

МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

# БАЗЫ ДАННЫХ

*Методические рекомендации к лабораторным работам  
для студентов специальности 1-53 01 02  
«Автоматизированные системы обработки информации»  
очной и заочной форм обучения*

Часть 2



Могилев 2023

УДК 004.65  
ББК 32.973.26-0.18.2  
Б17

Рекомендовано к изданию  
учебно-методическим отделом  
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «29» ноября 2022 г., протокол № 5

Составители: канд. техн. наук, доц. К. В. Захарченков;  
канд. техн. наук, доц. Т. В. Мрочек

Рецензент канд. техн. наук, доц. В. М. Ковальчук

Методические рекомендации содержат описание пяти лабораторных работ, выполняемых во втором семестре изучения дисциплины «Базы данных». Рассматриваются основы работы с Microsoft SQL Server и языком Transact-SQL.

Учебно-методическое издание

БАЗЫ ДАННЫХ

Часть 2

Ответственный за выпуск	В. В. Кутузов
Корректор	И. В. Голубцова
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.  
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ №

Издатель и полиграфическое исполнение:  
Межгосударственное образовательное учреждение высшего образования  
«Белорусско-Российский университет».  
Свидетельство о государственной регистрации издателя,  
изготовителя, распространителя печатных изданий  
№ 1/156 от 07.03.2019.  
Пр-т Мира, 43, 212022, г. Могилев.

© Белорусско-Российский  
университет, 2023

## Содержание

7	Создание запросов с помощью конструкторов СУБД.....	4
8	Создание форм и отчетов с помощью конструкторов СУБД.....	7
9	Создание и изменение таблиц средствами SQL.....	12
10	Создание связей между таблицами средствами SQL.....	17
11	Добавление, изменение и удаление данных в таблицах средствами SQL.....	19
	Список литературы.....	29

## Часть 2

### 7 Создание запросов с помощью конструкторов СУБД

**Цель:** приобрести навыки работы в СУБД Access по построению запросов.

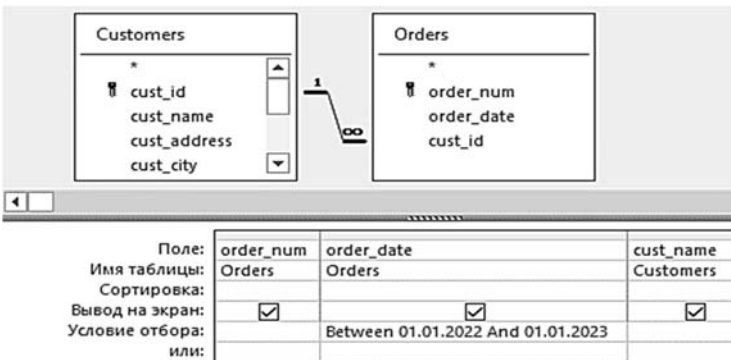
#### *Теоретические положения*

Запросы создаются для выборки данных из одной или нескольких связанных таблиц по заданным условиям, для проведения вычислений и статистической обработки данных [8, с. 54–74]. С помощью запроса можно обновить, удалить, добавить данные в таблицу, создать новые таблицы (таблица 7.1).

При выполнении запроса на выборку Access извлекает записи из таблиц и формирует результирующий набор данных – динамический (или виртуальный) набор записей, не хранящийся в базе данных, т. е. прекращающий свое существование после закрытия запроса. Сохраняется только структура запроса – перечень таблиц, список полей, порядок сортировки, тип запроса и т. д.

Запросы в СУБД Access можно создавать с помощью: режима мастера; режима конструктора, являющегося графическим инструментом языка QBE (Query-by-Example – язык запросов по образцу); языка SQL (используется диалект Jet SQL).

Таблица 7.1 – Виды запросов в Access

Наименование	Назначение	Способ реализации																								
1	2	3																								
Запрос на выборку	<p>Позволяет отобразить записи из одной или нескольких таблиц по указанным полям, сгруппировать записи для вычисления сумм, средних значений и т. д.</p> <p>Пример запроса на выборку:</p> <p>Запрос должен вывести все записи о заказах, сделанных за указанный в условии отбора период</p>	<p>Вкладка «Создание» → Кнопка «Конструктор запросов». Для шаблонов в поиске используются следующие символы: ? или _ заменяет любой текстовый символ; # – любую одиночную цифру (0–9); * или % – любое количество символов; [a–л] – символы в заданном интервале; [!m–ю] – символы вне заданного интервала. Например, условие Like "M*" выбирает записи со значениями поля, которые начинаются на букву М</p>  <table border="1" data-bbox="758 1904 1428 2038"> <tr> <td>Поле:</td> <td>order_num</td> <td>order_date</td> <td>cust_name</td> </tr> <tr> <td>Имя таблицы:</td> <td>Orders</td> <td>Orders</td> <td>Customers</td> </tr> <tr> <td>Сортировка:</td> <td></td> <td></td> <td></td> </tr> <tr> <td>Вывод на экран:</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Условие отбора:</td> <td colspan="3">Between 01.01.2022 And 01.01.2023</td> </tr> <tr> <td>или:</td> <td></td> <td></td> <td></td> </tr> </table>	Поле:	order_num	order_date	cust_name	Имя таблицы:	Orders	Orders	Customers	Сортировка:				Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Условие отбора:	Between 01.01.2022 And 01.01.2023			или:			
Поле:	order_num	order_date	cust_name																							
Имя таблицы:	Orders	Orders	Customers																							
Сортировка:																										
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>																							
Условие отбора:	Between 01.01.2022 And 01.01.2023																									
или:																										

Продолжение таблицы 7.1

1	2	3																																												
Запрос на выборку с параметром	<p>При выполнении выводит приглашение для ввода данных</p> <table border="1" data-bbox="432 454 1353 663"> <tr> <td>Поле:</td> <td>order_num</td> <td>order_date</td> <td>cust_name</td> </tr> <tr> <td>Имя таблицы:</td> <td>Orders</td> <td>Orders</td> <td>Customers</td> </tr> <tr> <td>Сортировка:</td> <td></td> <td></td> <td></td> </tr> <tr> <td>Вывод на экран:</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Условие отбора:</td> <td></td> <td>[Введите дату заказа]</td> <td></td> </tr> <tr> <td>или:</td> <td></td> <td></td> <td></td> </tr> </table>	Поле:	order_num	order_date	cust_name	Имя таблицы:	Orders	Orders	Customers	Сортировка:				Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Условие отбора:		[Введите дату заказа]		или:				<p>В строку «Условие отбора» вводят имя параметра в квадратных скобках, отличающееся от имени поля. В квадратных скобках нельзя использовать символ «точка»</p>																				
Поле:	order_num	order_date	cust_name																																											
Имя таблицы:	Orders	Orders	Customers																																											
Сортировка:																																														
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>																																											
Условие отбора:		[Введите дату заказа]																																												
или:																																														
Перекрестный запрос	<p>Представляет результаты в виде сводной таблицы с группировкой по строкам и столбцам и вычислениями по заданной функции (Sum, Min, Max, Avg, Count и т. д.). Значения группируются по двум наборам данных, один из которых расположен в левом столбце таблицы, а второй – в верхней строке</p> <p>Пример: запрос на вывод количества заказов, сделанных покупателями из различных городов на различные даты</p> <table border="1" data-bbox="424 1469 1366 1682"> <tr> <td>Поле:</td> <td>cust_city</td> <td>order_date</td> <td>order_num</td> </tr> <tr> <td>Имя таблицы:</td> <td>Customers</td> <td>Orders</td> <td>Orders</td> </tr> <tr> <td>Групповая операция:</td> <td>Группировка</td> <td>Группировка</td> <td>Count</td> </tr> <tr> <td>Перекрестная таблица:</td> <td>Заголовки строк</td> <td>Заголовки столбцов</td> <td>Значение</td> </tr> <tr> <td>Сортировка:</td> <td></td> <td></td> <td></td> </tr> <tr> <td>Условие отбора:</td> <td></td> <td></td> <td></td> </tr> <tr> <td>или:</td> <td></td> <td></td> <td></td> </tr> </table> <p>Результат:</p> <table border="1" data-bbox="485 1715 1294 1877"> <tr> <td>cust_city</td> <td>12_01_2021</td> <td>03_02_2021</td> <td>01_05_2021</td> </tr> <tr> <td>Borisov</td> <td>1</td> <td></td> <td></td> </tr> <tr> <td>Gomel</td> <td></td> <td>1</td> <td></td> </tr> <tr> <td>Mogilev</td> <td></td> <td>2</td> <td>1</td> </tr> </table>	Поле:	cust_city	order_date	order_num	Имя таблицы:	Customers	Orders	Orders	Групповая операция:	Группировка	Группировка	Count	Перекрестная таблица:	Заголовки строк	Заголовки столбцов	Значение	Сортировка:				Условие отбора:				или:				cust_city	12_01_2021	03_02_2021	01_05_2021	Borisov	1			Gomel		1		Mogilev		2	1	<p>Вкладка «Создание» → Кнопка «Конструктор запросов» → Запрос на выборку → Добавить нужные таблицы или запросы и задать условия отбора → Изменить тип запроса на «Перекрестный». Для полей, значения которых группируются по строкам, в строке «Перекрестная таблица» выбрать «Заголовки строк». Для поля, значения которого представлены в запросе как заголовки столбцов, в строке «Перекрестная таблица» выбрать «Заголовки столбцов». Для поля, по которому производятся вычисления, в строке «Перекрестная таблица» выбрать «Значения», а в строке «Групповая операция» – нужную функцию. Для полей, содержащих условие, но без группировки, в строке «Групповая операция» выбирается «Условие», а строка «Перекрестная таблица» оставляется пустой</p>
Поле:	cust_city	order_date	order_num																																											
Имя таблицы:	Customers	Orders	Orders																																											
Групповая операция:	Группировка	Группировка	Count																																											
Перекрестная таблица:	Заголовки строк	Заголовки столбцов	Значение																																											
Сортировка:																																														
Условие отбора:																																														
или:																																														
cust_city	12_01_2021	03_02_2021	01_05_2021																																											
Borisov	1																																													
Gomel		1																																												
Mogilev		2	1																																											
Запрос записи без подчиненных	Предназначен для поиска записей основной таблицы, у которых нет связанных записей	Вкладка «Создание» → Кнопка «Мастер запросов». Указываются анализируемая (главная) таблица, подчиненная таблица и поля связи																																												

