечисляются те столбцы таблицы, указанной в предложении **From**, которые необходимо отобразить в результате запроса. В предложении **Where** указывается условие отбора данных.

Далее для примера запросы будут созданы для базы данных Poliklinika, диаграмма которой была изображена на рисунке 13.

!!! ЗАМЕЧАНИЕ. *Символьные значения в запросах заключаются в одинарные кавычки. Числовые константы записываются без кавычек.*

Пример 1. Создать запрос, выводящий на экран фамилию и имя тех врачей, которые расположены в списке после номера 3:

```
Select surname, name
From Doctor
Where id_doctor > 3
```

Представления (Views)

Отметим, что создаваемые в БД запросы не сохраняются в качестве объектов базы данных и не отображаются в окне обозревателя. Поэтому, например, их код можно сохранить в специально созданный каталог на диске.

!!! ЗАМЕЧАНИЕ. *Для сохранения созданного запроса в БД целесообразно сохранить его в виде представления.*

Представление – это виртуальная (не физическая) таблица, которая содержит выбранные поля и записи из одной или нескольких таблиц, или других представлений. Представление не содержит данные, оно зависит от тех таблиц, которые участвовали в его создании: при изменении данных и структуры в таблицах происходят изменения и в созданном представлении.

Выборка из представления осуществляется так же, как из таблиц, хотя другие операции имеют ряд ограничений.

Для создания представления будем использовать следующую конструкцию:

```
Create View View_Name
as
Select ….
From ….
[Where ….]
….
```

После создания представления запускаем его на выполнение (кнопка *Выполнить* или F5). Обновляем БД, открываем узел *Представления*. Созданное представление отобразится в данном узле.

Для внесения изменений в код представления можно воспользоваться командой Alter или выбрать контекстное меню данного объекта и проследовать по цепочке команд, указанных на рисунке 15.

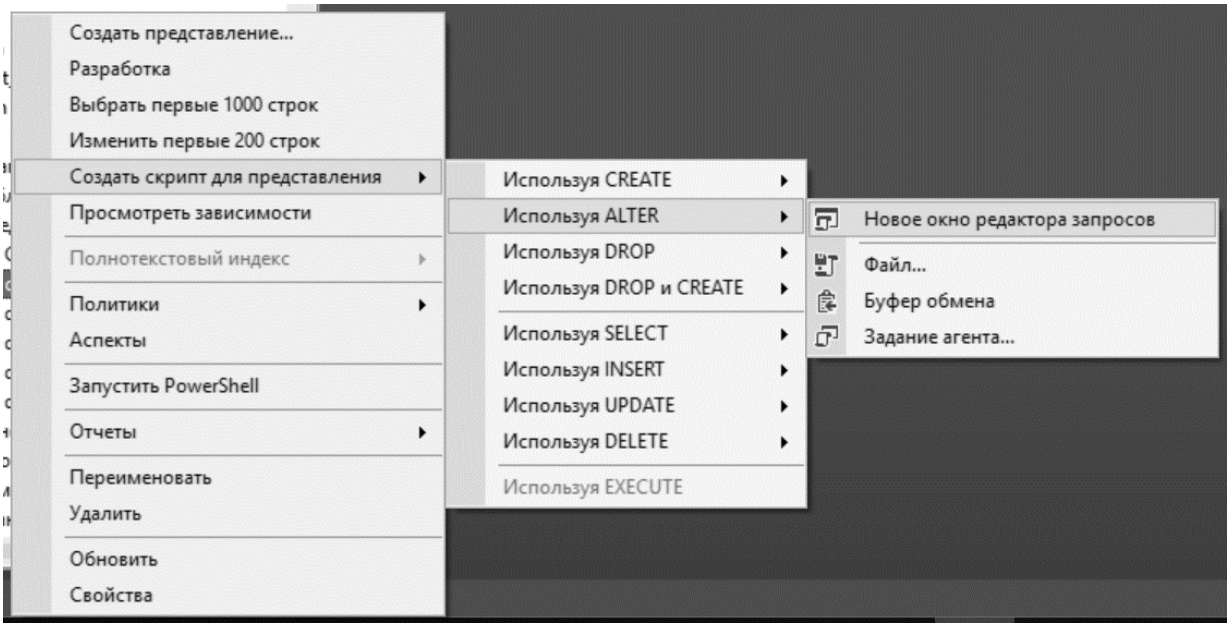


Рисунок 15 – Открытие скрипта объекта БД (View) для изменения

Создание запросов по обработке строковых значений

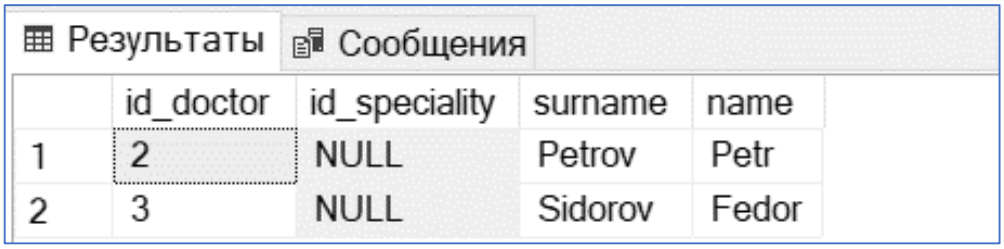
Поиск с отрицанием NOT. Любое логическое выражение можно превратить в его отрицание, используя оператор NOT.

Добавим строку с данными в таблицу Doctor:

```
Insert into Doctor ([surname],[name])
Values ('Ivanov','Vasil')
```

Пример 2. Необходимо выбрать все записи из таблицы Doctor кроме тех, которые относятся к врачам по фамилии «Ivanov» (рисунок 16):

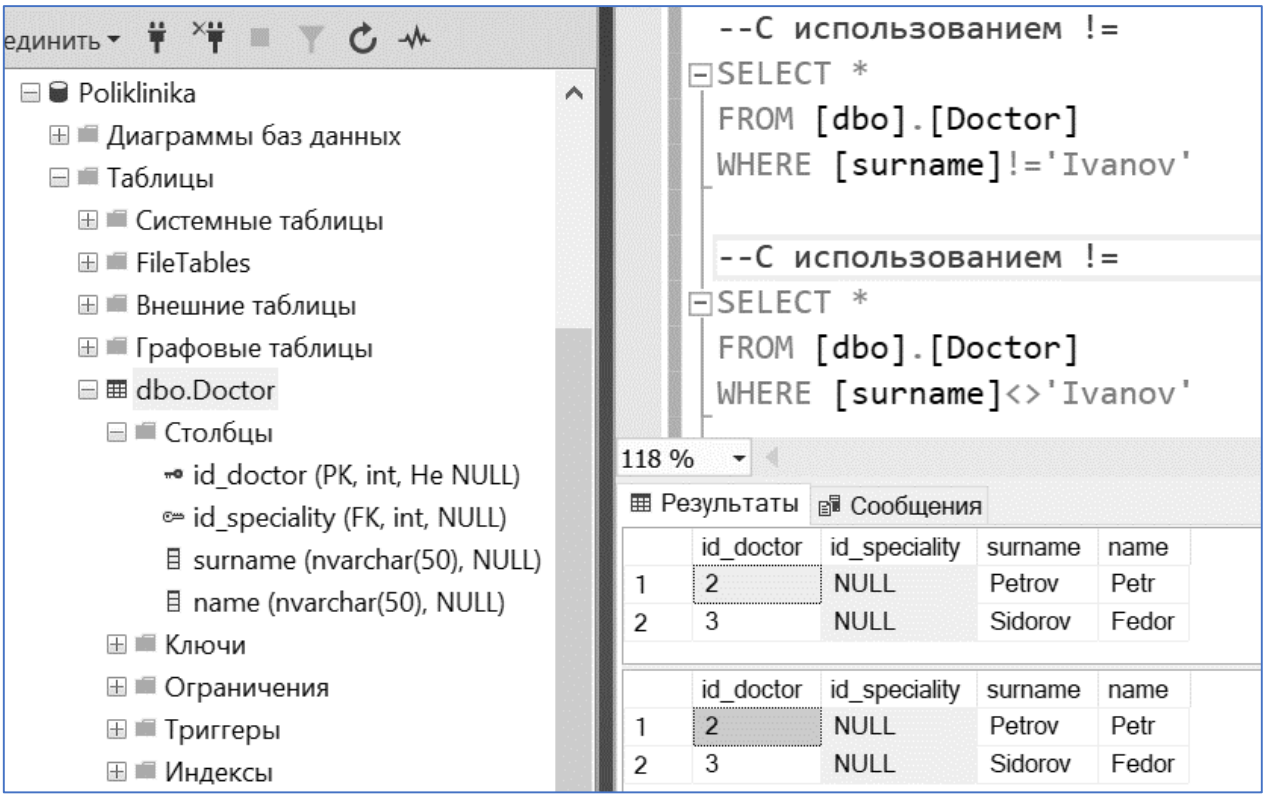
```
--С использованием NOT
Select *
From [dbo].[Doctor]
Where not [surname]='Ivanov'
```



	id_doctor	id_speciality	surname	name
1	2	NULL	Petrov	Petr
2	3	NULL	Sidorov	Fedor

Рисунок 16 – Создание запроса с помощью отрицания и его результат

Есть и другие способы получить этот же результат, поставив вместо NOT следующие операторы (рисунок 17).



	id_doctor	id_speciality	surname	name
1	2	NULL	Petrov	Petr
2	3	NULL	Sidorov	Fedor

	id_doctor	id_speciality	surname	name
1	2	NULL	Petrov	Petr
2	3	NULL	Sidorov	Fedor

Рисунок 17 – Разные способы создания отрицания в запросе и его результат

Применение сравнения LIKE. Кроме сравнения непосредственно со значением, условия поиска могут содержать специальный шаблон.

Сравнение LIKE использует следующие символы-заменители (таблица 16).

