ISSN 2221-1993 • **Informatics in School** • 2026 • Volume 25 No 2

DOI: 10.32517/2221-1993-2026-25-2-22-35

Д. С. Беленкевский, И. В. Симонова
Российский государственный педагогический университет им. А. И. Герцена, г. Санкт-Петербург, Россия

ПОДГОТОВКА БУДУЩИХ УЧИТЕЛЕЙ ИНФОРМАТИКИ ДЛЯ ОБУЧЕНИЯ ШКОЛЬНИКОВ ОСНОВАМ SQL И СОВРЕМЕННЫМ СУБД: ДЕМОНСТРАЦИОННЫЕ ПРИМЕРЫ И СИТУАЦИОННЫЕ ЗАДАЧИ*

Аннотация

В новой редакции ФГОС среднего общего образования обозначена необходимость усиления знаний, умений и навыков школьников в области разработки и проектирования информационных систем, работы с базами данных и языком структурированных запросов. В школьных учебниках рассматривается ряд связанных понятий, таких как: модель данных, реляционная база данных, первичный ключ, сущности, связи и т. д. В статье представлено описание содержания обучения будущих учителей информатики по тематике, связанной с теоретическими основами современных систем управления базами данных (СУБД), баз данных (БД) и информационных систем, дана сравнительная характеристика современных свободно распространяемых СУБД, поддерживающих язык SQL, которые могут быть использованы при обучении студентов. Рассматривается реализация информационной системы, включающей базу данных и приложение на языке Python, на примере ситуационной задачи «Менеджер домашних заданий школьника». Подробно описываются этапы создания базы данных: структура таблицы, типы данных полей, запросы к БД и т. д.; этапы написания кода в редакторе кода IDLE. Приведен демонстрационный пример БД-приложения на языке Python, созданного с использованием встроенных библиотек, он позволяет проиллюстрировать работу с базой данных через графический интерфейс. Материалы статьи не только предлагаются для обучения студентов — будущих учителей информатики, но и апробированы студентами педвуза в ходе предметно-содержательной и производственной практики. Результаты показали заинтересованность студентов в разработке подобных информационных систем совместно с учащимися профильных классов.

Ключевые слова: подготовка учителей информатики, разработка баз данных, системы управления базами данных, проектирование баз данных, информационные системы, профильное обучение информатике.

Информация об авторах

Беленкевский Дмитрий Сергеевич, ассистент кафедры цифрового образования, Институт информационных технологий и технологического образования, Российский государственный педагогический университет им. А. И. Герцена, г. Санкт-Петербург, Россия; *e-mail*: belenkevskijd@herzen.spb.ru

Симонова Ирина Викторовна, доктор пед. наук, профессор, профессор кафедры цифрового образования, Институт информационных технологий и технологического образования, Российский государственный педагогический университет им. А. И. Герцена, г. Санкт-Петербург, Россия; *e-mail:* ir_1@mail.ru

* Материалы к статье можно скачать на сайте ИНФО: https://infojournal.ru/journals/school/school_02-2026/

D. S. Belenkevskiy, I. V. Simonova
The Herzen State Pedagogical University of Russia, St. Petersburg, Russia

PREPARING FUTURE INFORMATICS TEACHERS TO TEACH SCHOOLCHILDREN THE BASICS OF SQL AND MODERN DBMS: DEMONSTRATION EXAMPLES AND CASE STUDIES

Abstract

The new edition of the Federal State Educational Standard for Secondary General Education identifies the need to strengthen schoolchildren's knowledge, skills, and abilities in the development and design of information systems, working with databases, and structured query language. School textbooks cover a number of related concepts, such as data models, relational databases, primary keys, entities, relationships, etc. The article describes the training content for future informatics teachers on topics related to the theoretical foundations of modern database management systems (DBMS), databases, and information systems. It also provides a comparative analysis of modern open source DBMSs that support the SQL language and can be used in teaching students. The article discusses the implementation of an information system including a database and a Python application using the example of the situational task "Student homework manager". The stages of database creation are described in detail: table structure, field data types, database queries, etc.; and the stages of writing code in the IDLE code editor. A demonstration example of a Python database application, created using built-in libraries, is presented. It illustrates working with a database through a graphical interface. The article's materials are not only offered for teaching future informatics teachers but were also tested by students at the pedagogical university during their internships and practical training. The results demonstrated the students' interest in developing similar information systems in collaboration with schoolchildren in specialized classes.

Keywords: informatics teacher training, database development, database management systems, database design, information systems, specialized training in informatics.

ISSN 2221-1993 • **Информатика в школе** • 2026 • Том 25 № 2

Information about the authors

Dmitry S. Belenkevsky, Assistant Professor, Department of Digital Education, Institute of Information Technologies and Technological Education, The Herzen State Pedagogical University of Russia, St. Petersburg, Russia; *e-mail:* belenkevskijd@herzen.spb.ru

Irina V. Simonova, Doctor of Sciences (Education), Professor, Professor at the Department of Digital Education, Institute of Information Technologies and Technological Education, The Herzen State Pedagogical University of Russia, St. Petersburg, Russia; *e-mail:* ir_1@mail.ru

1. Введение

Новая редакция ФГОС среднего общего образования акцентирует внимание на профилизации и персонализации обучения, уточняет требования к предметным результатам и применению конкретных знаний и умений, усиливает практико-ориентированную направленность обучения и проектную деятельность [13]. Согласно требованиям к предметным результатам по учебному предмету «Информатика» (углубленный уровень), делается акцент на информационные модели и моделирование, в связи с чем в содержании усиливается раздел, связанный с теоретическими основами современных систем управления базами данных (СУБД), баз данных (БД) и информационных систем (ИС). Эти положения находят отражение в утвержденных учебных материалах для школы, также о необходимости более глубокого изучения данного раздела свидетельствует опыт реализации образовательной программы для бакалавров педагогического образования, профиль «Информатика и информационные технологии в образовании»: «Тематические проекты помогают повысить интерес обучающихся к изучаемым дисциплинам и мотивируют их к совершенствованию навыков работы с современными средствами информационных технологий» [2].