Продолжение таблицы 7.1

1	2	3																													
<p>Запрос на выборку с групповыми операциями</p>	<p>Позволяет выделить группы записей с одинаковыми значениями в указанных полях группировки и выполнить статистические вычисления по заданной функции (Sum, Min, Max, Avg, Count (подсчет количества непустых значений поля в группе)). Результат содержит по одной записи для каждой группы</p> <p>Пример: вывод количества заказов по различным городам</p> <table border="1" data-bbox="502 719 1286 981"> <tr> <td>Поле:</td> <td>cust_city</td> <td>order_num</td> </tr> <tr> <td>Имя таблицы:</td> <td>Customers</td> <td>Orders</td> </tr> <tr> <td>Групповая операция:</td> <td>Группировка</td> <td>Count</td> </tr> <tr> <td>Сортировка:</td> <td></td> <td></td> </tr> <tr> <td>Вывод на экран:</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Условие отбора:</td> <td></td> <td></td> </tr> <tr> <td>или:</td> <td></td> <td></td> </tr> </table> <p>Результат:</p> <table border="1" data-bbox="625 1014 1155 1176"> <thead> <tr> <th>cust_city</th> <th>Count-order_num</th> </tr> </thead> <tbody> <tr> <td>Borisov</td> <td>1</td> </tr> <tr> <td>Gomel</td> <td>1</td> </tr> <tr> <td>Mogilev</td> <td>3</td> </tr> </tbody> </table>	Поле:	cust_city	order_num	Имя таблицы:	Customers	Orders	Групповая операция:	Группировка	Count	Сортировка:			Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Условие отбора:			или:			cust_city	Count-order_num	Borisov	1	Gomel	1	Mogilev	3	<p>В бланк запроса включают поля, по которым производится группировка, и поля, для которых выполняются групповые функции. При нажатии кнопки «Итоги» на вкладке «Конструктор» в бланке появится строка «Групповая операция», в которой для вычисляемых полей вместо «Группировка» указывают функцию</p>
Поле:	cust_city	order_num																													
Имя таблицы:	Customers	Orders																													
Групповая операция:	Группировка	Count																													
Сортировка:																															
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>																													
Условие отбора:																															
или:																															
cust_city	Count-order_num																														
Borisov	1																														
Gomel	1																														
Mogilev	3																														
<p>Запрос на обновление</p>	<p>Используется для обновления значений указанных полей таблицы новыми значениями</p> <p>Пример: обновление наименования города Борисова на Брест</p> <table border="1" data-bbox="643 1480 1190 1693"> <tr> <td>Поле:</td> <td>cust_city</td> </tr> <tr> <td>Имя таблицы:</td> <td>Customers</td> </tr> <tr> <td>Обновление:</td> <td>*Brest*</td> </tr> <tr> <td>Условие отбора:</td> <td>*Borisov*</td> </tr> <tr> <td>или:</td> <td></td> </tr> </table>	Поле:	cust_city	Имя таблицы:	Customers	Обновление:	*Brest*	Условие отбора:	*Borisov*	или:		<p>Вкладка «Создание» → Кнопка «Конструктор запросов» → Запрос на выборку → Добавить обновляемую таблицу → Изменить тип запроса на «Обновление». Для обновляемого поля в строке «Обновление» вводится выражение, вычисляющее значение</p>																			
Поле:	cust_city																														
Имя таблицы:	Customers																														
Обновление:	*Brest*																														
Условие отбора:	*Borisov*																														
или:																															
<p>Запрос на удаление</p>	<p>Предназначен для удаления записей, удовлетворяющих заданному условию</p>	<p>Вкладка «Создание» → Кнопка «Конструктор запросов» → Запрос на выборку → Добавить требуемую таблицу → Изменить тип запроса на «Удаление». В первом столбце в первом поле бланка ставится «ИмяПоля.*» (т. е. «все поля»), после чего в строке «Удаление» будет выведен текст «Из». В следующих столбцах выбирают поля, для которых задают условия отбора</p>																													

Окончание таблицы 7.1

1	2	3
Запрос на добавление	Предназначен для вставки записей из одной или нескольких таблиц в одну целевую таблицу	Вкладка «Создание» → Кнопка «Конструктор запросов» → Запрос на добавление
Запрос на создание таблицы	Служит для создания новой таблицы на основе записей существующих таблиц	Строится как запрос на выборку, а затем тип запроса меняется на «Создание таблицы» и в появившемся диалоговом окне задается имя новой таблицы

### **Задание**

Для спроектированной базы данных необходимо разработать 15 запросов, представляющих изученные виды запросов (с учетом запросов в техническом задании, разработанном в лабораторной работе № 1).

**Содержание отчета:** тема и цель работы; SQL-код запросов и скриншоты результатов выполнения запросов.

### **Контрольные вопросы**

1 Что такое запрос? Какие способы создания запросов существуют?

2 Как при создании запроса установить условия отбора? Как осуществить поиск данных в диапазоне значений? Как используются выражения в запросе? Как сортируются данные в запросе? Как установить параметры в запросе? Как суммировать данные в запросе?

3 Каково назначение запросов с групповыми операциями и перекрестных запросов? Какие функции используются в данных запросах?

4 Логические операторы и их назначение. Назначение операторов BETWEEN, IN, LIKE.

5 Встроенные функции даты и времени. Встроенные функции и операторы для работы с текстом. Встроенные функции преобразования типов данных.

6 Как создаются запросы на обновление, добавление, удаление?

## **8 Создание форм и отчетов с помощью конструкторов СУБД**

**Цель:** приобрести навыки работы в СУБД MS Access по созданию форм и отчетов.

### **Теоретические положения**

**Формы** являются основным средством организации интерфейса пользователя в приложениях Access [8].

Существуют следующие способы создания формы [8, с. 75–89]:

– с помощью инструмента «Форма», который позволяет создать форму на

основе указываемой готовой таблицы (или запроса);

- с помощью конструктора форм, который позволяет создавать и редактировать формы любой степени сложности;

- с помощью мастера форм (создание простых настраиваемых форм);

- с помощью инструмента «Разделенная форма», что позволяет одновременно отображать данные в двух представлениях – в режиме таблицы, отображаемой в верхней части формы, и в режиме формы для ввода данных в запись, выделенную в таблице;

- с помощью инструмента «Несколько элементов», что позволяет создать форму, в которой отображается несколько записей;


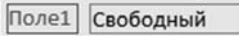

- с помощью инструмента «Пустая форма».

Однотабличная форма создается с помощью инструментов «Форма», «Разделенная форма», «Несколько элементов», размещенных на вкладке ленты «Создание» (Create) в группе «Формы» (Forms).

Многотабличная форма создаётся на основе многотабличного запроса для работы с данными нескольких взаимосвязанных таблиц.

Одиночная многотабличная форма создаётся на основе многотабличного запроса с помощью инструментов «Форма», «Несколько элементов» или с помощью мастера форм (мастер позволит определить набор взаимосвязанных таблиц и создаст источник записей формы – многотабличный запрос).

Составная многотабличная форма создаётся с помощью инструментов «Форма», «Несколько элементов». Источником данных главной формы является таблица или запрос. Одна или несколько подчиненных форм строятся на основе таблицы или запроса, подчиненного источнику записей главной формы.

*Создание вычисляемых полей в форме.* Следует открыть форму в режиме конструктора и найти панель элементов управления , далее выбрать элемент «Поле» и перенести его на свободное место формы. На форме появится элемент . Переименовать подпись *Поле1*, например, в *Итого*. Щелкнув правой кнопкой мыши по окошку с надписью *Свободный*, выбрать «Свойства». В открывшемся диалоговом окне во вкладке «Данные» в строке «Данные» открыть с помощью кнопки  «Построитель выражений». В окне «Построитель выражений» можно записать математические выражения, выбирая нужные таблицы и поля из этих таблиц, участвующие в вычислениях.

Вычисление итоговых значений в форме производится в примечании формы с помощью встроенных статистических функций, записываемых в вычисляемых выражениях в элементе управления «Поле».

*Создание кнопок на форме.* Если на форме поместить кнопку, то в появившемся окне «Создание кнопок» в разделе «Категории» можно связать кнопку с каким-либо действием – открытием другой формы, выполнением запроса, печатью таблицы, выходом из приложения и т. д.

**Кнопочная форма** – это форма, содержащая набор кнопок, направляющих пользователя к другим формам (обычно при щелчке мышью кнопки формы), т. е. своего рода главное меню БД.