Таблица 16 – Символы-заменители при использовании сравнения LIKE в условии запроса SQL

Символ-заменитель	Характеристика
% (знак процента)	Любое количество любых символов
_ (знак подчеркивания)	Один произвольный символ
[ ] (квадратные скобки)	Любой одиночный символ, содержащийся в диапазоне. Например, ([a-f]) или ([abcdef])
[^] (отрицание)	Любой одиночный символ, не содержащийся в диапазоне. Например, ([^a-f]) или ([^abcdef])

Дополним таблицу Doctor данными:

```
Insert into Doctor ([surname],[name])
Values ('Kazakov','Petr'), ('Kulinich','Dmitri');
```

Пример 3. Создать запрос для поиска врачей, фамилии которых начинаются на букву «K» (буквы латинского алфавита):

```
Select *
From [dbo].[Doctor]
Where [surname] like 'K%'
```

Применение логических операторов AND и OR. Несколько условий поиска можно объединить в одном предложении Where посредством использования логических операторов **AND** и **OR**, а также отрицания.

Оператор AND объединяет условия, каждое из которых должно быть выполнено обязательно.

Оператор OR объединяет условия, среди которых должно быть выполнено хотя бы одно.

!!! ЗАМЕЧАНИЕ. *Следует помнить о приоритете логических операций: наибольший приоритет у отрицания, затем – у операции AND, наименьший приоритет имеет операция OR. Для изменения приоритета в выражении, содержащем несколько логических операторов, следует применить скобки.*

Пример 4. Создать запрос для выбора врачей, фамилии которых заканчиваются на ov, имеющих имена Ivan или Fedor и номер по списку, не равный 2:

```
Select *
From [dbo].[Doctor]
Where [surname]    like '%ov'     and
        ([name]    like 'Ivan'    or
        [name] like 'Fedor')      and
        [id_doctor] != 2
```

!!! ЗАМЕЧАНИЕ. *Для проверки наличия/отсутствия неопределенного значения NULL/NOT NULL нельзя использовать математические знаки «=», «<>». Для этого применяется конструкция IS NULL или IS NOT NULL.*

Дополним таблицу Speciality данными (она является родительской для таблицы Doctor):

```
Insert into [dbo].[Speciality]
Values ('терапевт'), ('стоматолог'), ('психолог');
```

Внесем изменения в таблицу Doctor (рисунок 18).

	id_doctor	id_speci...	surname	name
	2	1	Petrov	Petr
	3	*NULL*	Sidorov	Fedor
	4	2	Ivanov	Vasil
	5	*NULL*	Kazakov	Petr
	6	3	Kulinich	Dmitri
▶*	*NULL*	*NULL*	*NULL*	*NULL*

Рисунок 18 – Измененные данные таблицы Doctor

Пример 5. Вывести данные о врачах, для которых не определен код их специальности (результат выполнения представлен на рисунке 19).

```
--Запрос с ошибкой
Select *
From [dbo].[Doctor]
Where [id_speciality] = NULL

--Верный запрос
Select *
From [dbo].[Doctor]
Where [id_speciality] is NULL
```

118 %

Результаты | Сообщения

	id_doctor	id_speciality	surname	name
1	3	NULL	Sidorov	Fedor
2	5	NULL	Kazakov	Petr

Рисунок 19 – Запросы и их результаты

Поиск с использованием оператора IN. В данном случае будут отбираться значения, соответствующие указанным в списке, которые отделяются друг от друга запятыми. В списке возможно применение оператора NOT.

Пример 6. Создать запрос, который возвращает данные о врачах, работающих по следующим специальностям: терапевт или стоматолог, т. е. данные о врачах, для которых id_speciality равно 1 или 2 (рисунок 20).

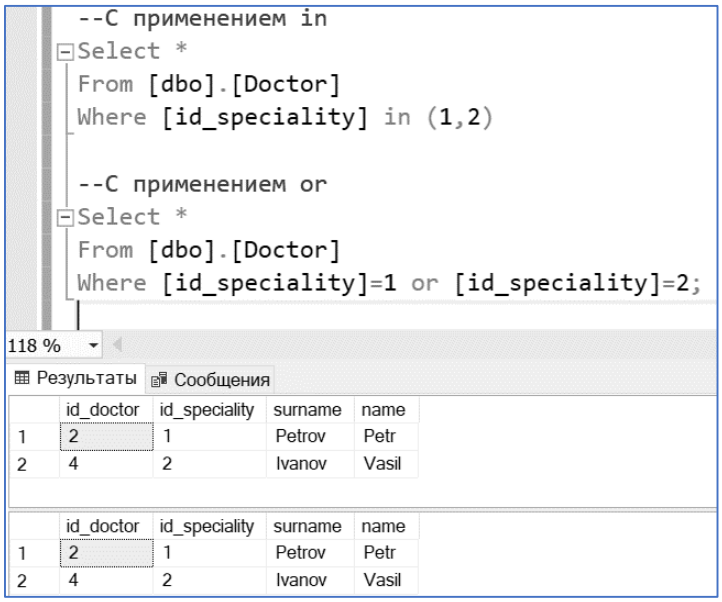


Рисунок 20 – Запрос с применением оператора IN и его результаты

Поиск с использованием оператора BETWEEN. Данный оператор проверяет попадание искомого значения в заданный диапазон.

Пример 7. Создать запрос, который возвращает данные о врачах с идентификатором от 2 до 5. Предложите несколько вариантов выполнения:

1) Select *
   From [dbo].[Doctor]
   Where [id_doctor] between 2 and 5

2) Select *
   From [dbo].[Doctor]
   Where [id_doctor]>=2 and [id_doctor]<=5

3) Select *
   From [dbo].[Doctor]
   Where [id_doctor] in (2,3,4,5)

4) Select *
   From [dbo].[Doctor]
   Where [id_doctor]=2 or [id_doctor]=3 or [id_doctor]=4 or [id_doctor]=5

Простые запросы с группировкой. Агрегатные функции

Синтаксис классического однотабличного запроса с условием на T-SQL может быть дополнен следующим образом:

```
Select Col_1, Col_2, ….
From Table_Name
Where Условие выборки
Group by Список полей группировки
Having Условие выборки для группы
Order by Список полей для сортировки (ASC – по возрастанию (по умолчанию), DESC – по убыванию)
```

Внесем некоторые пояснения:

– предложение Group by объединяет (группирует) в одну запись все записи, которые содержат одинаковые значения в указанном столбце таблицы;

– при этом предложение Where определяет, какие записи должны участвовать в группировании, т. е. фильтрует их до группирования;

– группировка позволяет применять агрегатные функции.

Агрегатные функции еще называют итоговыми, суммирующими. Они обрабатывают набор строк для подсчета и возвращения одного значения.

К основным агрегатным функциям относят следующие:

1. **COUNT( )** – возвращает количество строк результата запроса или в группе;
2. **MAX( )** – возвращает максимальное значение в диапазоне;
3. **MIN( )** – возвращает минимальное значение в диапазоне;
4. **AVG( )** – возвращает среднее значение данных в диапазоне;
5. **SUM( )** – возвращает сумму значений диапазона данных.

Агрегатные функции также могут определять соответствующие им значения при выполнении условия в предложении Where.

Дадим некоторые комментарии по применению агрегатных функций:

– обычно в качестве выражения, которое принимает агрегатная функция, выступает название столбца таблицы, для значений которого проводится вычисление;

– функции AVG и SUM работают только со значениями числового типа;

– функции MIN, MAX и COUNT могут обрабатывать как числовые, так и строковые значения, а также значения типа «дата»;

– все агрегатные функции, кроме COUNT (*), игнорируют значения NULL.

!!! ЗАМЕЧАНИЕ. *Выборка может содержать повторяющиеся значения. Если необходимо выполнить вычисления только над уникальными значениями столбца таблицы, исключив из набора значений повторяющиеся данные, то применяется оператор* ***DISTINCT****.*

Разберемся с частью запроса Having. Здесь, как и в Where, формируется условие, но есть особенности применения этих двух предложений (таблица 17).

Таблица 17 – Некоторые особенности применения Where и Having

Where	Having
Отбирает строки в соответствии с условием ***до*** группирования и вычисления агрегатных функций	Отбирает строки ***после*** группирования и вычисления агрегатных функций.
Не содержит агрегатные функции	***Содержит*** агрегатные функции

Таким образом, последовательность применения всех частей в запросе должна строго соблюдаться.

Пример 8. Дополнить таблицу Doctor столбцом experience (стаж работы) и заполнить произвольными числами от 1 до 15. Создать запросы, определяющие:

– количество врачей в поликлинике, средний стаж работы врачей через функцию AVG, наибольший и наименьший стаж среди врачей;

– количество врачей по фамилии «Ivanov»;

– номер(а) специальности(ей), по которым со стажем более пяти лет работает более двух врачей.