Наши исследования и опыт подготовки будущих учителей информатики показывают, что в обучении проектированию информационных систем и баз данных особая роль отводится практико-ориентированности содержания обучения, а также «изучению теоретических основ построения реляционных баз данных, базирующихся на математическом аппарате теории множеств и математической логике» [7].

Содержание по этой теме включает лекционные материалы, охватывающие базовые аспекты теории СУБД, БД и информационных систем, и лабораторные задания. Многолетний опыт показал, что подробное описание шагов выполнения лабораторной работы, включая создание базы данных, таблиц, добавление данных и разработку адаптивного приложения на языке Python, обеспечивает возможность самостоятельного решения обучающимися аналогичных ситуационных задач [1, 3].

В настоящей статье предложена реализация процесса обучения проектированию БД, которая предполагает [10, 12]:

- формирование набора знаний, умений и навыков в области проектирования баз данных и управления ими при многообразии доступных СУБД;
- формирование умения работы с данными, написания SQL-запросов, создания многотабличных баз данных, реализации отношений (таблиц), работы с запросами для фильтрации и сортировки данных;
- развитие умений при работе с графическими оболочками администрирования баз данных (MySQL Workbench, SQLite Browser, pgAdmin, IBExpert);
- получение опыта реализации БД-приложений при интеграции существующих баз данных с языками программирования Python, Object Pascal (Delphi).

2. Содержание обучения

2.1. Основные понятия теории баз данных

Анализ содержания научной, учебной литературы, публикаций и наш опыт преподавания позволили выделить **ключевые понятия в содержании обучения базам данных будущих учителей информатики** [4, 9, 11]:

- Общие понятия: база данных, система управления базами данных, модель данных, информационная система.
- Модели данных: иерархическая, сетевая, реляционная, документоориентированная, ключ-значение, графовая, колоночная, объектно-ориентированная и т. д.
- Реляционная модель данных: таблица (отношение), поле (атрибут), запись (кортеж), типы данных, домен, первичный ключ, внешний ключ, ограничения и т. д.
- Базовый SQL: CREATE, ALTER, DROP, SELECT, INSERT, UPDATE, DELETE.
- Расширенный SQL: управление транзакциями, функции агрегации; фильтрация, сортировка, группировка полей; объединение таблиц и т. д.
- Типы связей: один к одному, один ко многим, многие ко многим.
- Нормализация отношений: первая нормальная форма (1НФ), вторая нормальная форма (2НФ), третья нормальная форма (3НФ).

2.2. Обзор СУБД для обучения будущих учителей информатики

В таблице 1 представлен сравнительный анализ актуальных СУБД, доступных для обучения, — SQLite, MySQL, PostgreSQL. Общими критериями отбора выступили: поддержка реляционной модели данных, свободное распространение, кроссплатформенность, наличие интеграции с языками программирования (Python) [5, 6].

Перечисленные в таблице 1 системы являются свободно распространяемыми, но нужно проверять, что устанавливаемое программное обеспечение соответствует законодательству Российской Федерации и отвечает нормативно-правовым актам образовательного учреждения. При скачивании СУБД даже из официальных источников могут быть ограничения со стороны провайдеров.

На основе опыта преподавания цикла дисциплин по проектированию и разработке баз данных нами были выбраны следующие СУБД: в начале обучения студентов используется файл-серверная СУБД SQLite, а затем возможен переход к профессиональным решениям — MySQL или PostgreSQL.

ISSN 2221-1993 • **Informatics in School** • 2026 • Volume 25 No 2

Таблица 1

Сравнение актуальных систем управления базами данных

№ п/п	Критерии сравнения	SQLite	MySQL	PostgreSQL
1	Типы данных	Integer, Real, Text, Blob, Null	Integer, Smallint, Bigint, Real, Double, Decimal, Varchar, Text, Date, Time, Timestamp, Boolean, Json и т. д.	Integer, Smallint, BIgint, Real, Double, Decimal, Varchar, Text, Date, Time, Timestamp, Boolean, Json, Uuid, Array, Inet и т.д.
2	Особенности диалекта SQL	Гибкая типизация	Строгое соответствие типов	Строгое соответствие типов
		Встроенный автоинкремент	Автоинкремент через ключевое слово	Автоинкремент как специальный псевдотип
		Работа с датами через функции	Полная поддержка типов Date, Time, Timestamp	Полная поддержка типов Date, Time, Timestamp
3	Работа с Python	Встроена в Python	Установка через менеджер пакетов	Установка через менеджер пакетов
4	Развертывание (тип сервера)	Встроенная (файл-серверная, библиотека)	Клиент-серверная	Клиент-серверная
5	Наличие GUI-оболочки	DB Browser DBeaver	XAMPP — PhpMyAdmin MySQL Workbench	PgAdmin4 DBeaver
6	Адаптация под школы	Не требует установки, включена в стандартную библиотеку Python	Требуется установка дополнительного ПО для развертывания сервера XAMPP, MySQL Workbench	Требуется установка дополнительного ПО для развертывания сервера (устанавливается вместе с СУБД)
7	Тип лицензии	Общественное достояние (Public Domain)	GNU GPL (Community Edition)	PostgreSQL License (свободная, MIT-style)