Для автоматического создания кнопочной формы следует выбрать на ленте «Работа с базами данных» → «Диспетчер кнопочных форм» (Database Tools →



Switchboard Manager). Если диспетчер кнопочных форм отсутствует, то его надо включить. Для этого следует выбрать пункт меню «Файл», выбрать раздел «Параметры» и в нем – «Панель быстрого доступа». Затем на вкладке «Настроить панель быстрого доступа» выбрать из раскрывающегося списка строку «Вкладка «Работа с базами данных» и в списке команд выделить «Диспетчер кнопочных форм», а дальше нажать на кнопку «Добавить». При первом нажатии этой кнопки Access сообщит, что не может найти кнопочную форму и предложит ее создать. Диспетчер кнопочных форм выводит на экран список страниц. Каждая страница – отдельная часть меню кнопочной формы.

Окно «Изменение страницы кнопочной формы» позволяет создавать команды меню, удалять те, которые больше не нужны, изменять порядок их следования (этот порядок определяет порядок команд на кнопочной форме). Для создания команды меню необходимо указать текст, появляющийся на форме, и команду, которую должна выполнить Access при нажатии кнопки. На каждой странице кнопочной формы можно поместить только восемь команд меню. Если нужно больше, следует добавить дополнительные страницы в меню.

*Автоматический запуск главной кнопочной формы при открытии приложения.* Предполагается, что форма, которая должна отображаться первой, уже создана. Следует открыть вкладку «Файл» и выбрать «Параметры». В открывшемся окне в левой колонке в группе «Текущая база данных» в списке «Форма просмотра» выбрать форму, отображаемую при запуске БД, и нажать ОК. Чтобы отобразилась начальная форма, нужно закрыть БД и открыть ее повторно.

**Макросы** – это небольшие программы на языке макрокоманд СУБД Access, состоящие из последовательности определенных команд (одной или нескольких макрокоманд) [8, с. 96]. Макросы являются простейшими средствами автоматизации действий над объектами Access. Если нажать правой кнопкой мыши по полю кнопки и выбрать **Обработка событий** → **Макросы**, откроется окно конструктора макросов, в котором надо выбрать макрокоманду из выпадающего списка (всего имеется около 50 различных макрокоманд). Все созданные макросы будут отображаться во вкладке **Макросы**.

**Отчеты** используются для отображения информации, содержащейся в таблицах, в отформатированном виде, который легко читается как на экране компьютера, так и на бумаге [8, с. 90–95].

Отчет можно создать в режиме конструктора, в режиме мастера и в режиме макета. В ходе создания отчета простыми средствами перетаскивания в отчет нужных полей из таблиц базы данных конструируется запрос – источник записей отчета. Использование свойств WYSIWYG позволяет сразу увидеть, как именно будут выглядеть данные на странице отчета, и усовершенствовать макет.

Разработка и форматирование отчета аналогичны форме.

Отчет можно отобразить в трех режимах: в режиме конструктора (что позволяет изменить внешний вид и макет отчета); в режиме просмотра образца (можно просмотреть все элементы готового отчета, но в сокращенном виде); в режиме предварительного просмотра.

Инструменты отчета расположены на вкладке «Создание» в группе «Отчеты».

Инструмент «Отчет» позволяет создать простой табличный отчет, содержащий все поля из источника записей, который выбран в области навигации.

Инструмент «Конструктор отчетов» открывает в режиме конструктора пустой отчет, в который можно добавить необходимые поля и элементы управления.

Инструмент «Пустой отчет» позволяет открыть пустой отчет в режиме макета и отобразить область задач «Список полей», из которой можно добавить поля в отчет.

Инструмент «Мастер отчетов» служит для вызова пошагового мастера, с помощью которого можно задать поля, уровни группировки и сортировки и параметры макета.

Инструмент «Наклейки» вызывает мастер, в котором можно выбрать стандартный или настраиваемый размер подписей, набор отображаемых полей и порядок их сортировки.

Основные элементы отчета (в режиме конструктора) показаны на рисунке 8.1.

Заголовок отчета													
Таблица1											=Дата()		
Верхний колонтитул													
Код													
Область данных													
Код													
Нижний колонтитул													
												"Страница " & [Page] & " из " & [Pages]	
Примечание отчета													
=Count(*)													

Рисунок 8.1 – Основные элементы отчета

Заголовок отчета обычно включает информацию, помещаемую на титульном листе, например, логотип, название отчета, дату.

Верхний колонтитул отображается вверху каждой страницы и используется в случае, когда нужно, чтобы название отчета и другая общая информация повторялись на каждой странице.

Область данных отображает элементы управления и записи из источника данных, составляющие основное содержание отчета.

Нижний колонтитул применяется для нумерации страниц и отображения другой постраничной информации внизу каждой страницы;

Примечание отчета служит для отображения итогов и другой сводной информации по всему отчету один раз в конце отчета. Если в примечании отчета поместить вычисляемый элемент управления, использующий, например, статистическую функцию Sum, сумма рассчитывается для всего отчета.

*Группировка данных* в отчетах – средство для придания смысла большим объемам данных за счет упорядочивания их в группах меньшего размера. Затем можно выполнять вычисления в каждой отдельной группе.

Существуют три способа применения группировки для анализа информации в отчете.

1 Применение группировки в запросе. В этом случае в ваш отчет не включаются подробности. Он только отображает вычисленные суммы, средние значения, максимумы или минимумы. Здесь создается сводный отчет с группировкой и затем используется для формирования отчета.

2 Применение группировки в отчете. В этом случае можно разделить информацию большого объема на подгруппы. При этом можно видеть все данные и применять промежуточные итоги и другие вычисления. Можно также добавить несколько уровней группировки для выявления глубинных тенденций.

3 Применение подчиненных отчетов. Этот метод создает тот же эффект, что и группировка в отчете. Единственное отличие – формирование отчета из двух отдельных частей.

Для создания групп выполните следующие действия.

1 Перейдите в **Режим макета** или **Конструктор**.

2 Выберите поле, которое хотите использовать для сортировки. Обычно следует сортировать таблицу по тому полю, которое планируется применять для группировки.

3 Для сортировки данных щелкните правой кнопкой мыши поле, по которому собираетесь сортировать, и выберите команду сортировки (например, **Сортировка от А до Я (Sort A to Z)** или **Сортировка от минимального к максимальному (Sort Smallest to Largest)**).

Точное название в меню команды сортировки зависит от типа данных, хранящихся в поле.

4 Щелкните правой кнопкой мыши поле, которое хотите использовать для группировки, и выберите команду **Группировка (Group On)**. Программа Access отсортирует ваши результаты по этому полю и затем сгруппирует их.

Когда группировка задана, появляются дополнительные возможности:

- можно вставить дополнительную сортировку в пределах каждой подгруппы;
- можно выполнить сводные вычисления для каждой группы;
- можно расставить разрывы страниц в начале каждой новой группы.

Любой из этих вариантов легче всего добавить с помощью панели **Группировка, сортировка и итоги (Group, Sort, and Total)**. Для ее отображения в **Конструкторе** выберите на ленте **Инструменты конструктора отчетов | Конструктор** → **Группировка и итоги** → **Группировка и сортировка (Report Design Tools | Design → Grouping & Totals → Group & Sort)** или в **Режиме макета - Работа с макетами отчетов | Формат** → **Группировка и итоги** → **Группировка и сортировка (Report Layout Tools | Formatting → Grouping & Totals → Group & Sort)**.

### Задание

Для созданной базы данных необходимо для всех таблиц разработать формы, оформить главную кнопочную форму.

Необходимо реализовать все отчеты, указанные в подразделе 4.2 технического задания (лабораторная работа № 1) и в функциональной модели в лабораторной работе № 2.

**Содержание отчета:** тема и цель работы; скриншоты отчетов.

### ***Контрольные вопросы***

- 1 Какие способы создания форм существуют?
- 2 Что такое главная кнопочная форма и как организовать автоматический запуск главной кнопочной формы при открытии приложения?
- 3 Что такое макрос? Для чего он используется в базе данных? Каким образом можно выполнить макрос? Перечислите основные виды макрокоманд и их значение.
- 4 Для чего служат отчеты и из каких разделов они могут состоять?
- 5 Какие элементы форматирования используют для оформления отчетов?
- 6 Как в отчете задаются уровни группировки и сортировка?
- 7 С какой целью и в какие разделы отчета добавляются вычисляемые поля?

## **9 Создание и изменение таблиц средствами SQL**

**Цель:** получить навыки создания и изменения структуры таблиц средствами T-SQL.

### ***Теоретические положения***

Процесс создания таблицы начинается с проектирования ее будущей структуры. В процессе проектирования необходимо решить следующие вопросы:

- 1) для хранения каких данных предназначена создаваемая таблица;
- 2) каким образом будет обеспечиваться целостность данных в ней. При этом определяется, какие из ограничений целостности будут реализованы декларативно, т. е. непосредственно при создании таблицы с помощью ограничений на значения столбцов (CONSTRAINTS), а какие процедурно, т. е. с помощью хранимых процедур, триггеров и т. п.

SQL Server позволяет управлять значениями столбцов при помощи следующих механизмов.

- 1 Определение первичного ключа (PRIMARY KEY).
- 2 Определение внешнего ключа (FOREIGN KEY).
- 3 Создание уникальных столбцов (UNIQUE) в нескольких столбцах таблицы, помимо первичного ключа.
- 4 Наложение проверочных ограничений на значения столбцов (CHECK).
- 5 Определение значений по умолчанию (DEFAULT).
- 6 Определение для столбца возможности содержать неопределенные значения (NULL).