Добавим столбец в таблицу, заполним его значениями (рисунок 21), создадим новые строки с данными.

id_doctor	id_speci...	surname	name	experien...
2	1	Petrov	Petr	5
3	1	Sidorov	Fedor	2
4	2	Ivanov	Vasil	7
5	2	Kazakov	Petr	12
6	3	Kulinich	Dmitri	10
7	3	Ivanov	Pavel	8
8	2	Ivanov	Anton	9
NULL	NULL	NULL	NULL	NULL

Рисунок 21 – Таблица Doctor с обновленными данными

Результаты выполнения запросов представлены на рисунках 22–24.

```
Select Count(*) AS Кол_Врачей,
       Avg(experience) AS CP_Стаж,
       Max(experience) AS Max_Стаж,
       Min(experience) AS Min_Стаж
From [dbo].[Doctor]
```

18 %

Результаты | Сообщения

	Кол_Врачей	CP_Стаж	Max_Стаж	Min_Стаж
1	7	7.571428	12	2

Рисунок 22 – Вычисление агрегатных функций

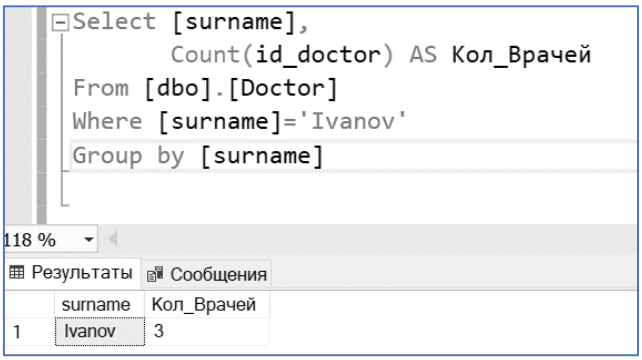


Рисунок 23 – Применение группирования

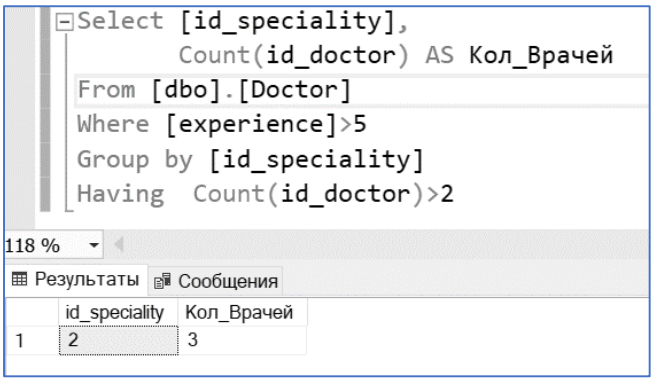


Рисунок 24 – Использование условия выборки для группы Having

Вычисляемые столбцы

Для вычисления некоторых значений создают запросы с вычисляемыми столбцами. Такие столбцы можно создать и в таблицах, но это приведет к нарушению третьей нормальной формы.

Пример 9. Пусть некоторая часть заработной платы зависит от стажа (не учитываем сейчас премию и другие надбавки) и высчитывается следующим образом: *Базовая ставка оклада** (*стаж/100*) + *Базовая ставка оклада.* Вычислить эту часть заработной платы при условии, что базовая ставка оклада равна 700 ден. ед.

Соответствующий код и результат выполнения представлены на рисунке 25.

Рисунок 25 – Создание вычисляемого столбца в тексте запроса

!!! ЗАМЕЧАНИЕ. *Для выполнения примера 9 использовалась функция CONVERT (новый тип, выражение), которая конвертирует получаемый результат в новый тип данных.*

Подзапросы

С помощью SQL возможно вкладывать запросы друг в друга.

Использование агрегатных функций в подзапросах

Пример 10. Создать запрос по выводу всех данных о работниках, стаж которых больше среднего значения стажа по предприятию.

Искомое множество будет состоять из последних четырех записей таблицы, изображенной на рисунке 25.

Попробуем выполнить данное задание следующим образом: используем условие в предложении Where, а также сформируем запрос с помощью группирования и условия в предложении Having.

В данном случае получим ошибку при выполнении либо ошибочный результат – пустое множество (рисунок 26).

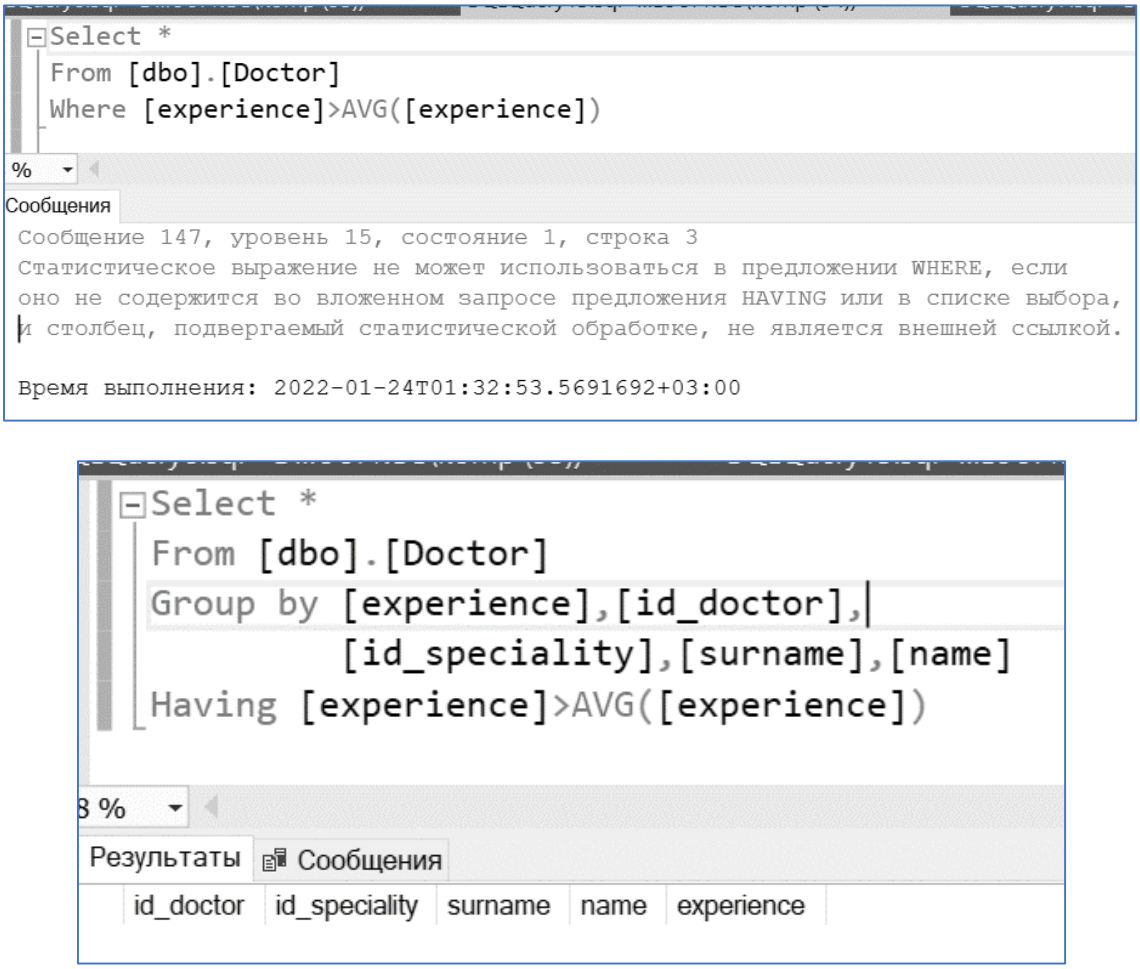
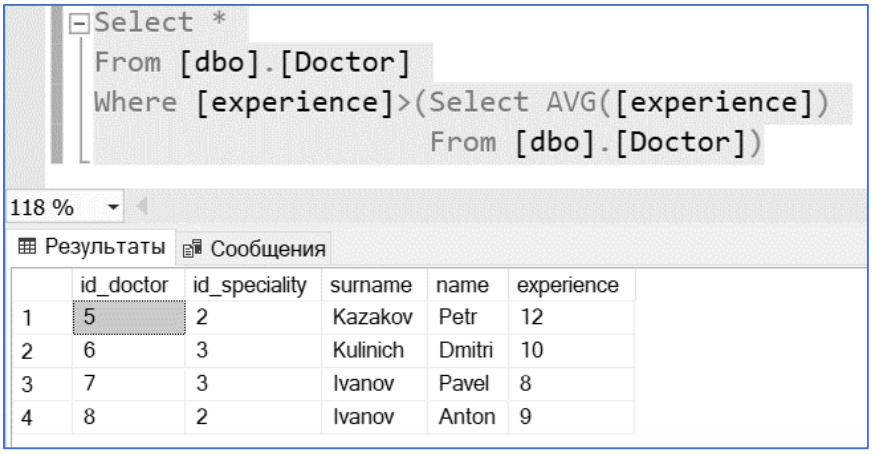


Рисунок 26 – Ошибочное применение условий при работе с агрегатными функциями

Что же здесь не так? При использовании обычного запроса мы не получим нужный результат, поскольку сравнение стажа конкретного врача происходит со средним значением стажа по строке, а не со средним значением стажа по предприятию.

К тому же в обоих случаях сравнение происходит со значением стажа, которое вычисляется ***после*** сравнения.

Для получения искомого результата необходимо использовать подзапрос: в нем сначала вычисляется среднее значение стажа по предприятию, а затем оно передается в основной запрос, в котором и происходит сравнение (рисунок 27).



	id_doctor	id_speciality	surname	name	experience
1	5	2	Kazakov	Petr	12
2	6	3	Kulinich	Dmitri	10
3	7	3	Ivanov	Pavel	8
4	8	2	Ivanov	Anton	9

Рисунок 27 – Применение подзапроса при работе с агрегатными функциями

Применение подзапросов при работе со связанными таблицами

Допустим, что в базе данных есть таблицы T1 и T2, которые связаны между собой через ключевое поле id (для T1 – это первичный ключ, для T2 – соответствующий внешний ключ). В общем случае синтаксис такого запроса возможно прописать следующим образом:

```
Select T1.Col1, …, T1.Coln
From  T1
Where T1.id IN (Select T2.id
                From T2
                Where Условие) and/or …
```

В схему базы данных Poliklinika включено четыре таблицы, которые последовательно связаны между собой отношениями «один ко многим» (см. рисунок 13).

Пример 11. Вывести названия специальностей, по которым работают врачи с фамилией «Ivanov».

В данном запросе будут участвовать две таблицы: Doctor и Speciality, которые связаны через ключ id_speciality. Код запроса и результат его выполнения представлен на рисунке 28.