3. Ситуационная задача и реализация демонстрационного примера

Практическая часть обучения осуществляется с использованием набора ситуационных задач, при составлении которых учитывается ряд требований [8, 14]:

- ***Контекст задачи должен учитывать интересы обучающихся и профессиональную направленность***.
- ***Решение задачи включает несколько этапов***, для которых предлагается развернутое описание действий и указывается уровень сложности:
  - *первый этап* предполагает проектирование базы данных, описание таблиц, отношений, полей и типов данных; результатом этапа является логическая модель БД;
  - *второй этап* заключается в выборе СУБД на основе учета когнитивной сложности задачи, создании таблиц БД, наполнении базы данных;
  - *третий этап* предполагает написание запросов на языке SQL — выборку данных, использование функций агрегации, сортировку, фильтрацию и т. п.;
  - *на четвертом этапе* реализуется связь базы данных с приложением на языке Python (см. табл. 1) (особенности технологии связи с конкретной СУБД проиллюстрируем ниже на примерах);
  - *пятым, заключительным, этапом* решения задачи является разработка полноценного БД-приложения.
- ***Контекст задачи может формироваться следующим образом:***
  - выявляются необходимые типы данных и допустимые (базовые) манипуляции с ними;
  - составляется список общих полей, которые удовлетворяют выбранным типам и подходят под большинство запросов к БД;
  - список полей дополняется специфичными для соответствующей области полями;
  - формируется контекст задачи и список возможных запросов.

Далее мы рассмотрим пример ситуационной задачи, ее содержание приведено в таблице 2.

При формировании контекста задачи рекомендуется указывать обоснование разработки (практической работы) и цель решения задачи. Указание типов данных, рекомендуемых наименований и ограничений позволяет избежать ошибок при создании базы данных и формировании дальнейших действий с данными. Усложнение запросов от уровня к уровню, при сохранении общей

ISSN 2221-1993 • **Информатика в школе** • 2026 • Том 25 № 2

Таблица 2

Пример ситуационной задачи «Менеджер домашних заданий школьника»

Контекст задачи		
Вы являетесь студентом — будущим учителем информатики. Готовясь к педагогической практике в школе, вы разрабатываете учебно-методические материалы для учащихся по созданию информационной системы «Менеджер домашних заданий школьника». Учебно-методические материалы включают базу данных и набор запросов к ней, которые позволят систематизировать список домашних работ ученика и оценок, полученных за их выполнение. База данных состоит из одной таблицы. Для ввода и визуализации данных необходимо создать приложение на языке Python.		
Поля и типы данных (одна таблица)		
id	INTEGER PRIMARY KEY AUTOINCREMENT	
homework	TEXT (255), LENGTH(«homework») <= 255	
subject	TEXT (255), LENGTH(«subject») <= 255	
mark	INTEGER, «mark» BETWEEN 2 AND 5	
due_date	TEXT, LENGTH(«due_date») <= 10	
is_complete	INTEGER, DEFAULT 0, «is_complete» IN (0,1)	
Запросы к БД		
Уровень 1 (начальный)	Выведите все поля таблицы	`SELECT * FROM homeworks`

	Выведите поля: «предмет», «домашняя работа», «выполнить до» и отсортируйте результат по предмету в алфавитном порядке	`SELECT subject, homework, due_date`
		`FROM homeworks`
		`ORDER BY subject ASC`

	Выведите все невыполненные задания, отсортируйте записи по дате	`SELECT subject, homework, due_date`
		`FROM homeworks`
		`WHERE is_complete = 0`
		`ORDER BY due_date ASC`

Уровень 2 (средний)	Выведите по алфавиту все неповторяющиеся предметы, записи которых есть в базе данных	`SELECT DISCTINCT subject`
		`FROM homeworks`
		`ORDER BY subject ASC`

	Найдите все выполненные домашние задания по русскому языку с указанием текста работы, оценки и даты, до которой работа выполнялась	`SELECT homework, mark, due_date`
		`FROM homeworks`
		`WHERE subject = 'Русский язык' AND is_complete = 1`

	Посчитайте количество выполненных и невыполненных заданий (функция COUNT)	`SELECT is_complete, COUNT(*) as homeworks_count`
		`FROM homeworks`
		`GROUP BY is_complete`

Уровень 3 (продвинутый)	Оформите результат предыдущего запроса таким образом, чтобы вместо 0 и 1 было текстовое представление значений полей (функция IIF или CASE)	`SELECT`
		`IIF(is_complete = 1, 'Выполнено', 'Не выполнено')`
		`  as status,`
		`COUNT(*) as homeworks_count`
		`FROM homeworks`
		`GROUP BY is_complete`

	Выведите количество всех оценок по каждому предмету, отсортируйте записи по предмету и оценкам (функция COUNT)	`SELECT subject, mark, COUNT(*)`
		`FROM homeworks`
		`WHERE mark IS NOT NULL`
		`GROUP BY subject, mark`
		`ORDER BY subject, mark`

	Посчитайте сложность предмета согласно среднему баллу оценок, полученных за домашнюю работу (функция AVG)	`SELECT`
		`    subject,`
		`    AVG(mark) as avg_mark,`
		`    IIF(AVG(mark) >= 4, 'Легкий предмет', 'Сложный`
		`      предмет') as complexity`
		`FROM homeworks`
		`WHERE mark IS NOT NULL`
		`GROUP BY subject`

ISSN 2221-1993 • **Informatics in School** • 2026 • Volume 25 No 2

структуры запросов, предполагает добавление новых условий выборки.