Для создания таблицы используется следующая инструкция T-SQL:

```
CREATE TABLE [ [<имя базы данных>.<имя схемы>.<имя таблицы>
( { <определение столбца>
  |, <AS определение вычисляемого столбца>
  |, <ограничение таблицы>}
)
```

Определение каждого столбца таблицы имеет следующий формат:

```
<определение столбца> ::=
<имя столбца> <тип данных>
[ NULL | NOT NULL ]
[ DEFAULT <выражение>
| IDENTITY [( <начальное значение>, <приращение>)] ]
[ ROWGUIDCOL ]
[ <ограничение столбца> ... ]
```

Подобным образом необходимо описать каждый столбец в таблице. Прежде всего следует определить имя столбца и тип хранимых в нем данных. При описании могут быть использованы следующие ключевые слова.

DEFAULT – определяет значение по умолчанию, используемое, если при вводе строки в операторе INSERT не было явно указано его значение.

IDENTITY – предписывает системе осуществлять заполнение столбца уникальными целочисленными значениями автоматически. При этом также можно указать начальное значение (seed) и приращение (increment). Такой столбец может быть в таблице только один и называется автоинкрементным или столбцом идентификаторов. Для автоинкрементного столбца нельзя задавать значение по умолчанию (ограничение DEFAULT). Такой столбец может иметь тип данных TINYINT, SMALLINT, INT, BIGINT, DECIMAL или NUMERIC. Если ему задается тип данных DECIMAL или NUMERIC, то количество дробных знаков должно быть указано нулевым.

ROWGUIDCOL – указывает, что столбец является столбцом идентификаторов GUID (используемым для хранения глобального идентификационного номера). Только один столбец типа UNIQUEIDENTIFIER в таблице может быть назначен в качестве столбца ROWGUIDCOL. Свойство ROWGUIDCOL не обеспечивает уникальности значений, хранимых в столбце. Кроме того, при указании данного свойства автоматического формирования значений для новых строк, вставляемых в таблицу, не выполняется. Для создания уникальных значений в каждом столбце следует использовать функцию NEWID().

Кроме того, для столбца можно определить ограничения на значения (на уровне столбца или на уровне таблицы) следующим образом:

```
<ограничение столбца> ::=
[ CONSTRAINT <имя_ограничения> ]
```

```

{
  { <PRIMARY KEY > | <UNIQUE> }
  [ CLUSTERED | NONCLUSTERED ]
  [ WITH FILLFACTOR = fillfactor ]
  | [ [FOREIGN KEY]
  REFERENCES [ ИмяСхемыРодительскойТаблицы .] РодительскаяТаблица
  [ (СтолбецРодительскойТаблицы) ]
  [ ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
  [ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
  | CHECK ( ЛогическоеВыражение )
}

```

После ключевого слова CONSTRAINT указывается имя ограничения на значение. Ограничения могут носить произвольные названия, но, как правило, применяются следующие префиксы:

- "PK\_" – для PRIMARY KEY;                      – "FK\_" – для FOREIGN KEY;
- "CK\_" – для CHECK;                                 – "UQ\_" – для UNIQUE.
- "DF\_" – для DEFAULT;

Имена ограничений задавать необязательно, при установке соответствующих атрибутов SQL Server автоматически определяет их имена. Но, зная имя ограничения, можно к нему обращаться, например, для его удаления.

После этого требуется определить тип ограничения.

PRIMARY KEY – определяет столбец как первичный ключ таблицы. В качестве альтернативы можно определить столбец как уникальный, воспользовавшись ключевым словом UNIQUE. При необходимости можно также указать, будет ли индекс, создаваемый для данного ограничения, кластерным (ключевое слово CLUSTERED) или некластерным (NONCLUSTERED). Однако кластерный индекс можно определить только для одного ограничения, поэтому требуется решить, с каким ограничением (первичный ключ или уникальный столбец) он будет использоваться. Если создается индекс, указывается также степень заполнения его страниц (ключевое слово WITH FILLFACTOR).

FOREIGN KEY – определяет столбец как внешний ключ таблицы. Одновременно, используя ключевое слово REFERENCES, необходимо указать имя родительской таблицы, с которой будет связана создаваемая подчиненная таблица.

Пример создания таблицы с ограничениями на значения столбцов:

```

CREATE TABLE Customers
(
  Id INT CONSTRAINT PK_Customer_Id PRIMARY KEY IDENTITY,
  age INT
  CONSTRAINT DF_Customer_Age DEFAULT 18
  CONSTRAINT CK_Customer_Age CHECK(age >0 AND age < 100)
)

```

Следует заметить, что можно определить вычисляемый столбец, который будет производным от других столбцов. При этом на хранение его значений не требуется физической памяти, поскольку они в любой момент могут быть получены для любой строки путем вычислений. Определение вычисляемого столбца имеет следующий формат: имя\_столбца AS выражение. Выражение может включать имена столбцов, функции и арифметические операции.

Более подробно вопросы создания таблиц описаны в [4, 5, 7, 9].

Для изменения структуры таблицы средствами T-SQL предусмотрена специальная команда:

```
ALTER TABLE table
{[ALTER COLUMN
column_name {new_data_type [(precision [, scale])]}
[NULL | NOT NULL]
| {ADD | DROP} ROWGUIDCOL}]
|ADD
{[<column_definition> ]
column_name AS computed_column_expression}[,...n]
[WITH CHECK | WITH NOCHECK] ADD
{<table_constraint>}[...n]
|DROP
[CONSTRAINT] constraint_name | COLUMN column }[...n]
{CHECK | NOCHECK} CONSTRAINT {ALL | constraint_name}[,...n]}
{ENABLE | DISABLE} TRIGGER {ALL | trigger_name}[....n]} }
```

Рассмотрим синтаксис данной команды. С помощью команды ALTER TABLE можно изменить определение уже существующих столбцов, удалить любой из них, а также добавить в таблицу новые столбцы.

*Изменение определения столбца.* Данная операция осуществляется с использованием ключевого слова ALTER COLUMN, после которого помещается имя изменяемого столбца (column\_name). Можно изменить тип данных столбца (new\_data\_type), размерность (precision) и точность (scale). Можно указать, разрешено ли столбцу содержать значения NULL. В этом случае обязательно нужно указать тип данных для столбца, даже если вы не хотите его изменять (просто укажите существующий тип данных). Если определяется для столбца свойство NOT NULL, нужно позаботиться о том, чтобы на момент изменения столбец не содержал ни одного значения NULL.

*Добавление в таблицу нового столбца.* Для определения нового столбца необходимо использовать ключевое слово ADD. За ним следует описание столбца, которое имеет такой же формат, как и при создании столбца с помощью команды CREATE TABLE. Здесь же можно наложить на таблицу новые ограничения на значения столбцов. Определив ключевое слово WITH CHECK, системе предписывается при добавлении новых ограничений на значения столбцов FOREIGN KEY или CHECK осуществлять проверку данных в таблице на соответствие этим ограничениям. По умолчанию данная проверка проводится для всех

вновь создаваемых ограничений. Когда выполнение подобной проверки не требуется, необходимо использовать ключевое слово `WITH NOCHECK`.

*Удаление столбцов из таблицы.* В случае необходимости можно удалить из таблицы некоторые столбцы. Для этого используется ключевое слово `DROP`. Можно удалить как конкретный столбец (ключевое слово `COLUMN`), так и определенное ограничение на значение столбца. Однако нельзя удалять следующие столбцы: столбцы, задействованные в индексе; столбцы, полученные в результате репликации; столбцы, для которых определены любые ограничения на значения; столбцы, для которых определены значения по умолчанию; столбцы, связанные с правилом.

*Управление ограничениями на значения столбцов.* Иногда бывает необходимо отключить ограничения на значения столбцов `FOREIGN KEY` или `CHECK`. Отключение конкретного ограничения (`NOCHECK CONSTRAINT`) означает, что при вводе новых строк данные не будут проверяться на соответствие этому ограничению. Когда снова потребуется сделать ограничение активным, используйте ключевое слово `CHECK CONSTRAINT`. При необходимости можно управлять всеми ограничениями сразу с помощью ключевого слова `ALL`.

*Управление триггерами.* Ключевое слово `DISABLE TRIGGER` отключает триггер. При этом в процессе изменения данных в таблице те действия, которые определены в триггере как реакция системы на эти изменения, не производятся, хотя триггер продолжает существовать. Чтобы активизировать триггер, необходимо использовать команду с ключевым словом `ENABLE TRIGGER`. Если требуется управлять сразу всеми триггерами, используется ключевое слово `ALL`.

Более подробно вопросы изменения таблиц описаны в [1–7, 9–14].

### **Задание**

Необходимо создать средствами T-SQL три резервные таблицы, обеспечивающие возможность сохранения копий строк при удалении данных из других таблиц разрабатываемой базы данных. При этом должны быть реализованы все изученные виды ограничений на значения столбцов.

Далее необходимо изменить структуру указанных трех таблиц так, чтобы в них можно было хранить информацию о времени удаления данных из исходных таблиц.