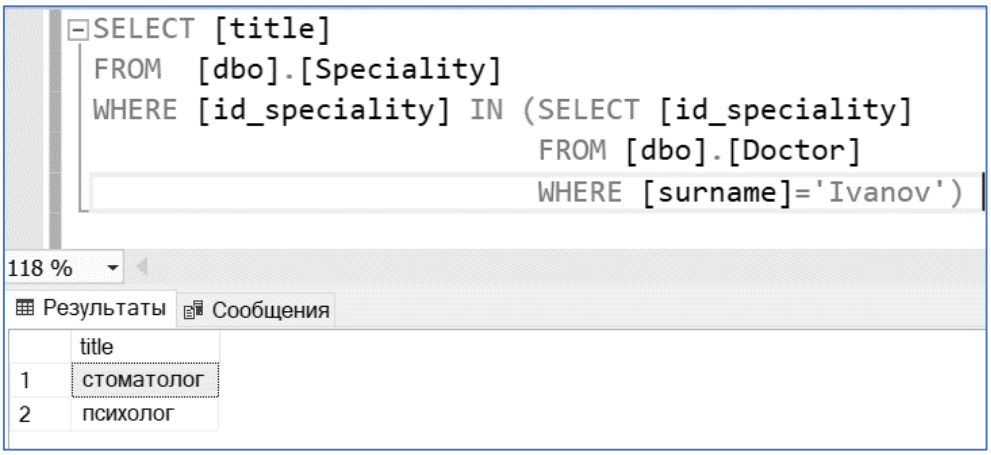


Рисунок 28 – Применение подзапроса при работе со связанными таблицами

Оператор JOIN

Реляционная операция *соединение* реализуется посредством оператора JOIN. В общем виде синтаксис работы с этим оператором можно представить следующим образом (на примере таблиц T1 и T2, которые связаны между собой через ключевое поле id (для T1 – это первичный ключ, для T2 – соответствующий внешний ключ)):

```
Select T1.Col1, …, T1.Coln,
       T2.Col1, …, T2.Coln,
From  T1
       JOIN
       T2
       ON T1.id = T2.id
Where Условие
```

Операция соединения позволяет выводить в результирующем множестве необходимые поля не только из первой, но и из всех связанных таблиц.

Пример 12. Вывести фамилии, имена, стаж и специализацию врачей, чей стаж работы более четырех лет. Результат отсортировать по возрастанию стажа работы.

Выполнение задания представлено на рисунке 29.

```
SELECT [surname],[name],[title],[experience]
FROM  [dbo].[Speciality]
        JOIN
      [dbo].[Doctor]
ON [dbo].[Speciality].[id_speciality]=[dbo].[Doctor].[id_speciality]
WHERE [experience]>4
ORDER BY [experience]
```

98 %

Результаты | Сообщения

	surname	name	title	experience
1	Petrov	Petr	терапевт	5
2	Ivanov	Vasil	стоматолог	7
3	Ivanov	Pavel	психолог	8
4	Ivanov	Anton	стоматолог	9
5	Kulinich	Dmitri	психолог	10
6	Kazakov	Petr	стоматолог	12

Рисунок 29 – Запрос с применением оператора JOIN

Операция соединения может также реализовываться, в частности, через операторы LEFT JOIN, RIGHT JOIN, FULL JOIN, CROSS JOIN.

Добавим в таблицу Doctor данные (см. рисунок 18), намеренно не заполняя значениями id_speciality (рисунок 30).

id_doctor	id_speciality	surname	name	experience
2	1	Petrov	Petr	5
3	1	Sidorov	Fedor	2
4	2	Ivanov	Vasil	7
5	2	Kazakov	Petr	12
6	3	Kulinich	Dmitri	10
7	3	Ivanov	Pavel	8
8	2	Ivanov	Anton	9
9	*NULL*	Rakova	Ilona	2
10	*NULL*	Medentsova	Ludmila	7
11	*NULL*	Zavialova	Valentina	14
NULL	*NULL*	*NULL*	*NULL*	*NULL*

Рисунок 30 – Обновленная таблица Doctor

Посмотрите на следующие рисунки и проведите аналогию с известными вам реляционными операциями (рисунки 31–33).

```
SELECT [title],D.[id_speciality],[surname],[name]
FROM [dbo].[Speciality] S
     LEFT JOIN
     [dbo].[Doctor] D
ON S.id_speciality=D.id_speciality
```

96 %

Результаты | Сообщения

	title	id_speciality	surname	name
1	терапевт	1	Petrov	Petr
2	терапевт	1	Sidorov	Fedor
3	стоматолог	2	Ivanov	Vasil
4	стоматолог	2	Kazakov	Petr
5	стоматолог	2	Ivanov	Anton
6	психолог	3	Kulinich	Dmitri
7	психолог	3	Ivanov	Pavel

Рисунок 31 – Запрос с применением оператора LEFT JOIN

```
SELECT [title],D.id_speciality,[surname],[name]
FROM [dbo].[Speciality] S
     RIGHT JOIN
     [dbo].[Doctor] D
ON S.id_speciality=D.id_speciality
```

96 %

Результаты | Сообщения

	title	id_speciality	surname	name
1	терапевт	1	Petrov	Petr
2	терапевт	1	Sidorov	Fedor
3	стоматолог	2	Ivanov	Vasil
4	стоматолог	2	Kazakov	Petr
5	психолог	3	Kulinich	Dmitri
6	психолог	3	Ivanov	Pavel
7	стоматолог	2	Ivanov	Anton
8	NULL	NULL	Rakova	Ilona
9	NULL	NULL	Medentsova	Ludmila
10	NULL	NULL	Zavialova	Valentina

Рисунок 32 – Запрос с применением оператора RIGHT JOIN

```
Select
    [surname],
    [name],
    [title],
    [dbo].[Doctor].[id_speciality]
From  [dbo].[Speciality]
Cross Join [dbo].[Doctor]
```

	surname	name	title	id_speciality
1	Petrov	Petr	терапевт	1
2	Sidorov	Fedor	терапевт	1
3	Ivanov	Vasil	терапевт	2
4	Kazakov	Petr	терапевт	2
5	Kulinich	Dmitri	терапевт	3
6	Ivanov	Pavel	терапевт	3
7	Ivanov	Anton	терапевт	2
8	Rakova	Ilona	терапевт	NULL
9	Medentsova	Ludmila	терапевт	NULL
10	Zavialova	Valentina	терапевт	NULL
11	Petrov	Petr	стоматолог	1
12	Sidorov	Fedor	стоматолог	1
13	Ivanov	Vasil	стоматолог	2
14	Kazakov	Petr	стоматолог	2
15	Kulinich	Dmitri	стоматолог	3
16	Ivanov	Pavel	стоматолог	3
17	Ivanov	Anton	стоматолог	2
18	Rakova	Ilona	стоматолог	NULL
19	Medentsova	Ludmila	стоматолог	NULL
20	Zavialova	Valentina	стоматолог	NULL
21	Petrov	Petr	психолог	1
22	Sidorov	Fedor	психолог	1
23	Ivanov	Vasil	психолог	2
24	Kazakov	Petr	психолог	2
25	Kulinich	Dmitri	психолог	3
26	Ivanov	Pavel	психолог	3
27	Ivanov	Anton	психолог	2
28	Rakova	Ilona	психолог	NULL
29	Medentsova	Ludmila	психолог	NULL
30	Zavialova	Valentina	психолог	NULL

Рисунок 33 – Запрос с применением оператора CROSS JOIN

Порядок выполнения лабораторной работы

1. Изучите теоретическую часть.
2. Выполните задания, представленные в приложении Д.
3. Продемонстрируйте преподавателю выполнение в СУБД созданных запросов.
4. Сформируйте отчет.

Содержание отчета

1. Цель работы.
2. Краткие теоретические сведения.
3. Описание созданных запросов: скрипты и скриншоты выполнения.
4. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №11
РЕАЛИЗАЦИЯ РАСШИРЕННЫХ ВОЗМОЖНОСТЕЙ УПРАВЛЕНИЯ ДАННЫМИ

Цель работы: изучить принципы действия хранимых процедур, пользовательских функций и триггеров.

Теоретические сведения

Хранимые процедуры

Часто при работе с базой данных используют одни и те же запросы либо их набор. Хранимые процедуры позволяют объединить последовательность запросов, сохранить их и обращаться к ним при необходимости. Таким образом, хранимая процедура позволяет упростить код, повысить безопасность и производительность работы с объектами базы данных.

В общем виде синтаксис создания процедуры может быть представлен следующей структурой:

```
Create procedure Имя_процедуры (параметры)
   Begin
     операторы
   End
```

Параметры – это те данные, которые будут передаваться процедуре при ее вызове. Их имена должны начинаться с символа @ (как и имена переменных). Операторы – это запросы, составляющие тело процедуры.

Пример 13. Создать процедуру вставки новой записи в таблицу Doctor.

Напомним, что указанная таблица состоит из следующих полей с определенными типами данных:

id_doctor – int;
id_ speciality – int;
surname – nvarchar(50);
name – nvarchar(50);
experience – numeric(18, 0).

Следовательно, целесообразно создать процедуру, которая будет получать значения пяти переменных, составляющих запись таблицы. Имена этих переменных целесообразно задать аналогично именам атрибутов таблицы (добавив первый символ @), типы их данных совпадут с типами данных соответствующих полей.

```
Create procedure Proc_Insert
(@id_speciality int,
 @surname nvarchar(50),
 @name nvarchar(50),
 @experience numeric(18,0))
As
Begin
        Insert into [dbo].[Doctor]
```

```
([id_speciality],
 [surname],
 [name],
 [experience])
        Values
(@id_speciality,
 @surname,
 @name,
 @experience);
End
```

Для вызова процедуры и передачи ей значений параметров необходимо записать оператор EXECUTE (или EXEC):

```
Exec Proc_Insert  @id_speciality = 3,
                  @surname = 'Larionich',
                  @name = 'Anfisa',
                  @experience = 35;
```

Для изменения кода процедуры или ее удаления применяются соответственно операторы ALTER и DROP.