Опишем последовательность действий (инструкцию для студентов) при разработке демонстрационного примера «Менеджер домашних заданий школьника» на Python, включая создание базы данных в DB Browser.

Шаг 1.

Создайте на своем компьютере рабочую папку под проект, например, *homework-app*. Затем откройте редактор для работы с базами данных DB Browser и найдите в левом верхнем углу раздел *New Database*. Вам предложат указать файл базы данных — назовите его *homework_manager* и сохраните в созданной папке. Далее у вас откроется окно редактирования определения таблицы.

Заполните поля данными, как показано на рисунке 1. Обратите внимание на прописанные ограничения в поле *Проверить*.

Нажмите кнопку *OK* — таблица появится в списке созданных баз данных.

На рисунке 2 приведена структура таблицы в СУБД SQLite3 — именно такая должна получиться на шаге 1.

Шаг 2.

Внесите от пяти записей в вашу базу данных (раздел *Browse data*). Проверьте корректность внесенных данных — они должны удовлетворять типам данных в таблице.

Обратите внимание: значение поля «id» будет подставляться автоматически, поскольку при создании первичного ключа был указан AUTOINCREMENT (см. нижнюю строку на рисунке 1).

Пример заполненной таблицы представлен на рисунке 3.

Шаг 3.

Реализуйте запросы на языке SQL (представлены в таблице 2) — для этого откройте раздел *Execute SQL*. Результаты сохраните в вашей рабочей папке.