**Содержание отчета:** тема и цель работы; прокомментированный SQL-код выполнения задания.

### ***Контрольные вопросы***

- 1 Перечислить этапы проектирования структуры таблиц.
- 2 С помощью каких команд T-SQL можно создать, изменить или удалить таблицу [4, с. 381, 384–418]?
- 3 Каков синтаксис команды изменения структуры таблицы средствами T-SQL?
- 4 Какие задачи решаются с помощью команды `ALTER TABLE`?



5 Как изменить определение столбца таблицы?

6 Как добавить новый столбец или удалить столбец из таблицы?

7 Охарактеризовать ограничения, которые можно наложить на значения столбцов (PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, DEFAULT, NULL).

8 С помощью каких команд производится управление ограничениями на значения столбцов?

## 10 Создание связей между таблицами средствами SQL

**Цель:** изучить основы создания связей между таблицами средствами T-SQL.

### *Теоретические положения*

При создании связи между таблицами необходимо определить ее тип: идентифицирующая или неидентифицирующая (обязательная или необязательная), 1:1, 1:M, M:M.

Для реализации связи необходимо, чтобы одна из таблиц содержала ссылку (внешний ключ) на вторую. Внешний ключ – это столбец (или группа столбцов таблицы), содержащий значения, совпадающие со значениями первичного ключа в этой же или другой таблице.

Синтаксис предложения FOREIGN KEY следующий:

```
[CONSTRAINT c_name]
[[FOREIGN KEY] ({col_name1} ,...)]
REFERENCES table_name ({col_name2},...)
[ON DELETE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]
[ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]
```

Предложение FOREIGN KEY явно определяет все столбцы, входящие во внешний ключ. В предложении REFERENCES указывается имя таблицы, содержащей столбцы, создающие соответствующий первичный ключ. Количество столбцов и их тип данных в предложении FOREIGN KEY должны совпадать с количеством соответствующих столбцов и их типом данных в предложении REFERENCES (и, конечно же, они должны совпадать с количеством столбцов и типами данных в первичном ключе таблицы, на которую они ссылаются).

Таблица, содержащая внешний ключ, называется ссылающейся (или дочерней) таблицей, а таблица, содержащая соответствующий первичный ключ, – ссылаемой или родительской.

Создадим, например, две таблицы, связанные отношением 1:1. Столбец ссылки id\_a\_link таблицы TableB нужно сделать уникальным внешним ключом. Это гарантирует, что в таблице TableB может быть только одна запись, которая соответствует значению в столбце PRIMARY KEY в таблице TableA.

```
CREATE TABLE TableA (
id_a INT PRIMARY KEY IDENTITY(1,1),
name VARCHAR(255));
```

```
CREATE TABLE TableB (
id_b INT PRIMARY KEY IDENTITY(1,1),
name VARCHAR(255),
id_a link INT UNIQUE,
FOREIGN KEY (id_a_link) REFERENCES TableA (id_a));
```

### Задание

Необходимо установить связи между резервными таблицами, созданными в лабораторной работе № 9, соответствующие отношениям между основными таблицами, средствами T-SQL. Обоснование реализованных типов связей представить в виде таблицы 10.1.

Таблица 10.1 – Обоснование реализованных типов связей

Наименование типа созданной связи	SQL-определение столбцов, участвующих в связи со стороны родительской таблицы	SQL-определение столбцов, участвующих в связи со стороны дочерней таблицы	SQL-определение созданной связи

**Содержание отчета:** тема и цель работы; подробно прокомментированный SQL-код выполнения задания.

### Контрольные вопросы

- 1 Какие типы связей можно реализовать в MS SQL Server?
- 2 Для чего предназначены предложения FOREIGN KEY и REFERENCES?
- 3 Какие ограничения на значения столбцов можно накладывать?
- 4 Как создаются идентифицирующие и неидентифицирующие (обязательные и необязательные) связи?
- 5 Как создаются связи 1:1, 1:M, M:M?

## 11 Добавление, изменение и удаление данных в таблицах средствами SQL

**Цель:** научиться добавлять, изменять и удалять данные в таблицах с помощью команд T-SQL.

### *Теоретические положения*

#### *11.1 Добавление данных в таблицу средствами T-SQL*

Для вставки данных в таблицу средствами T-SQL используется команда **INSERT**, имеющая следующий синтаксис [4, 5, 7, 9]:

```
INSERT [INTO] { имя_таблицы | имя_представления }
{ [ (column1, ..., column_n) ]
{ VALUES ( { DEFAULT | NULL
| выражение }[,... n ] ) | временная_таблица | инструкция_выполнения } }
| DEFAULT VALUES
```

Ключевое слово **INSERT** и необязательное ключевое слово **INTO** вводят инструкцию. Аргумент «имя\_таблицы» задает целевую таблицу, в которую необходимо вставить данные. Можно в качестве цели вставки задать имя представления. Далее может указываться список столбцов, разделенных запятыми, который будет получать вставляемые данные. Вставляемые значения можно задать ключевыми словами **DEFAULT**, **NULL** или выражениями.

Если введено ключевое слово **DEFAULT** для столбца, который не имеет значения по умолчанию, то инструкция вставит значение **NULL** (разумеется, если в столбце разрешены пустые значения). В противном случае инструкция **INSERT** вызовет ошибку, и данные вставлены не будут [4].

В качестве альтернативы можно использовать инструкцию **SELECT** для создания временной таблицы, которая станет поставщиком данных для вставки.

Предложение **DEFAULT VALUES** использует значения таблицы по умолчанию для каждого столбца в новой строке. Инструкция **INSERT** может вставлять множество строк данных, если в качестве поставщика данных используется временная таблица или результаты инструкции выполнения.

Не обязательно перечислять столбцы в том порядке, в котором они выводятся в таблице. Главное, чтобы списки столбцов и значений соответствовали друг другу. Если в явном виде не указан список столбцов таблицы, то инструкция **INSERT** будет пытаться вставить значения в каждый столбец таблицы в порядке предоставления значений.

Если таблица имеет триггер **INSTEAD OF INSERT**, то вместо любой инструкции **INSERT**, пытающейся поместить строки в эту таблицу, будет выполняться код в триггере [4].

В таблице 11.1 приведены примеры **оператора INSERT** для вставки данных в таблицу, SQL-код определения которой

```
CREATE TABLE Book
(
  b_id_book int PRIMARY KEY IDENTITY (1, 1),
  b_title varchar (100) NULL,
  b_publisher varchar (50) NOT NULL DEFAULT 'ИНФРА-М'
)
```

Таблица 11.1 – Примеры инструкций INSERT

Задача	SQL-код решения
Добавление в таблицу строки с указанием имен столбцов	INSERT INTO Book (b_title, b_publisher) VALUES ('Война и мир', 'BHV');
Добавление в таблицу строки без указания имен столбцов	INSERT INTO Book VALUES ('Война и мир', 'BHV');
Добавление в таблицу строки с подстановкой значения, заданного ограничением DEFAULT	INSERT INTO Book VALUES ('Анна Каренина', DEFAULT);
Добавление в таблицу строки с указанием измененного порядка следования столбцов	INSERT INTO Book (b_publisher, b_title) VALUES ('BHV', 'Война и мир');
Добавление в таблицу строки в том случае, если все столбцы имеют значения по умолчанию (второй столбец в таблице Book тоже имеет значение по умолчанию – NULL)	INSERT INTO Book DEFAULT VALUES;
Добавление в таблицу нескольких строк	INSERT INTO Book VALUES ('Исповедь', DEFAULT), ('Воскресение', DEFAULT);
Добавление в таблицу строки с заданным значением в столбце идентификаторов	SET IDENTITY_INSERT Book ON; INSERT INTO Book (b_id_book, b_title, b_publisher) VALUES (17285, 'Детство', 'BHV')
Добавление строк, значения которых определяются на основе подзапроса	INSERT INTO Copy_Book VALUES ('Исповедь', (SELECT b_publisher FROM Book WHERE b_id_book = 1)), ('Воскресение', (SELECT b_publisher FROM Book WHERE b_id_book = 2));
Добавление в копию таблицы строк, отобранных в подзапросе на основе некоторого условия	INSERT INTO Copy_Book SELECT * FROM Book WHERE b_title = 'Воскресение';
Добавление данных во вновь создаваемую таблицу Book_copy с помощью инструкции SELECT INTO, которая является вариацией простой инструкции SELECT	SELECT b_id_book, b_publisher, b_title INTO Book_copy FROM Book WHERE b_id_book = 1

Если столбец имеет свойство IDENTITY (столбец идентификаторов, счетчик), при вставке строки имя этого столбца и значение поля для этого столбца в команде INSERT не указывают. Для такого столбца сервер автоматически

вычисляет новое значение. Оператор SET IDENTITY\_INSERT < имя таблицы > { ON | OFF } отключает (ON) или включает (OFF) автоинкремент.

## 11.2 Удаление данных из таблиц средствами T-SQL

Для удаления записей при помощи запросов из существующей таблицы можно использовать инструкцию DELETE. Ее синтаксис следующий:

```
DELETE [FROM] { имя_таблицы | имя_представления }
[ FROM <источник табличного типа>]
[ WHERE условия_поиска ]
```

В качестве исходного объекта для удаляемых строк следует задать либо имя таблицы или представления, либо результаты выполнения функции OPENQUERY, OPENROWSET или OPENDATASOURCE. При помощи источника табличного типа можно конкретизировать данные, удаляемые из таблицы в первом предложении FROM, т. е. можно выполнять соединения таблиц, что логически заменяет использование подзапросов в предложении WHERE для идентификации удаляемых строк.