!!! ЗАМЕЧАНИЕ. *Хранимая процедура может не иметь параметров.*

Пример 14. Разработать процедуру для получения фамилии, имени, стажа врача под номером 7, а также названия его специальности.

Выполнение примера приведено на рисунке 34.

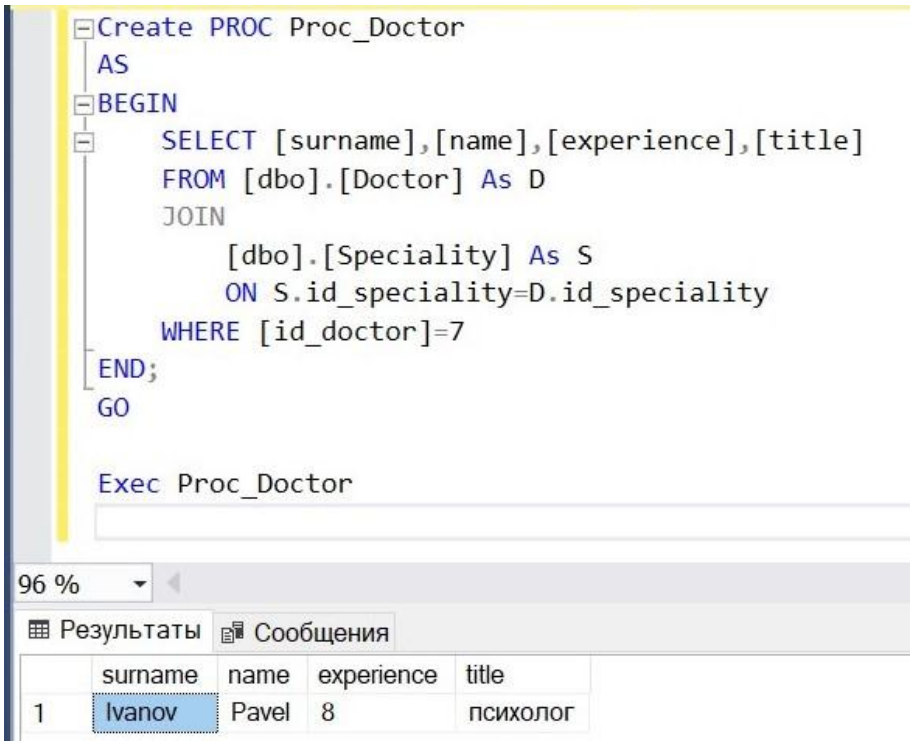


Рисунок 34 – Создание процедуры без параметров, ее вызов, результат

!!! ЗАМЕЧАНИЕ. *Хранимая процедура может иметь выходные параметры.*

Пример 15. Разработать хранимую процедуру, подсчитывающую количество врачей, работающих в поликлинике. Результат вывести в виде текстового сообщения: «В поликлинике работает n врачей», где n – результат выполнения хранимой процедуры.

```
Create procedure Count_vrach_output  @count_vrach int output
  As
Begin
   Select  @count_vrach=count([id_doctor])
   From [dbo].[Doctor];
End;
Go

--Объявляем и инициализируем переменную, в которую будет возвращено значение выходного параметра хранимой процедуры
Declare @count_vrach int;

--Вызываем процедуру
Execute count_vrach_output @count_vrach output;

-- Выводим полученное значение в виде текстового сообщения
Print 'В поликлинике работает ' + Convert(varchar(5),@count_vrach) + ' врача(ей)'
```

Пользовательские функции

Функция, определенная пользователем, – подпрограмма, которая возвращает значение – результат выполнения функции.

В SQL Server возможно создание пользовательских функций, которые возвращают данные в виде скалярного значения или таблицы.

Пример 16. Создать скалярную функцию, возвращающую часть заработной платы врача, которая высчитывается следующим образом:

*Базовая ставка оклада** (*стаж/100*) + *Базовая ставка оклада*,

при условии, что базовая ставка оклада равна 700 ден. ед.

В функцию будем передавать две переменные: базовую ставку – переменную @baz_stav и код врача – переменную @id_doctora. Получать будем результат – часть заработной платы соответствующего врача в формате numeric(10,2) – переменная @otvet.

Соответствующий код, вызов функции и передача ей необходимых значений параметров изображены на рисунке 35.

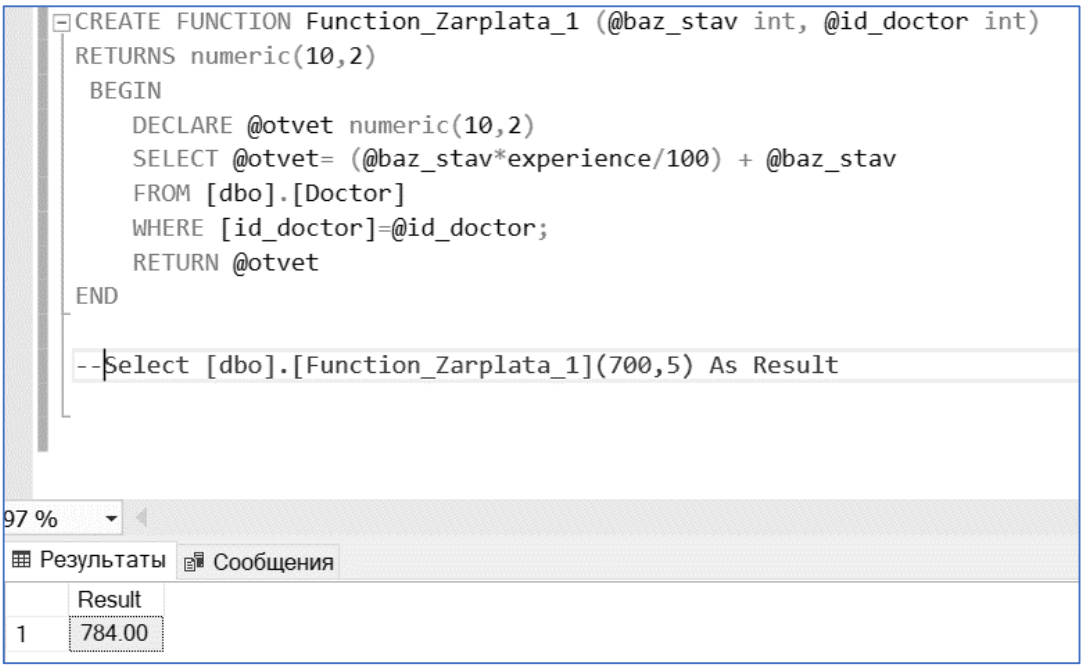


Рисунок 35 – Создание функции и ее вызов

В последней строке кода функции (перед END) возвращается результат. Для этого нужно написать ключевое слово RETURN, после которого пишется возвращаемое значение или переменная. Для вызова и выполнения функции используем оператор SELECT.

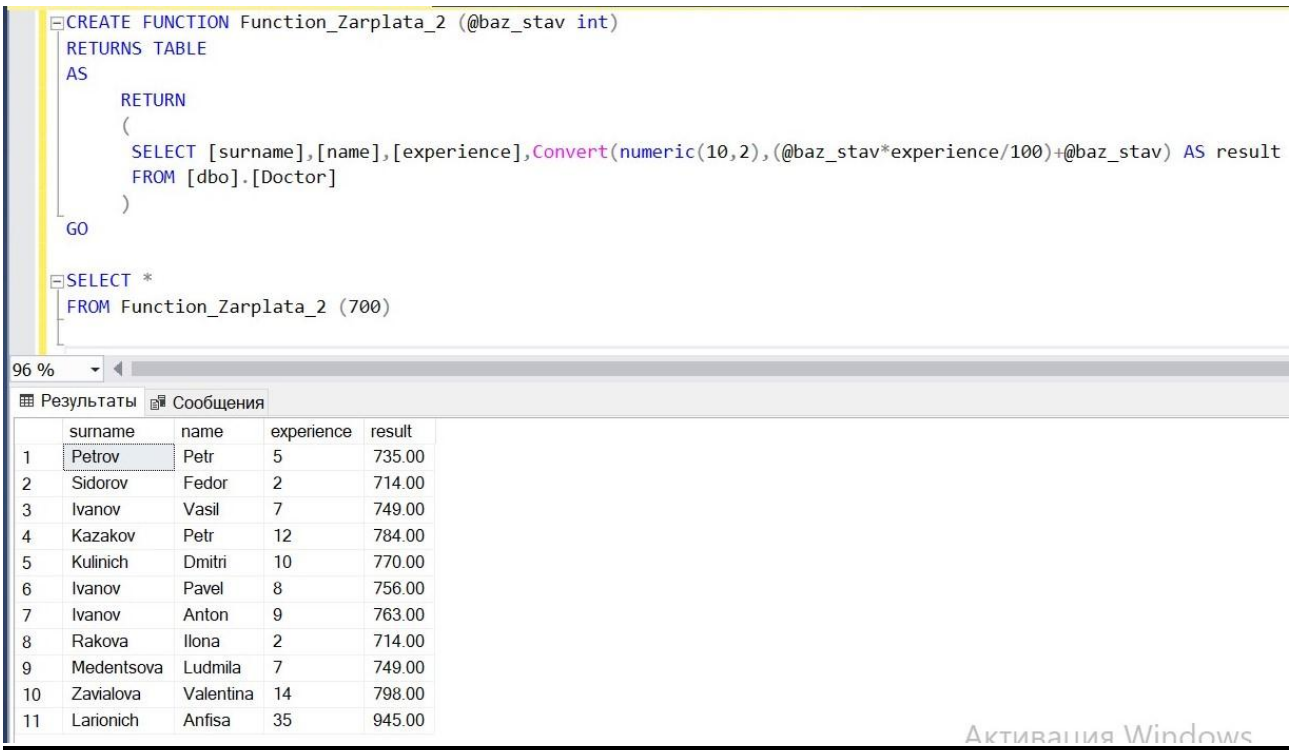
!!! ЗАМЕЧАНИЕ. *Передаваемые значения параметров перечисляются в том же порядке, в котором они были объявлены в функции.*

Обращаться к функции можно напрямую через оператор SET:

```
DECLARE  @otvet numeric(10,2)
SET      @otvet=[dbo].[Function_Zarplata_1](700,5)
PRINT    @otvet
```

Пример 17. Создать функцию, которая будет выводить по переданной ей величине базовой ставки оклада результат вычисления части зарплаты, а также фамилии, имена врачей и стаж их работы.