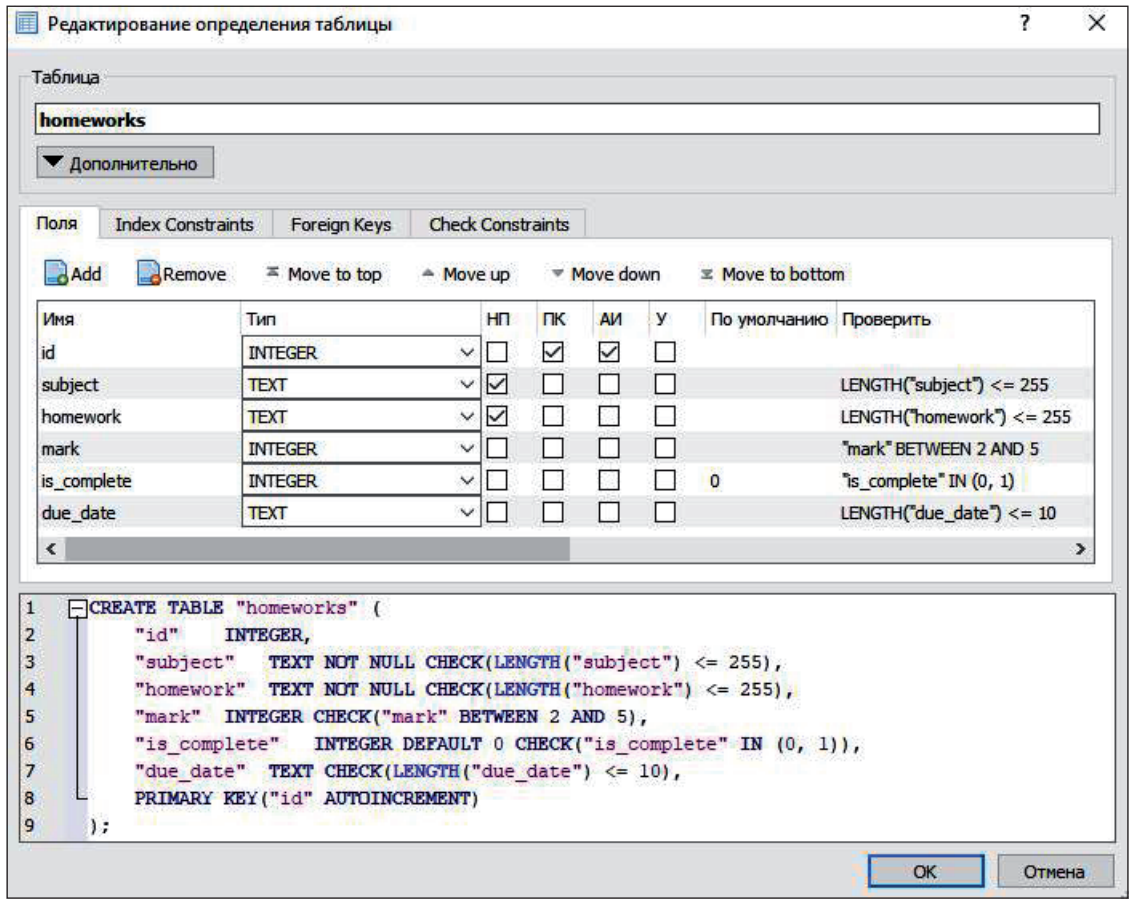

Рис. 1. Окно ***Редактирование определения таблицы***

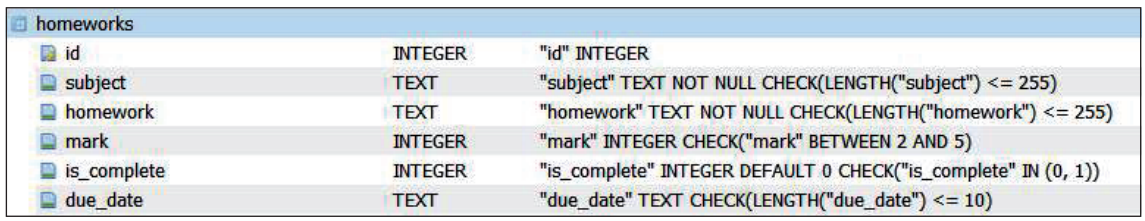

Рис. 2. Схема таблицы «Домашние работы (homeworks)»

ISSN 2221-1993 • **Информатика в школе** • 2026 • Том 25 № 2

Для самопроверки на рисунках 4, 5 приведены два запроса к базе данных с результатами их выполнения.

Вариант интерфейса БД-приложения изображен на рисунке 6. Далее опишем создание макета приложения по шагам с комментариями.

Шаг 4.

После работы с базой данных откройте IDLE (встроенный редактор кода для Python). В разделе меню *File* выберите *New File*, затем сохраните этот файл (команда *Save as*) под именем *main.py* в вашей папке проекта (*homework-app*).

	id	subject	homework	mark	is_complete	due_date
	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	Математика	Упр. 243, 245 (а, б), 250	NULL	0	16.03.2026
2	2	Русский язык	Подготовка к диктанту	4	1	10.03.2026
3	3	Информатика	Параграф 7, ответить на …	5	1	12.03.2026
4	4	Русский язык	Упр. 15	NULL	0	18.03.2026

Рис. 3. Заполненная таблица «Домашние работы (homeworks)»

SQL 1*

```
SELECT subject, homework, due_date
FROM homeworks
WHERE is_complete = 0
ORDER BY due_date ASC
```

	subject	homework	due_date
1	Математика	Упр. 243, 245 (а, б), 250	16.03.2026
2	Русский язык	Упр. 15	18.03.2026

Рис. 4. Результат SQL-запроса: вывод невыполненных домашних работ

SQL 1*

```
SELECT
IIF(is_complete = 1, 'Выполнено', 'Не выполнено') as status,
COUNT(*) as homeworks_count
FROM homeworks
GROUP BY is_complete
```

	status	homeworks_count
1	Не выполнено	2
2	Выполнено	2

Рис. 5. Результат SQL-запроса: вывод количества домашних работ со статусом

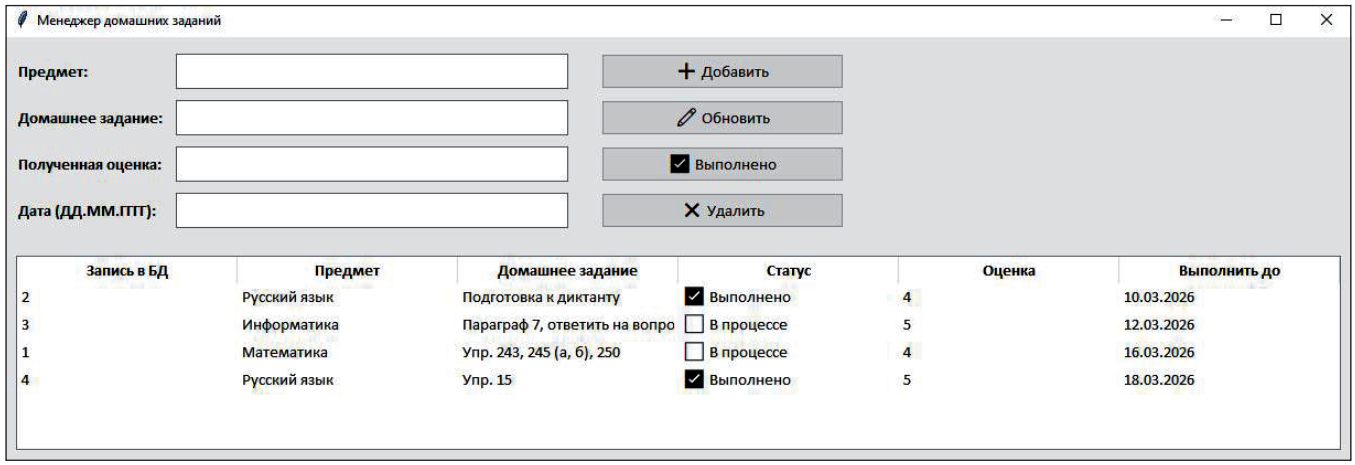

Рис. 6. БД-приложение «Менеджер домашних заданий», реализация на Python

ISSN 2221-1993 • **Informatics in School** • 2026 • Volume 25 No 2

Опишем подключение модулей и вспомогательных функций (листинг 1).

Вы можете проверить работу функций: напишите следующий фрагмент кода:

```
print(get_data('select * from homeworks'))
```

и запустите *main.py*. В качестве результата будет выведен список кортежей ваших записей БД.

Лабораторная работа построена в функциональном стиле для удобства вызова и повторного использования кода. В листинге 2 представлены функции-заглушки, которые будут наполнены программным кодом в следующих шагах работы. Скопируйте их в свой модуль.