Инструкция DELETE не может удалять из таблицы строки со значениями NULL на пассивной стороне внешнего объединения (OUTER JOIN).

Если инструкция DELETE пытается нарушить работу триггера или ограничения, поддерживающего целостность ссылок, она не будет выполнена.

При выполнении инструкции DELETE для таблицы, в которой определен триггер INSTEAD OF DELETE, сама инструкция DELETE не будет выполнена. Вместо этого будут выполняться действия триггера для каждой удаляемой строки в таблице [4].

В таблице 11.2 приведены примеры использования оператора DELETE.

*Примечание* – Во избежание нежелательных последствий с помощью инструкции SELECT INTO создается копия таблицы Book и выполняется работа уже с копией:

```
SELECT * INTO Book_copу FROM Book
DELETE FROM Book_copу /* удаление всех строк из таблицы Book_copу */
DROP TABLE Book_copу /* удаление таблицы Book_copу */
```

Таблица 11.2 – Примеры инструкций DELETE

Задача	SQL-код решения
Удаление из таблицы Book строк с указанием условия отбора удаляемых строк	DELETE FROM Book_copу WHERE b_publisher LIKE 'A%';
Удаление тех строк из таблицы Book_copу, для которых нет соответствующих строк в таблице Book, с использованием стандартного синтаксиса оператора DELETE, при этом для определения удаляемых строк используется подзапрос	DELETE FROM Book_copу WHERE b_publisher = 'Лира' AND b_id_book NOT IN (SELECT b_id_book FROM Book);

## Окончание таблицы 11.2

Задача	SQL-код решения
Удаление первых двух строк таблицы Book_copy. Подзапрос (инструкция SELECT в круглых скобках) возвращает базовый набор строк для инструкции DELETE. Результату этого подзапроса присваивается псевдоним bc, а директива WHERE задает параметры сравнения строк из bc с базовой таблицей. Затем директива DELETE автоматически удаляет все совпавшие строки	<pre>SELECT * INTO Book_copy FROM Book DELETE Book_copy FROM (SELECT TOP 2 *       FROM Book_copy ) AS bc WHERE Book_copy.b_id_book =       bc.b_id_book</pre>
Удаление тех строк из таблицы Book_copy, для которых нет соответствующих строк в таблице Book, с использованием дополнительного предложения FROM, которое вводит источник табличного типа, конкретизирующий данные, удаляемые из таблицы в первом предложении FROM	<pre>DELETE FROM Book_copy FROM Book_copy AS bc LEFT JOIN Book ON bc.b_id_book = Book.b_id_book WHERE bc.b_publisher = 'Лира' AND Book.b_id_book IS NULL;</pre>

Инструкция TRUNCATE TABLE ИмяЦелевойТаблицы удаляет все строки в целевой таблице, не записывая в журнал транзакций удаление отдельных строк, за счет чего выполняется быстрее и требует меньших ресурсов системы.

### 11.3 Изменение данных в таблицах средствами T-SQL

Инструкция UPDATE реализует один из способов изменения любых данных, содержащихся в таблице. Можно написать инструкцию UPDATE так, чтобы она влияла только на отдельное поле в отдельной строке либо чтобы она вычисляла изменения в столбце во всех строках таблицы. Можно также написать инструкцию, которая будет изменять все строки из множества столбцов [3].

Синтаксис инструкции UPDATE [4, 5, 7, 8]:

```
UPDATE { имя_таблицы | имя_представления }
SET { имя_столбца = {выражение | DEFAULT | NULL}
| @ переменная = выражение | @переменная-столбец = выражение
{ [ FROM {исходная_таблица} [... n] ]
[ WHERE условие_поиска ] }
```

Для обновляемых строк или ячеек нужно задать имя таблицы или представления. Ключевое слово SET предваряет вносимые изменения. Можно задать значение столбца равным выражению, значению по умолчанию или пустому значению NULL. Можно присвоить выражение локальной переменной. Можно объединить присвоение значения локальной переменной и столбцу в одном и том же выражении. В одной директиве SET можно задать множество столбцов.

Если инструкция UPDATE нарушает некоторое ограничение таблицы, то выполнение инструкции UPDATE будут отменено.

Если инструкция UPDATE влияет на строки в таблице, в которой есть триггер INSTEAD OF UPDATE, то вместо инструкции UPDATE будут выполняться

инструкции в триггере [4].

В таблице 11.3 приведены примеры использования **оператора UPDATE**.

Таблица 11.3 – Примеры инструкций UPDATE

Задача	SQL-код решения
Обновление в таблице Book_cору всех значений столбца b_publisher на основе выражения для определения новых значений	UPDATE Book_cору SET b_publisher = CONCAT('издательство ', b_publisher)
Обновление в таблице Book значений столбца с указанием условия отбора строк для обновления	UPDATE Book SET b_publisher = 'AZ' WHERE b_publisher = 'Лира';
Обновление в таблице Book значений нескольких столбцов с указанием условия отбора строк для обновления	UPDATE Book SET b_publisher = 'AZ', b_title = 'Букварь' WHERE b_publisher = 'Лира';
Обновление в столбце b_publisher таблицы Book значения 'ИНФРА-М' на значение 'Y' во всех строках, где b_publisher='ИНФРА-М'	UPDATE Book SET Book.b_publisher = 'Y' FROM (SELECT * FROM Book WHERE b_publisher='ИНФРА-М') AS BS WHERE Book.b_id_book = BS.b_id_book

Операция обновления столбца со свойством IDENTITY представляет собой комбинацию инструкции DELETE с удалением ненужного значения из ячейки столбца идентификаторов и инструкции INSERT со вставкой требуемого значения в ячейку столбца идентификаторов.

#### **11.4 Некоторые способы отбора записей в запросах**

Оператор IN позволяет определить набор значений, которые должны иметь столбцы: WHERE выражение [NOT] IN (выражение). Выражение в скобках после IN определяет набор значений. Этот набор может вычисляться динамически на основе, например, еще одного запроса либо это могут быть константы.

Например, выберем книги издательств Amedia, либо Aquarius, либо BHV:

```
SELECT * FROM Book
WHERE publisher IN ('Amedia', 'Aquarius', 'BHV')
```

Оператор BETWEEN определяет диапазон значений с помощью начального и конечного значений, которым должно соответствовать выражение: WHERE выражение [NOT] BETWEEN начальное\_значение AND конечное\_значение.

В таблице 11.4 представлены допустимые подстановочные символы, их значения и примеры использования.

Таблица 11.4 – Подстановочные символы, используемые в шаблонах LIKE

Подстановочный знак	Значение	Пример	Результат
%	Символ-шаблон, заменяющий любую последовательность символов	WHERE [title] LIKE 's%'	Строка, начинающаяся с s: Samantha или Sven
_ (подчеркивание)	Символ-шаблон, заменяющий любой одиночный символ	WHERE [titleofcourtesy] LIKE 'm_.'	Ms. Mr.
[<список символов>]	Один символ из списка	WHERE [firstname] LIKE '[sp]%'	Строка, в которой первый символ s или p: Sara или Paul
[<диапазон символов>]	Один символ из диапазона. При этом можно перечислить сразу несколько диапазонов (например, [0-9a-z])	WHERE [freight] LIKE '6[5-7]%'	Строка, в которой второй символ – цифра 5, 6 или 7: 65,83 или 677,54
[^<список или диапазон символов>]	В сочетании с квадратными скобками исключает из поискового образца символы из списка или диапазона	WHERE [shipaddress] LIKE '[^0-9]%'	Строка, в которой первый символ не цифра
ESCAPE ''	Для поиска символа, который является подстановочным знаком, его указывают после Escape-символа с помощью ключевого слова ESCAPE	WHERE col1 LIKE '!_%' ESCAPE '!'	Поиск строки, которая начинается со знака подчеркивания (_), используя в качестве Escape-символа (!)
'ymd'	Для поиска даты используется форма без разделителей, которая не зависит от языка входа в систему для всех типов данных даты и времени	WHERE [birthdate] = '19581208'	1958-12-08 00:00:00.000

### 11.5 Некоторые функции T-SQL

Далее будут рассмотрены примеры использования некоторых наиболее распространенных стандартных функций T-SQL. Так, в таблице 11.5 описано применение строковых функций.