В данном случае очевидно, что результат будет представлен в виде таблицы. Соответствующий код, вызов функции и результат выполнения представлены на рисунке 36.



```
CREATE FUNCTION Function_Zarplata_2 (@baz_stav int)
RETURNS TABLE
AS
     RETURN
     (
      SELECT [surname],[name],[experience],Convert(numeric(10,2),(@baz_stav*experience/100)+@baz_stav) AS result
      FROM [dbo].[Doctor]
     )
GO

SELECT *
FROM Function_Zarplata_2 (700)
```

96 %

Результаты | Сообщения

	surname	name	experience	result
1	Petrov	Petr	5	735.00
2	Sidorov	Fedor	2	714.00
3	Ivanov	Vasil	7	749.00
4	Kazakov	Petr	12	784.00
5	Kulinich	Dmitri	10	770.00
6	Ivanov	Pavel	8	756.00
7	Ivanov	Anton	9	763.00
8	Rakova	Ilona	2	714.00
9	Medentsova	Ludmila	7	749.00
10	Zavialova	Valentina	14	798.00
11	Larionich	Anfisa	35	945.00

Активация Windows

Рисунок 36 – Создание табличной функции и ее вызов

!!! ЗАМЕЧАНИЕ. *Пользовательская функция может не иметь параметров.*

Триггеры

Триггер – это специальный тип хранимой процедуры, которая вызывается автоматически при выполнении определенных действий в базе данных. Рассмотрим триггеры, которые реагируют на выполнение команд INSERT, UPDATE, DELETE.

!!! ЗАМЕЧАНИЕ. *При создании триггера для операций INSERT и DELETE создаются специальные таблицы INSERTED и DELETED, в которые помещаются вставляемые и удаляемые данные. Эти таблицы доступны только во время срабатывания триггера.*

!!! ЗАМЕЧАНИЕ. *Операция UPDATE – это последовательное выполнение операций DELETE и INSERT, поэтому таблицы UPDATED нет, а изменяемая информация отправляется в таблицы INSERTED и DELETED.*

Триггеры срабатывают при наступлении некоторого события и могут выполняться в зависимости от установок конкретной СУБД до (BEFORE), после (AFTER) или вместо (INSTEAD OF) действия, запускающего триггер.

Пример 18. Построена новая стоматологическая поликлиника, поэтому все вновь пришедшие врачи-стоматологи переводятся туда и на работу в районную

поликлинику не принимаются. Создать триггер, который проверяет специальность врача и не позволяет вставлять новых врачей со специальностью «стоматолог» в таблицу Doctor или изменять их специальность на «стоматолог».

Рассмотрим, как работает триггер (таблица 18).

Таблица 18 – Описание выполнения триггера

Скрипт триггера	Описание выполнения
CREATE TRIGGER Trig_NotInsertUpdate_Stomatolog ON [dbo].[Doctor]	Создается триггер для таблицы [dbo].[Doctor]
AFTER INSERT, UPDATE	Триггер срабатывает на операции вставки и изменения
AS BEGIN DECLARE @id_speciality int	Начинается запись тела триггера. Объявляется целочисленная переменная @id_speciality, которая будет передавать данные в таблицу INSERTED
SELECT @id_speciality=D.id_speciality FROM [dbo].[Doctor] As D, Inserted As I WHERE D.id_doctor=I.id_doctor	Создается запрос, в котором находится информация о добавленной (измененной) записи в таблице
IF @id_speciality=2 Begin	Проверяется критерий отказа на вставку/изменение записи при id_doctor = 2 (соответствует специальности «стоматолог»).
ROLLBACK TRAN RAISERROR('Вставка записи не возможна, т.к. врачи-стоматологи переводятся в новую поликлинику', 16, 10) End	Происходит откат транзакции (вставки, изменения записи) и создается сообщение об ошибке, которое выводится на экран
END	Завершается запись триггера

Триггер запускается на выполнение. После попытки вставки в таблицу Doctor новой записи или изменения в записи специальности под номером 2 появится сообщение об ошибке (рисунок 37).

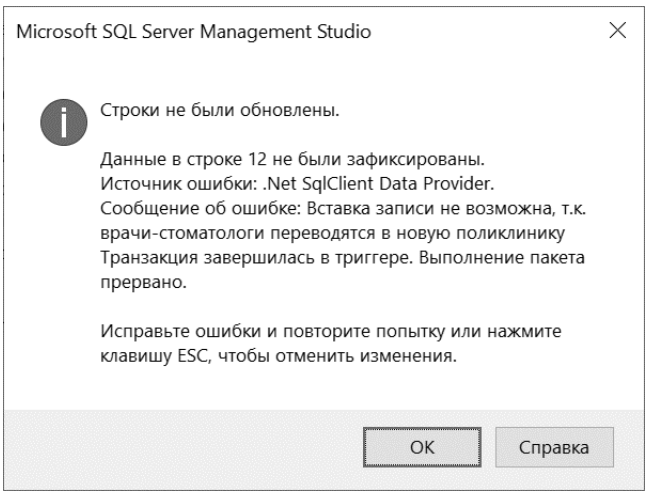


Рисунок 37 – Вывод сообщения об ошибке при срабатывании триггера

Пример 19. Следующий триггер выполняется, когда есть попытка удалить любую запись из таблицы Doctor.

```
Create trigger Trig_delete
On [dbo].[doctor]
After Delete
As
Begin
        If @@Rowcount >= 1
        Begin
        Rollback Tran
        Raiserror ('удаление в таблице невозможно', 16, 10)
        End
End
```

!!! ЗАМЕЧАНИЕ. *Для выполнения триггера в примере 19 использовалась системная функция MS SQL Server @@ROWCOUNT, которая производит подсчет затронутых строк. В нашем случае она считает количество строк, которые пробуют удалить из таблицы. И при любом их количестве происходит откат транзакции.*

!!! ЗАМЕЧАНИЕ. *При удалении таблицы автоматически уничтожаются все связанные с ней объекты, включая триггеры.*

Для изменения кода триггера или его удаления стандартно применяется соответственно операторы ALTER и DROP.

Иногда бывает необходимо не удалить триггер, а отключить его на некоторое время. Выполняется это с помощью следующей конструкции:

```
ALTER TABLE Имя_Таблицы
DISABLE TRIGGER Имя_Триггера
```

Для возобновления работы триггера применяется следующий синтаксис:

```
ALTER TABLE Имя_Таблицы
ENABLE TRIGGER Имя_Триггера
```

Порядок выполнения лабораторной работы

1. Изучите теоретическую часть.
2. Выполните задания, представленные в приложении Е.
3. Продемонстрируйте преподавателю выполнение в СУБД созданных объектов (хранимых процедур, пользовательских функций, триггеров).
4. Сформируйте отчет.

Содержание отчета

1. Цель работы.
2. Краткие теоретические сведения.
3. Описание созданных запросов (скрипты и скриншоты выполнения).
4. Выводы по работе.

ПРИЛОЖЕНИЕ А
ЗАДАНИЯ ДЛЯ ЛАБОРАТОРНОЙ РАБОТЫ №1

Выбор заданий для выполнения определяется из таблицы А.1, где номеру студента в списке подгруппы соответствуют номер на пересечении варианта и номера задания.

Таблица А.1 – Выбор задания

Номер задания	**Вариант 1**	**Вариант 2**	**Вариант 3**	**Вариант 4**	**Вариант 5**
Задание 1	1	2	3	4	5
Задание 2	6	7	8	9	10
Задание 3	11	12	13	14	15

Перечень заданий

Примените следующие реляционные операции над множествами – отношениями (таблицами). За элемент множества примите кортеж отношения (строку таблицы). Учитывайте приоритет выполнения операций (изучите самостоятельно, обратите внимание на замечание на с. 50).