Программный код (листинг 3) включает инициализацию класса *Tk* для работы с графической библиотекой *Tkinter*, для этого создается экземпляр класса (*root*) и указываются базовые настройки (размер колонок, стили).

Главным для запуска окна является вызов метода *root.mainloop*() в конце функции, без него приложение не будет отображаться на экране пользователя.

При запуске модуля *main* (*Run*, *Run Module*) на экране появится окно приложения с заголовком «Менеджер домашних заданий» в верхней части и пустой областью снизу.

Опишем инициализацию таблицы и функцию получения данных из базы данных (листинг 4).

`# Импорты модулей`
`import sqlite3`
 	
`from tkinter import Tk`
`from tkinter import ttk`
`from tkinter import font`
`from tkinter import messagebox`
 	
`DB = 'homework_manager.db' # файл базы данных`
 	
`FIELDS = {`
`"id": "Запись в БД",`
`"subject": "Предмет",`
`"homework": "Домашнее задание",`
`"status": "Статус",`
`"mark": "Оценка",`
`"due_date": "Выполнить до"`
`} # словарь полей таблицы`
	Фрагмент кода содержит перечисление встроенных модулей Python и вспомогательные константы

`# Вспомогательные функции для работы с БД`
`def query(query, params=()):`
`"""Для запросов без возврата данных (INSERT, UPDATE, DELETE)"""`
`with sqlite3.connect(DB) as conn:`
`cursor = conn.cursor()`
`cursor.execute(query, params)`
 	
`def get_data(query, params=(), many = True):`
`"""Для запросов с возвратом данных (SELECT)"""`
`with sqlite3.connect(DB) as conn:`
`cursor = conn.cursor()`
`cursor.execute(query, params)`
`data = cursor.fetchall() if many else cursor.fetchone()`
`return data`
	Колонка содержит функции для работы с запросами к БД, их допустимо напрямую скопировать в свой модуль *main.py*, они пригодятся в дальнейшей работе
	
	Используется Python-конструкция *with*, которая завершает соединение с БД после выполнения запроса
	
	Переменной *cursor* присвоен курсор БД *conn.cursor()* для выполнения запросов в модуле

Листинг 1. Подключение модулей Python и вспомогательных функций

```
def add_homework(table, subject_entry, homework_entry, mark_entry, date_entry):
    """Добавление новой записи"""
    pass

def update_homework(table, subject_entry, homework_entry, mark_entry, date_entry):
    """Обновление существующей записи"""
    pass

def toggle_complete(table):
    """Отметка задания как выполненное/невыполненное"""
    pass

def delete_homework(table):
    """Удаление существующей записи"""
    pass

def refresh_table(table):
    """Получение или обновление данных в таблице"""
    pass
```

Листинг 2. Список применяемых функций

ISSN 2221-1993 • **Информатика в школе** • 2026 • Том 25 № 2

```
def clean_entries(*entries):
    """Очистка любого количества полей ввода"""
    pass

def init_table(parent):
    """Создание таблицы"""
    pass

def clean_on_click(event, *entries):
    """Очистка полей ввода по правому клику мыши"""
    pass

def select_on_dbl_click(event, table, subject_entry, homework_entry, mark_entry, date_entry):
    """Заполнение полей при выборе записи (с прямым запросом к БД)"""
    pass

def create_input_frame(parent, label_text, row, column):
    """Создание поля ввода с заголовком"""
    pass

def main():
    """Функция-оркестр, которая собирает остальные функции в единую точку"""
    pass

if __name__ == "__main__":
    """Точка входа"""
    main()
```

Листинг 2 (окончание). Список применяемых функций

```
def main():
    """Главная функция"""
    # Создание окна
    root = Tk()
    root.title('Менеджер домашних заданий')

    root.grid_rowconfigure(1, weight=1)
    root.grid_columnconfigure(1, weight=1)

    # Настройка стилей (базовый шрифт и его размер)
    style = ttk.Style()

    default_font = font.nametofont("TkTextFont")
    font_family, font_size = "Calibri", 11
    default_font.configure(family=font_family, size=font_size)

    # Применяем стиль к компонентам - кнопкам, заголовкам, таблице
    style.configure('TButton', font=(font_family, font_size), width=30)
    style.configure('TLabel', font=(font_family, font_size, "bold"))
    style.configure("Treeview.Heading", font=(font_family, font_size, "bold"))
    style.configure("Treeview", font=(font_family, font_size), rowheight=25)

    """
    В эту часть функции будем добавлять описание интерфейса БД-приложения - контейнеры, таблицу, кнопки
    """

    # Запуск окна
    root.mainloop()
```

Листинг 3. Описание главной функции БД-приложения

```
def init_table(parent):
    """Создание таблицы"""
    table = ttk.Treeview(parent, columns=list(FIELDS.keys()),
show="headings")

    # Настройка заголовков
    for key, values in FIELDS.items():
        table.heading(key, text=values)

    return table
```

Функция *init_table*() возвращает экземпляр класса *Treeview*, который в *Tkinter* представляет виджет для отображения массива данных. Для того чтобы задать заголовки полей БД, используется константа *FIELDS*. Параметр *show* со значением *«headings»* определяет виджет как таблицу