Таблица 11.5 – Строковые функции T-SQL

Функция	Назначение	Пример
LEN(s)	В качестве параметра в функцию передается строка s, для которой возвращается количество символов	SELECT LEN('Sun') -- 3
LTRIM(s), RTRIM(s)	В качестве параметра принимает строку s, в которой удаляет начальные (для RTRIM()) конечные) пробелы	SELECT LTRIM(' Sun')
CHARINDEX(s1, s2)	Возвращает начальную позицию первого вхождения подстроки в строке. Первым параметром передается подстрока, вторым – строка, в которой ведется поиск	SELECT CHARINDEX('u', 'Sun') -- 2



Продолжение таблицы 11.5

Функция	Назначение	Пример
PATINDEX (%p%, s)	Возвращает начальную позицию первого вхождения шаблона <i>p</i> в указанной строке <i>s</i> или ноль, если шаблон <i>p</i> не найден	SELECT PATINDEX('%u%', 'Sun') -- 2
LEFT( <i>s</i> , <i>length</i> ), RIGHT( <i>s</i> , <i>length</i> )	Возвращает указанное число символов символического выражения слева (для RIGHT( <i>s</i> , <i>length</i> ) – справа). Первый параметр функции – строка <i>s</i> , второй – количество символов <i>length</i> , которые надо вырезать	SELECT LEFT('Sun', 2) -- Su
SUBSTRING ( <i>s</i> , <i>a</i> , <i>length</i> )	Вырезает из строки <i>s</i> , начиная с позиции <i>a</i> , подстроку определенной длины <i>length</i>	SELECT SUBSTRING('Mouse', 2, 3) -- ous
REPLACE( <i>s1</i> , <i>s2</i> , <i>s3</i> )	Заменяет все вхождения указанного строкового значения <i>s2</i> другим строковым значением <i>s3</i> . Первый параметр функции – исходная строка <i>s1</i> , второй – подстрока <i>s2</i> , которую надо заменить, а третий – подстрока <i>s3</i> , на которую надо заменить	SELECT REPLACE('Sun', 'u', 'o') -- Son
TRANSLATE (inputString, characters, translations)	Возвращает строку, указанную в качестве первого аргумента, после преобразования символов, указанных во втором аргументе, в конечный набор символов, указанный в третьем аргументе. TRANSLATE возвращает ошибку, если выражения characters и translations имеют разную длину. TRANSLATE возвращает значение NULL, если любой из аргументов имеет значение NULL	Замена квадратных и фигурных скобок обычными: SELECT TRANSLATE('2*[3+4] / {7-2}', '[ ]', '()'); -- 2*(3+4) / (7-2)
CONCAT()	Возвращает строку, возникающую в результате объединения двух или более строковых значений	SELECT CONCAT('Ma', 'ma')
CONCAT_WS ( separator, arg1, arg2 [, argN]... )	Возвращает строку, возникающую в результате объединения двух или более строковых значений, разделенных символами, указанными в разделителе separator. Разделитель между NULL не добавляется	SELECT CONCAT_WS('-', '111', '11', '11') -- 111-11-11
LOWER( <i>s</i> ), UPPER( <i>s</i> )	Преобразует строку <i>s</i> в нижний (для UPPER – в верхний) регистр	SELECT LOWER('Sun')
STUFF( <i>s1</i> , <i>a</i> , <i>length</i> , <i>s2</i> )	Удаляет из строки <i>s1</i> количество символов <i>length</i> , начиная с позиции <i>a</i> , и вставляет на их место строку <i>s2</i>	SELECT STUFF('Notebook', 5, 0, ' in a ' ) -- Note in a book
SOUNDEX()	Функция SOUNDEX преобразует алфавитно-цифровую строку в четырехсимвольный код (для оценки степени сходства двух строк), чье значение зависит от способа звучания строки при произношении на английском языке	SELECT SOUNDEX ( 'Smith' ), SOUNDEX ( 'Smythe' ); -- S530 S530
DIFFERENCE( <i>s1</i> , <i>s2</i> )	Возвращает целое число от 0 до 4, которое является разницей между значениями SOUNDEX() двух строк <i>s1</i> и <i>s2</i> . Функция SOUNDEX() возвращает число, которое характеризует звучание строки, что позволяет определить сходным образом звучащие строки. Значение 0 указывает на слабое сходство значений SOUNDEX или его отсутствие; значение 4 – на сильное сходство или одинаковые значения SOUNDEX	SELECT DIFFERENCE ('Smith', 'Smythe'); -- 4 SELECT DIFFERENCE('spelling', 'telling') -- 2

## Окончание таблицы 11.5

Функция	Назначение	Пример
REVERSE( <i>s</i> )	Переворачивает строку <i>s</i> , где символы переставлены в обратном порядке справа налево	SELECT REVERSE('123')
REPLICATE( <i>s</i> , <i>n</i> )	Повторяет <i>n</i> раз строку <i>s</i>	SELECT REPLICATE('a', 3) -- 'aaa'
SPACE( <i>length</i> )	Возвращает строку пробелов длиной, указанной в параметре <i>length</i>	SELECT SPACE(4)
STR ( <i>float_expression</i> [, <i>length</i> [, <i>decimal</i> ]])	Возвращает символьные данные типа varchar, преобразованные из числовых данных типа float. Параметр <i>length</i> – общая длина (по умолчанию 10), включает десятичную запятую, знак, цифры и пробелы. <i>decimal</i> – число знаков справа от десятичной запятой. Значение <i>decimal</i> должно быть меньше или равно 16	SELECT STR(123.45, 6, 1); -- 123.5 SELECT STR(FLOOR(123.45), 8, 3); -- 123.000
FORMAT ( <i>value</i> , <i>format</i> [, <i>culture</i> ] )	Возвращает значение в указанных формате и культуре (не обязательно) при форматировании значения даты, времени и чисел с учетом локали в виде строк (Для общих преобразований типов данных используются CAST и CONVERT)	SELECT FORMAT(123456, '###-##-#') AS 'Telephone Number'; -- 123-45-6

Некоторые математические функции приведены в таблице 11.6.

Таблица 11.6 – Математические функции T-SQL

Функция	Назначение	Пример
CEILING( <i>n</i> )	Возвращает наименьшее целое значение, большее или равное параметру <i>n</i>	SELECT CEILING(4.88) --5
FLOOR ( <i>numeric_expression</i> )	Возвращает наибольшее целое число, меньшее или равное указанному числовому выражению	SELECT FLOOR(123.45), FLOOR(-123.45); -- 123 -124
RAND ([ <i>seed</i> ])	Возвращает произвольное число типа FLOAT в диапазоне значений от 0 до 1. Для указанного <i>seed</i> возвращаемый результат всегда будет один и тот же	SELECT RAND(11) -- 0,713778322925506
ROUND( <i>n</i> , <i>p</i> , [ <i>t</i> ])	Округляет значение <i>n</i> с точностью до <i>p</i> . Если аргумент <i>p</i> положительное число, округляется дробная часть числа <i>n</i> , если отрицательное – целая часть. При использовании необязательного аргумента <i>t</i> ≠ 0 число <i>n</i> не округляется, а усекается	SELECT ROUND(5.4567, 3) -- 5.4570 SELECT ROUND(345.4567, -1) -- 350.0000 SELECT ROUND(345.4567-1, 1) -- 340.0000

Агрегатные функции выполняют вычисления над значениями в наборе строк и возвращают одиночное значение. Все агрегатные функции, за исключением COUNT(\*), игнорируют значения NULL (таблица 11.7).

Выражение в функциях AVG и SUM должно представлять числовое значение. Выражение в функциях MIN, MAX и COUNT может представлять числовое или строковое значение или дату.

Агрегатные функции могут использоваться только в списке предложения SELECT и в составе предложения HAVING.

Таблица 11.7 – Некоторые агрегатные функции

Синтаксис	Назначение
AVG()	Вычисляет среднее значение для указанного столбца
SUM()	Суммирует все значения в указанном столбце
MIN()	Находит минимальное значение в указанном столбце
MAX()	Находит максимальное значение в указанном столбце
COUNT()	Подсчитывает количество строк с непустым (NOT NULL) значением указанного столбца
COUNT(*)	Подсчитывает общее количество строк, удовлетворяющих условию, включая пустые (NULL)

Некоторые функции для работы с датой и временем приведены в таблице 11.8.

Таблица 11.8 – Функции T-SQL для работы с датой и временем

Функция	Назначение	Пример
GETDATE()	Возвращает текущую системную дату и время	SELECT GETDATE();
DATEPART ( <i>datepart</i> , <i>date</i> )	Возвращает указанную в параметре <i>datepart</i> часть даты <i>date</i> в виде целого числа. Для <i>datepart</i> могут использоваться следующие сокращения: year – yy, yyyу quarter – qq, q month – mm, m day – dd, d week – wk, ww dayofyear – dy, y hour – hh minute – mi, n second – ss, s weekday – dw millisecond – ms microsecond – mcs	SELECT DATE- PART(month, '2023-01-30') -- 1
DATENAME ( <i>datepart</i> , <i>date</i> )	Возвращает указанную в параметре <i>datepart</i> часть даты <i>DATE</i> в виде строки символов – ее названия	SELECT DATE- NAME(weekday, '2023-01-30')
DATEDIFF ( <i>datepart</i> , <i>startdate</i> , <i>enddate</i> )	Вычисляет разницу между двумя частями дат <i>startdate</i> и <i>enddate</i> и возвращает целочисленный результат со знаком в единицах, указанных в аргументе <i>datepart</i>	SELECT DATEDIFF(year, BirthDate, GET- DATE()) AS age FROM employee
DATEADD ( <i>datepart</i> , <i>num- ber</i> , <i>date</i> )	Прибавляет <i>number</i> (целое число со знаком) в область <i>datepart</i> и возвращает измененное значение <i>date</i> – даты или времени (значение аргумента <i>n</i> также может быть отрицательным)	Добавляет 3 дня: SELECT DATE- ADD(DAY, 3, GET- DATE())
CURRENT_ TIMESTAMP()	Возвращает текущую системную метку времени (операционной системы компьютера) базы данных в виде значения <i>datetime</i> без смещения часового пояса	SELECT CURRENT_ TIMESTAMP()
EOMONTH ( <i>date</i> )	Возвращает последнее число месяца, содержащего указанную дату, с необязательным смещением	SELECT DAY(EOMONTH '(2022-12-25')) --31
DATEFROM- PARTS( <i>year</i> , <i>month</i> , <i>day</i> )	По целочисленным значениям года, месяца и дня создает дату типа <i>DATE</i>	SELECT DATE- FROMPARTS (2021, 2, 28)
YEAR( <i>date</i> ), MONTH( <i>date</i> ), DAY( <i>date</i> )	Возвращает целое число, показывающее год (месяц, день) в указанной дате	SELECT YEAR '(2022-01-30')

Некоторые функции преобразования данных приведены в таблице 11.9.