Задание 1:

– (C – B) ⋃ A
– M = A **Where** A.col_1 = ‘*выбрать первые два элемента*’
– N = B **Where** B.col_1 = ‘*выбрать последние два элемента*’
– (M×N) **Projection** {A.col_1, B.col_1}

Задание 2:

– (B ⋃ C) ⋂ A
– M = A **Where** A.col_1 = ‘*выбрать первый элемент*’
– N = C **Where** C.col_1 = ‘*выбрать последние три элемента*’
– (M×N) **Projection** {A.col_2, C.col_2}

Задание 3:

– (A ⋂ C) ⋃ B
– M = B **Where** B.col_1 = ‘*выбрать второй элемент*’
– N = C **Where** C.col_1 = ‘*выбрать первый, третий, пятый элементы*’
– (M×N) **Projection** {B.col_1, C.col_1}

Варианты выполнения заданий

Вариант 1

A

Страна	Столица
Армения	Ереван
Азербайджан	Баку
Беларусь	Минск
Грузия	Тбилиси
Казахстан	Астана
Кыргызстан	Бишкек

B

Страна	Столица
Грузия	Тбилиси
Казахстан	Астана
Кыргызстан	Бишкек
Латвия	Рига
Литва	Вильнюс
Молдова	Кишинев
Россия	Москва
Таджикистан	Душанбе
Туркменистан	Ашхабад
Узбекистан	Ташкент

C

Страна	Столица
Грузия	Тбилиси
Казахстан	Астана
Кыргызстан	Бишкек
Латвия	Рига
Узбекистан	Ташкент
Украина	Киев
Эстония	Таллин
Молдова	Кишинев

Вариант 2

A

Название реки	Длина на территории Беларуси, км
Березина	613
Днепр	690
Вилия	276
Неман	459
Сож	493
Западная Двина	328
Щара	325
Друть	295
Ясельда	242
Птичь	421
Случь	197

B

Название реки	Длина на территории Беларуси, км
Березина	613
Днепр	690
Вилия	276
Неман	459
Сож	493
Припять	500

C

Название реки	Длина на территории Беларуси, км
Днепр	690
Сож	493
Припять	500
Птичь	421
Друть	295
Ясельда	242
Щара	325

Вариант 3

A

Космонавт	Старт первого полета
Гагарин Юрий	12.04.1961
Титов Герман	06.08.1961
Попович Павел	12.08.1962
Терешкова Валентина	16.06.1963
Леонов Алексей	18.03.1965
Климук Петр	18.12.1973
Гречко Георгий	10.01.1975
Манаров Муса	21.12.1987

B

Космонавт	Старт первого полета
Гагарин Юрий	12.04.1961
Титов Герман	06.08.1961
Климук Петр	18.12.1973
Гречко Георгий	10.01.1975
Савицкая Светлана	19.08.1982
Манаров Муса	21.12.1987
Аубакиров Токтар	02.10.1991

C

Космонавт	Старт первого полета
Гагарин Юрий	12.04.1961
Попович Павел	12.08.1962
Климук Петр	18.12.1973
Манаров Муса	21.12.1987
Аубакиров Токтар	02.10.1991

Вариант 4

A

Чемпион зимних Олимпийских игр (Беларусь)	Год
Гришин Алексей	2010
Цупер Алла	2014
Кушнир Антон	2014
Домрачева Дарья	2014
Домрачева Дарья	2018
Гуськова Анна	2018
Алимбекова Динара	2018
Кривко Ирина	2018
Скардино Надежда	2018

B

Чемпион зимних Олимпийских игр (Беларусь)	Год
Гришин Алексей	2010
Кушнир Антон	2014

C

Чемпион зимних Олимпийских игр (Беларусь)	Год
Цупер Алла	2014
Домрачева Дарья	2014
Домрачева Дарья	2018
Гуськова Анна	2018
Алимбекова Динара	2018
Кривко Ирина	2018
Скардино Надежда	2018

Вариант 5

A

Страна	Наивысшая точка
Армения	Гора Арагац (4095 м)
Азербайджан	Гора Базардюзю (4466 м)
Беларусь	Гора Дзержинская (345 м)
Грузия	Гора Шхара (5193,2 м)
Казахстан	Хан-Тенгри (7010 м)
Кыргызстан	Пик Победы (7439 м)
Россия	Гора Эльбрус (5642 м)
Таджикистан	Пик Исмоила Сомони (7495 м)
Туркменистан	Гора Айрыбаба (3139 м)
Узбекистан	Пик Хазрет-Султан (4643 м)
Украина	Гора Говерла (2061 м)
Эстония	Гора Суур-Мунамяги (318 м)

B

Страна	Наивысшая точка
Грузия	Гора Шхара (5193,2 м)
Казахстан	Хан-Тенгри (7010 м)
Кыргызстан	Пик Победы (7439 м)
Россия	Гора Эльбрус (5642 м)
Таджикистан	Пик Исмоила Сомони (7495 м)

C

Страна	Наивысшая точка
Армения	Гора Арагац (4095 м)
Азербайджан	Гора Базардюзю (4466 м)
Беларусь	Гора Дзержинская (345 м)
Латвия	Вершина Гайзинькалнс (311,6 м)
Литва	Холм Аукштояс (293,84 м)
Молдова	Гора Баланешты (428,2 м)
Россия	Гора Эльбрус (5642 м)
Таджикистан	Пик Исмоила Сомони (7495 м)
Туркменистан	Гора Айрыбаба (3139 м)
Узбекистан	Пик Хазрет-Султан (4643 м)
Украина	Гора Говерла (2061 м)
Эстония	Гора Суур-Мунамяги (318 м)

ПРИЛОЖЕНИЕ Б
ЗАДАНИЯ ДЛЯ ЛАБОРАТОРНОЙ РАБОТЫ №5

Таблица Б.1 – Определение варианта для выполнения заданий

Номер варианта задания	**Номер по списку в подгруппе**		
1	1	6	11
2	2	7	12
3	3	8	13
4	4	9	14
5	5	10	15
Задание	**Комментарий к заданию**		
Задание 1. Рассмотреть аномалии при работе с соответствующим отношением	Привести пример аномалии вставки в отношении, предложить способы его модификации по устранению этой аномалии	Привести пример аномалии удаления в отношении, предложить способы его модификации по устранению этой аномалии	Привести пример аномалии обновления в отношении, предложить способы его модификации по устранению этой аномалии
Задание 2. На каждом этапе нормализации определить первичные ключи			
Задание 3. Нормализовать отношение до 3НФ:	Пояснить, каким образом производилась нормализация к 1НФ, 2НФ, 3НФ		
Задание 3.1. Нормализация до 1НФ	Указать многозначные и составные поля. Пояснить, что происходит с ними при нормализации		
Задание 3.2. Нормализация до 2НФ	Указать существующие частичные и полные функциональные зависимости		
Задание 3.3. Нормализация до 3НФ	Указать существующие транзитивные функциональные зависимости		
Задание 4. Проверить условия НФБК			

Варианты выполнения заданий[*]

Вариант 1

Таблица Б.2 – Студенты

ФИО преподавателя	Информация о преподавателе (должность, стаж работы, кафедра)	Дисциплина	Группа	ФИО старосты	Контакты старосты (номер телефона, e-mail)
Маленя И. С.	Ассистент, 4, философии	Философия	882513	Иванов И. И.	+555(71)2222222, 882513@mail.com
		Логика	962145	Петров П. Л.	962145@mail.com
Белкин А. М.	Доцент, 8, физики	Физика	202015	Сидорова Н. Г.	+872(54)42020152
		Физика	202015	Никулич П. Р.	+375(99)2222222, 202015@gmail.com
Зайцевич С. С.	Ассистент, 10, информатики	Базы данных	215636	Иванов И. М.	8(071)2222223, 215636@mail.by

Вариант 2

Таблица Б.3 – Сессия

ФИО студента	Информация о студенте (группа, факультет, год поступления)	Академические задолженности по предметам	ФИО преподавателя, предмет, дата пересдачи	ФИО старосты	Контакты старосты (номер телефона, e-mail)
Булков А. П.	202015, ФКТ, 2020	Философия, ООП	Захаров П. К., философия, 12.02.2022; Крупеня А. Л., ООП, 18.02.2022	Сидорова Н. Г.	+872(54)42020152
Зимцевич С. С.	202015, ФКТ, 2018	ООП	Крупеня А. Л., ООП, 18.02.2022	Иванов И. М.	8(071)2222223, 215636@mail.by
Медведев Е. В.	708015, МТФ, 2017	Нет	Нет	Обухопич Н. В.	+742(59)2222223, 708015@mail.by

[*]Информация в таблицах не относится к действительности, все совпадения случайны.

Вариант 3

Таблица Б.4 – Группы и кураторы

ФИО студента	Информация о студенте (группа, факультет, год поступления)	Оценки по всем предметам за сессию	Средний балл за сессию	ФИО куратора	Кафедра, рабочий телефон	Стаж работы в вузе
Новиков А. М.	202015, ФКТ, 2020	8, 6, 2, 3	4,75	Сидорова Н. Г.	ПОИТ, 223015	15
Купрович С. М.	202015, ФКТ, 2018	4, 4, 4, 2	3,5	Иванов И. М.	ВМСиС, 223018	12
Муштич Е. В.	708015, МТФ, 2017	5, 6, 7, 8	6,5	Обухопич Н. В.	ПДиУ, 223014	7

Примечание – Средний балл за сессию для конкретного студента рассчитывается как среднее арифметическое его оценок по всем предметам за сессию.

Вариант 4

Таблица Б.5 – Кафедры

Факультет	ФИО декана	Кафедры	ФИО зав. кафедрой	Количество преподавателей на кафедре	Количество остепененных преподавателей	Доля остепененных преподавателей на кафедре
ФКП	Никропов А. Н.	КИиКП	Иванов И. Д.	22	10	45 %
		ИТ	Петрович С. М.	21	9	43 %
		БД	Сидоренок П. Р.	15	5	33 %
ФКСиС	Зарембо Л. П.	ПИиКП	Казаков П. Д.	18	9	50 %
		ИТвЛ	Унсович Р. Т.	24	12	50 %
		ОИТ	Шпунчик П. Л.	17	7	41 %

Примечание – Доля остепененных преподавателей на кафедре – это отношение преподавателей, имеющих ученую степень, к их общему количеству. Выражается в процентах.

Вариант 5

Таблица Б.6 – Товары

Наименование товара	Поставщики	Количество товара, шт.	Себестоимость, ден. ед.	Стоимость поставки, ден. ед.	Склад хранения (адрес, тип склада), ФИО зав. складом
Товар 1	ООО «Морозрыба»	500	30	15 000	Орловская обл., тип А, Кукушкина П. Р.
	ООО «Океан»	650	27	17 550	Челябинская обл., тип А, Звенюк Р. Д.
	ЗАО «Рыбхоз №28»	320	31	9920	Минская обл., тип В, Хлобич В. К.
Товар 2	ООО «Сладости для радости»	321	10	3210	Владимирская обл., тип А, Казаков П. Д.
	ЗАО «Вкусно есть»	850	9	7650	Гомельская обл., тип В, Унсович Р. Т.
	ООО «Конфетка»	1000	12	12 000	Пинский р-н, тип А, Шпунчик П. Л.