Листинг 4. Функция инициализации таблицы и запроса к базе данных

ISSN 2221-1993 • **Informatics in School** • 2026 • Volume 25 No 2

```
def refresh_table(table):
    """Получение или обновление данных в таблице"""
    # Очистка таблицы
    for row in table.get_children():
        table.delete(row)

    # Получение данных из БД
    rows = get_data('''
                SELECT id, subject, homework,
                    CASE WHEN is_complete = 1
                           THEN '☑ Выполнено'
                           ELSE '☐ В процессе' END as status,
                    CASE WHEN mark is NULL
                           THEN ''
                           ELSE mark END as mark,
                    due_date
                FROM homeworks
                ORDER BY due_date ''')

    # Заполнение таблицы
    for row in rows:
        table.insert("", 'end', values=row)
```

Функция *refresh_table*() получает данные из БД и обновляет при операциях с ними. В функции описываются очистка таблицы, чтобы избежать случайных записей, запрос к базе данных (*SELECT*) и заполнение таблицы актуальными данными

Обращение к функциям осуществляется внутри функции *main*() в той части кода, где был указан комментарий о добавлении интерфейса (см. листинг 3)

Для позиционирования виджетов используется контейнер (виджет *Frame*), который привязывается к базовому окну *root*. В *init_table*() передается контейнер таблицы для отображения таблицы внутри виджета, а не основного окна

Листинг 4 (окончание). Функция инициализации таблицы и запроса к базе данных

```
    # Контейнер таблицы
    table_frame = ttk.Frame(root)
    table_frame.grid(row=1, column=0, sticky='nsew', padx=10, pady=10)
    table_frame.grid_rowconfigure(0, weight=1)
    table_frame.grid_columnconfigure(0, weight=1)

    # Создание таблицы
    table = init_table(table_frame)
    table.grid(row=0, column=0, sticky='nsew')

    """
    В эту часть функции будем добавлять интерфейс БД-приложения
    """

    # Загрузка данных
    refresh_table(table)

    # Запуск окна
    root.mainloop()
```

Листинг 5. Вызов функций init_table() и refresh_table() внутри функции main()

В качестве входного параметра в *refresh_table*() передается экземпляр *table* для манипуляции виджетом внутри функции. Функция вызывается ближе к концу программного кода для избежания возможных ошибок (листинг 5).

После запуска модуля в окно будет отображаться таблица с заполненными данными из базы данных (рис. 7).

Представление БД-приложения готово — мы подключили базу данных с языком программирования и вывели основную информацию через графическую библиотеку *Tkinter*.

Для завершения работы добавим поля ввода и функциональные кнопки, позволяющие осуществлять манипуляции с данными — добавление новых записей, обновление и удаление.

После фрагмента кода, где добавлена таблица, сделаем сноску на новую строку и добавим следующую часть кода (листинг 6).

Рассмотрим функцию *create_input_frame* (листинг 7), позволяющую объединить виджеты *label* (заголовок) и *entry* (поле ввода) через виджет *Frame* (контейнер).

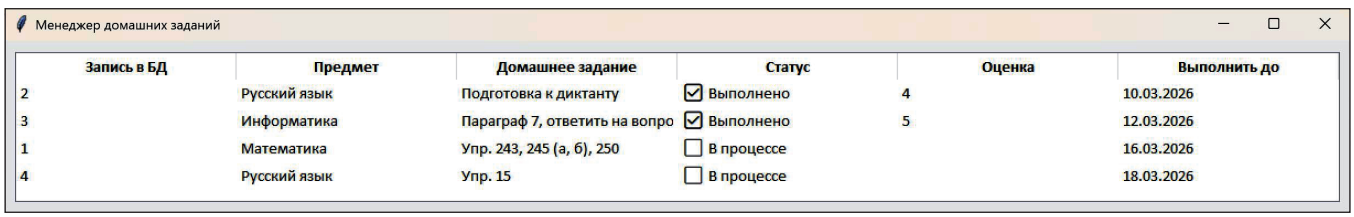

Рис. 7. Таблица домашних работ в БД-приложении

ISSN 2221-1993 • **Информатика в школе** • 2026 • Том 25 № 2

```
    # Контейнер интерфейса (поля и кнопки)
    main_frame = ttk.Frame(root)
    main_frame.grid(row=0, column=0, sticky='nsew')

    # Контейнер полей
    entries_frame = ttk.Frame(main_frame)
    entries_frame.grid(row=0, column=0, sticky='ew', padx=10, pady=10)

    # Поля для ввода данных
    subject_entry = create_input_frame(entries_frame, "Предмет:", 0, 0)
    homework_entry = create_input_frame(entries_frame, "Домашнее задание:", 1, 0)
    mark_entry = create_input_frame(entries_frame, "Полученная оценка:", 2, 0)
    date_entry = create_input_frame(entries_frame, "Дата (ДД.ММ.ГГГГ):", 3, 0)
```

В данном фрагменте описываются контейнеры «функциональной» части приложения, т. е. той, с которой может взаимодействовать пользователь

Для полей ввода созданы отдельный контейнер (*entries_frame*) и функция, которая позиционирует форму ввода в окне вместе с заголовком

Листинг 6. Описание контейнера интерфейса и полей, добавление полей ввода

```
def create_input_frame(parent, label_text, row, column):
    """Создание поля ввода с заголовком"""
    frame = ttk.Frame(parent)
    frame.grid(row=row, column=column, sticky='ew')

    frame.columnconfigure(0, weight=0)
    frame.columnconfigure(1, weight=1)

    label = ttk.Label(frame, text=label_text, anchor='w')
    label.grid(row=0, column=0, ipady=2, pady=5)

    entry = ttk.Entry(frame, width=50)
    entry.grid(row=0, column=1, sticky='e', ipady=3.5, pady=5, padx=10)

    return entry
```

В программном коде представлено создание контейнера *Frame* и пары «заголовок — поля ввода»

Результатом выполнения является возврат *entry* для последующей работы с вводимыми данными

Листинг 7. Описание функции create_input_frame()

Добавим кнопки для действий с данными базы данных в функцию *main* (листинг 8).