Таблица 11.9 – Функции преобразования данных T-SQL

Функция	Назначение	Пример
CAST( <i>expr</i> AS <i>type</i> [( <i>length</i> )])	Стандартная функция CAST() преобразовывает выражение <i>expr</i> в указанный тип данных <i>type</i> (если это возможно). По умолчанию <i>length</i> = 30	SELECT CAST (3.141 AS INT) -- 3
CONVERT( <i>type</i> [( <i>length</i> )] , <i>expr</i> [ , <i>style</i> ])	Нестандартная функция CONVERT() позволяет применить при преобразовании типов данных стиль (значения стилей указаны по адресу: <a href="https://learn.microsoft.com/ru-ru/sql/t-sql/functions/">https://learn.microsoft.com/ru-ru/sql/t-sql/functions/</a> )	SELECT CON- VERT(CHAR, GETDATE(), 6) -- 27 дек 22
TRY_CAST( <i>expr</i> AS <i>type</i> [( <i>length</i> )])	Возвращает значение, приведенное к указанному типу, если приведение проходит успешно; в противном случае возвращает NULL (аналогично работает и TRY_CONVERT( <i>type</i> [( <i>length</i> )] , <i>expr</i> [ , <i>style</i> ]))	SELECT TRY_CAST('12/27 /2022' AS date) -- NULL
TRY_PARSE( <i>string_value</i> AS <i>type</i> [ USING <i>culture</i> ])	Возвращает результат выражения <i>string_value</i> , преобразованный в запрошенный тип данных <i>type</i> , или значение NULL, если привести тип не удастся. Используется только для преобразования данных из строкового типа в типы даты или времени и числовые типы	SELECT TRY_PARSE('qwe rty' AS datetime2 USING 'en-US') -- NULL

Некоторые системные функции и функции безопасности приведены в таблице 11.10.

Таблица 11.10 – Некоторые функции T-SQL

Функция	Назначение	Пример
CURRENT_USER	Возвращает имя текущего пользователя. Она эквивалентна функции USER_NAME()	SELECT CURRENT_USER -- dbo
COALESCE( <i>a1</i> , <i>a2</i> , ...)	Возвращает первое значение выражения из списка выражений <i>a1</i> , <i>a2</i> , ... , которое не является значением NULL	SELECT COA- LESCE(NULL, NULL, 'value3', 'value4'); -- value3
COL_LENGTH('table', 'column')	Возвращает длину в байтах (типа SMALLINT) столбца <i>column</i> объекта <i>table</i> базы данных (таблицы или представления)	SELECT COL_LENGTH ( 'Employee', 'Id')
DATALENGTH( <i>expr</i> )	Возвращает число байтов, использованных для представления выражения <i>expr</i> (любого типа данных). Возвращает значение типа bigint, если <i>expr</i> имеет тип данных NVARCHAR(MAX), VARBINARY(MAX) ИЛИ VARCHAR(MAX); в противном случае INT	SELECT DATALENGTH (ProductName) FROM products;
GETANSINULL('dbname')	Возвращает 1, если использование значений NULL в базе данных <i>dbname</i> отвечает требованиям стандарта ANSI SQL	SELECT GETANSINULL ( 'AdventureWorks') -- 1
INDEX_COL( <i>table</i> , <i>i</i> , <i>no</i> )	Возвращает имя индексированного столбца таблицы <i>table</i> . Столбец указывается идентификатором индекса <i>i</i> и позицией <i>no</i> столбца в этом индексе	INDEX_COL ( 'SalesOrderDetail', 1, 1)

## Окончание таблицы 11.10

Функция	Назначение	Пример
NEWID()	Создает однозначный идентификационный номер ID, состоящий из 16-байтовой двоичной строки, предназначенной для хранения значений типа данных UNIQUEIDENTIFIER	CustomerID UNIQUEIDENTIFIER NOT NULL DEFAULT newid()
USER_NAME ([id])	Возвращает имя пользователя по указанному идентификатору <i>id</i> . Если идентификатор не указан, то возвращается имя текущего пользователя	SELECT USER_NAME(2) -- 'guest'
ROWCOUNT_ BIG	Возвращает количество (тип BIGINT) строк таблицы, обработанных последней инструкцией T-SQL	

**Задание**

Необходимо для разрабатываемой БД написать для каждой таблицы по три различных команды INSERT, UPDATE, DELETE с использованием различных функций (не менее 15 различных функций) и предикатов в условии отбора.

**Содержание отчета:** тема и цель работы; прокомментированный SQL-код выполнения задания.

**Контрольные вопросы**

1 Объяснить синтаксис команд INSERT, SELECT INTO, UPDATE, DELETE и указать ограничения для данных команд.

2 Охарактеризовать особенности выполнения команд INSERT, UPDATE, DELETE по отношению к столбцу IDENTITY.

3 Указать назначение, преимущества и ограничения инструкции TRUNCATE TABLE.

4 Перечислить подстановочные символы, используемые в шаблонах LIKE.

**Список литературы**

1 **Агальцов, В. П.** Базы данных [Электронный ресурс]: учебник: в 2 т. Т. 2. Распределенные и удаленные базы данных / В. П. Агальцов. – Москва : ФОРУМ ; ИНФРА-М, 2021. – 271 с. – Режим доступа: <https://znanium.com/catalog/product/1514118>. – Дата доступа: 03.12.2022.

2 **Агальцов, В. П.** Базы данных [Электронный ресурс]: учебник: в 2 кн. Кн. 1. Локальные базы данных / В. П. Агальцов. – Москва: ФОРУМ; ИНФРА-М, 2020. – 352 с.: ил. – Режим доступа: <https://znanium.com/catalog/product/1068927>. – Дата доступа : 03.12.2022.

3 **Бен-Ган, И.** Microsoft SQL Server 2012. Создание запросов : учебный курс Microsoft : пер. с англ. / И. Бен-Ган, Д. Сарка, Р. Талмейдж. – Москва : Русская редакция, 2015. – 720 с. : ил.

4 **Бондарь, А. Г.** Microsoft SQL Server 2014 / А. Г. Бондарь. – Санкт-Петербург : БХВ-Петербург, 2015. – 592 с. : ил.

5 **Гвоздева, Т. В.** Проектирование информационных систем: технология автоматизированного проектирования. Лабораторный практикум : учебно-справочное пособие / Т. В. Гвоздева, Б. А. Баллод. – Санкт-Петербург ; Москва ; Краснодар : Лань, 2018. – 156 с. : ил.

6 **Голицына, О. Л.** Базы данных [Электронный ресурс]: учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 4-е изд., перераб. и доп. – Москва: ФОРУМ; ИНФРА-М, 2020. – 400 с. – Режим доступа: <https://znanium.com/catalog/product/1053934>. – Дата доступа : 03.12.2022.

7 **Дадян, Э. Г.** Данные: хранение и обработка [Электронный ресурс]: учебник / Э. Г. Дадян. – Москва : ИНФРА-М, 2021. – 205 с. – Режим доступа: <https://znanium.com/catalog/product/1149101>. – Дата доступа : 03.12.2022.

8 **Кузин, А. В.** Разработка баз данных в системе Microsoft Access [Электронный ресурс]: учебник / А. В. Кузин, В. М. Демин. – 4-е изд. – Москва: ФОРУМ; ИНФРА-М, 2020. – 224 с. – Режим доступа: <https://znanium.com/catalog/product/1058247>. – Дата доступа : 03.12.2022.

9 **Кузин, А. В.** Базы данных: учебное пособие для студентов высших учебных заведений / А. В. Кузин, С. В. Левонисова. – 6-е изд., стер. – Москва: Академия, 2016. – 320 с.

10 **Куликов, С. С.** Работа с MySQL, MS SQL Server и Oracle в примерах [Электронный ресурс]: практическое пособие / С. С. Куликов. – Минск : БОФФ, 2016. – 556 с. – Режим доступа: [http://svyatoslav.biz/database\\_book/](http://svyatoslav.biz/database_book/). – Дата доступа: 20.09.2022.

11 **Куликов, С. С.** Реляционные базы данных в примерах : практическое пособие для программистов и тестировщиков [Электронный ресурс] / С. С. Куликов. – Минск: Четыре четверти, 2020. – 424 с. – Режим доступа : [http://svyatoslav.biz/relational\\_databases\\_book/](http://svyatoslav.biz/relational_databases_book/). – Дата доступа : 03.12.2022.

12 **Полищук, Ю. В.** Базы данных и их безопасность [Электронный ресурс]: учебное пособие / Ю. В. Полищук, А. С. Боровский. – Москва : ИНФРА-М, 2020. – 210 с. – Режим доступа: <https://znanium.com/catalog/product/1011088>. – Дата доступа: 03.12.2022.

13 **Советов, Б. Я.** Базы данных: теория и практика : учебник для бакалавров. – 2-е изд. – Москва : Юрайт, 2012. – 463 с.

14 **Шустова, Л. И.** Базы данных: учебник [Электронный ресурс] / Л. И. Шустова, О. В. Тараканов. – Москва : ИНФРА-М, 2021. – 304 с. – Режим доступа: <https://znanium.com/catalog/product/1362122>. – Дата доступа: 03.12.2022.