Примечание – Стоимость поставки рассчитывается как произведение количества товара на себестоимость.

ПРИЛОЖЕНИЕ В
ВАРИАНТЫ ЗАДАНИЙ (ПРЕДМЕТНЫЕ ОБЛАСТИ) ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ №6 [6]

1. Автозаправка.
2. Автосалон.
3. Автостоянка.
4. Автошкола.
5. Аэропорт.
6. Банк.
7. Больница.
8. Военкомат.
9. Гостиница.
10. Грузоперевозки.
11. Детский сад.
12. Автовокзал.
13. Завод.
14. Зоопарк.
15. Кафе (бар).
16. Кинотеатр.
17. Компьютерная сеть.
18. Магазин продуктов.
19. Налоговая инспекция.
20. Общежитие.
21. Организация работы оператора связи.
22. Организация концертов.
23. Поликлиника.
24. Прокат велосипедов.
25. Ресторан.
26. СТО.
27. Столовая.
28. Киностудия.
29. Туристическое агентство.
30. Школа.

ПРИЛОЖЕНИЕ Г
ЭЛЕМЕНТЫ ОКНА MS SQL SERVER MANAGEMENT STUDIO

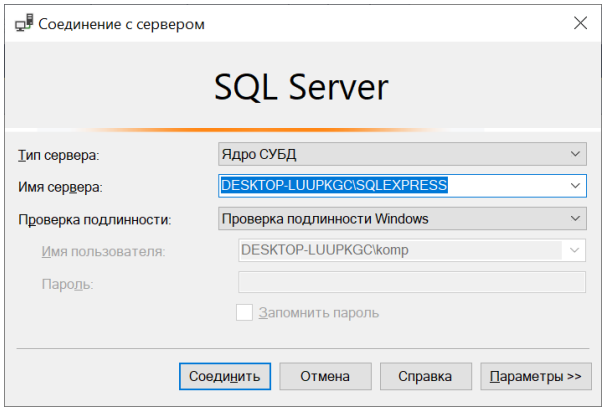
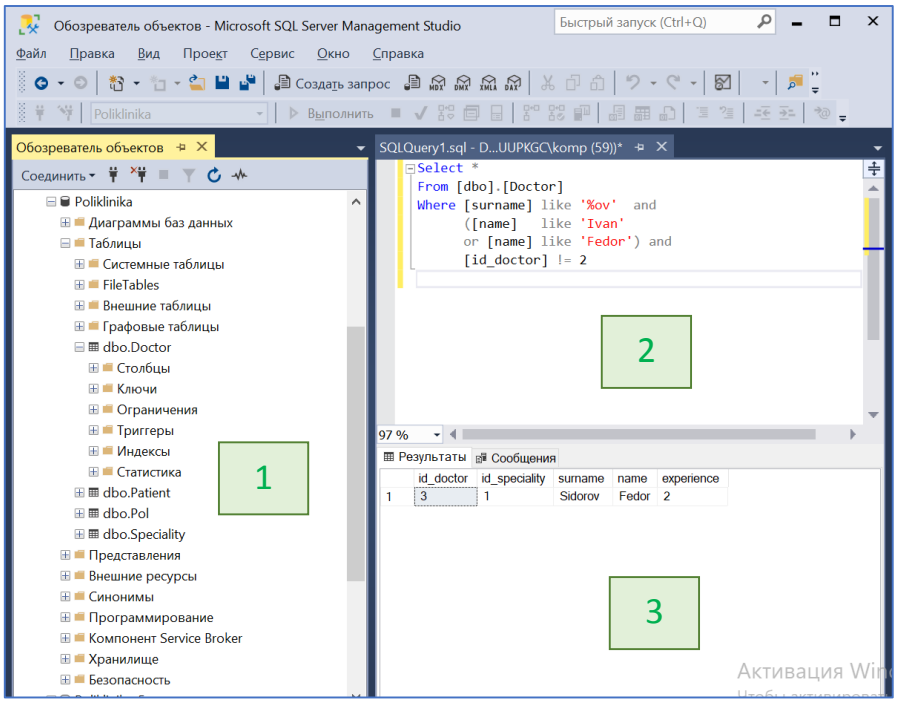


Рисунок Г.1 – Окно MS SQL Server соединения с сервером



1 – обозреватель объектов, содержащий все создаваемые и системные объекты SQL Server: базы данных, таблицы, их свойства, представления, хранимые процедуры, функции и др.;
2 – окно запросов, в котором записывается скрипт запроса и комментарии к нему, новое окно создается комбинацией клавиш Ctrl + N, скрипт запускается на выполнение клавишей F5; 3 – окно результатов выполнения запроса, в котором отображается следующая информация: результат выполнения запроса (созданного с помощью Select), информация об успешно созданном объекте базы данных, информация об ошибках и т. п.

Рисунок Г.2 – Окно обозревателя объектов MS SQL Server Management Studio

ПРИЛОЖЕНИЕ Д
ЗАДАНИЯ ДЛЯ ЛАБОРАТОРНОЙ РАБОТЫ №10

Задание 1. Агрегатные функции

Измените структуру любой из таблиц вашей БД, добавив числовой столбец «Примечание» и заполнив его произвольными различными дробными значениями в диапазоне от 50 до 300. Создайте запросы и сохраните их в виде представлений, в которых:

1) выведите таблицу, содержащую MIN, MAX, AVG, SUM значений столбца «Примечание». Значения выведите с двумя знаками после запятой;

2) подсчитайте количество строк, в которых значение данных столбца «Примечание» не превышает значение MIN(Примечание)+50;

3) выведите строки с данными из таблицы, для которых значение первичного ключа не превышает значения AVG(Примечание)/100.

Задание 2. Подзапросы и соединения таблиц

Для запросов выберите две такие таблицы, которые связаны между собой еще через две другие (есть цепочка из четырех таблиц), одна из которых содержит столбец «Примечание» (созданный в задании 1). Составьте два запроса:

– с помощью оператора JOIN;

– с помощью подзапросов без использования JOIN.

Выведите данные из первой и последней таблиц этой цепочки, при условии, что значение в столбце «Примечание» не меньше AVG(Примечание). Укажите, какие данные из каких таблиц возможно вывести в каждом из создаваемых запросов.

ПРИЛОЖЕНИЕ Е
ЗАДАНИЯ ДЛЯ ЛАБОРАТОРНОЙ РАБОТЫ №11

Измените структуру любой из таблиц вашей базы данных, добавив числовое поле «Примечание» и заполнив его произвольными дробными различными значениями в диапазоне от 50 до 300. Выберите две такие таблицы, которые связаны между собой еще через две другие (есть цепочка из четырех таблиц), одна из которых содержит столбец «Примечание».

Процедуры

Используйте цепочку из четырех связанных таблиц. Создайте процедуры, которые по заданному пользователем диапазону для значений первичного ключа из первой в цепочке таблицы определят все соответствующие данные из последней таблицы в цепочке:

- процедура 1 – без входных параметров;
- процедура 2 – с входными параметрами;
- процедура 3 – с выходным параметром. Выведите сообщение о количестве строк в искомом результате.

Осуществите вызов каждой из процедур.

Триггеры

Создайте два триггера к любой из таблиц вашей базы данных для следующих операций:

1) вставка, изменение;
2) удаление.

Для каждого триггера продемонстрируйте выполнение (с помощью скриншотов), вывод сообщений об ошибке, поведение системы при отключении одного триггера.

Функции

Создайте функции, которые возвращают результаты, соответствующие запросам 1, 2, 3 лабораторной работы №10 (пункт «Агрегатные функции»).

Осуществите вызов каждой из них.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Оскерко, В. С. Технологии баз данных и знаний : учеб. пособие / В. С. Оскерко, З. В. Пунчик. – Минск : БГЭУ, 2015. – 215 с.
2. Куликов, С. С. Реляционные базы данных в примерах: практическое пособие для программистов и тестировщиков / С. С. Куликов. – Минск : Четыре четверти, 2020. – 424 с.
3. Документация по Microsoft SQL [Электронный ресурс]. – 2022. – Режим доступа : https://docs.microsoft.com/ru-ru/sql/?view=sql-server-ver15.
4. Советский энциклопедический словарь / Гл. ред. А. М. Прохоров. – 4-е изд. – М. : Сов. энциклопедия, 1988. – 1600 с.
5. Кузнецов, С. Д. Основы баз данных : учеб. пособие / С. Д. Кузнецов. – 2-е изд., испр. – М. : Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007. – 484 с.
6. Калабухов, Е. В. Электронный ресурс по учебной дисциплине «Базы данных» для специальности 1-40 02 01 «Вычислительные машины, системы и сети» [Электронный ресурс]. – 2022. – Режим доступа : https://erud.bsuir.by.
7. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт ; пер. с англ. – 8-е изд.– М. : Изд. дом «Вильямс», 2005. – 1328 с.
8. Математика в примерах и задачах : учеб. пособие / Л. И. Майсеня [и др.] ; под общ. ред. Л. И. Майсени. – Минск : Выш. шк., 2022. – 454 с.
9. Астровский, А. И. Высшая математика : учебник. В 2 ч. Ч. 1 / А. И. Астровский, М. П. Дымков. – Минск : БГЭУ, 2022. – 415 с.
10. Куликов, С. С. Электронный учебно-методический комплекс по учебной дисциплине «Базы данных» (часть 1) для специальности 1-40 02 01 «Программное обеспечение информационных технологий» [Электронный ресурс]. – 2022. – Режим доступа : https://erud.bsuir.by.

Св. план 2022, поз. 50

Учебное издание

Кунцевич Ольга Юрьевна

БАЗЫ ДАННЫХ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

ПОСОБИЕ

Редактор *Е. С. Юрец*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *О. И. Толкач*

Подписано в печать 08.06.2023. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 5,0. Уч.-изд. л. 5,0. Тираж 50 экз. Заказ 31.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, г. Минск