Все кнопки имеют общий стиль, описанный в листинге 3; дополнительно включены параметры: *text*, *cursor*, *command*, обеспечивающие действия по кнопке.

Далее представлены функции для работы с БД:

- добавление (*add_homework*);
- обновление (*update_homework*);
- изменение статуса домашней работы (*toggle_complete*);
- удаление домашней работы (*delete_homework*).

Рассмотрим каждую функцию по отдельности.

Функция добавления данных представлена в листинге 9.

Обновление данных осуществляется через функцию *update_homework* (листинг 10).

Рассмотрим функцию *toggle_complete*(), которая обновляет статус домашней работы для выбранной записи (листинг 11).

Для удаления записи из БД используется функция *delete_homework*() (листинг 12).

В листинге 13 представлены две вспомогательные функции — *clean_on_click*() и *on_dbl_click*() — для работы с таблицей и «динамики» БД-приложения.

```
    # Контейнер кнопок
    btn_frame = ttk.Frame(main_frame)
    btn_frame.grid(row=0, column=1, sticky='ew')

    add_btn = ttk.Button(btn_frame, text="➕ Добавить", cursor="hand2",
                         command=lambda: add_homework(table, subject_entry, homework_entry, mark_entry, date_entry))
    add_btn.grid(row=0, column=0, ipady=2, pady=5, padx=10)

    update_btn = ttk.Button(btn_frame, text="✎ Обновить", cursor="hand2",
                            command=lambda: update_homework(table, subject_entry, homework_entry, mark_entry, date_entry))
    update_btn.grid(row=1, column=0, ipady=2, pady=5, padx=10)

    complete_btn = ttk.Button(btn_frame, text="☑ Выполнено", cursor="hand2",
                              command=lambda: toggle_complete(table))
    complete_btn.grid(row=2, column=0, ipady=2, pady=5, padx=10)

    delete_btn = ttk.Button(btn_frame, text="✖ Удалить", cursor="hand2",
                            command=lambda: delete_homework(table))
    delete_btn.grid(row=3, column=0, ipady=2, pady=5, padx=10)
```

Листинг 8. Привязка виджетов кнопок к соответствующим функциям

```
def add_homework(table, subject_entry, homework_entry, mark_entry, date_entry):
    """Добавление новой записи"""
    subject = subject_entry.get().strip()
    homework = homework_entry.get().strip()
    mark = mark_entry.get().strip()
    due_date = date_entry.get().strip()

    if subject and homework:
        query(''' INSERT INTO homeworks (subject, homework, mark, due_date)
            VALUES (?, ?, ?, ?) ''', (subject, homework, mark, due_date))

        # Сообщение об успехе и обновление данных в таблице
        messagebox.showinfo("Успех", "Запись добавлена!")

        # Очищаем поля
        clean_entries(subject_entry, homework_entry, mark_entry, date_entry)
        refresh_table(table)
    else:
        messagebox.showwarning("Внимание", "Заполните предмет и задание!")
        return
```

Приведенный код обеспечивает добавление данных в БД: функция получает входные параметры в виде таблицы и полей ввода, далее обрабатывается значение каждого поля через метод *get*(). Метод *strip*() удаляет лишние пробелы

При отсутствии ключевых полей запрос не будет отработан

После выполнения запроса пользователь получает уведомление

Листинг 9. Функция добавления записи в БД

```
def update_homework(table, subject_entry, homework_entry, mark_entry, date_entry):
    """Обновление существующей записи"""
    selected = table.selection()

    if not selected:
        messagebox.showwarning("Внимание", "Выберите запись для обновления!")
        return

    # Получаем данные выбранной записи
    item = table.item(selected[0])
    values = item['values']
    hw_id = values[0] #ID записи

    subject = subject_entry.get().strip()
    homework = homework_entry.get().strip()
    mark = mark_entry.get().strip()
    due_date = date_entry.get().strip()

    if subject and homework:
        query(''' UPDATE homeworks SET subject = ?, homework = ?,
        mark = ?, due_date = ? WHERE id = ?
        ''', (subject, homework, mark, due_date, hw_id))

        messagebox.showinfo("Успех", "Запись обновлена!")

        # Очищаем поля
        clean_entries(subject_entry, homework_entry, mark_entry, date_entry)
        refresh_table(table)
    else:
        messagebox.showwarning("Внимание", "Заполните предмет и задание!")
```

Изменение данных выполняем по схожему алгоритму

Для получения текущей записи используется связка методов *item*() и *selection*() класса *Treeview*

При отсутствии ключевых полей запрос не будет отработан

После выполнения запроса пользователь получает уведомление

Листинг 10. Функция обновления записи в базе данных

```
def toggle_complete(table):
    """Отметка задания как выполненное/невыполненное"""
    selected = table.selection()

    if not selected:
        messagebox.showerror("Ошибка", "Запись не выбрана!")
        return

    item = table.item(selected[0])
    values = item['values']
    hw_id = values[0] #ID записи

    current_status = get_data('SELECT is_complete FROM homeworks
      WHERE id = ?', (hw_id,), many=False)[0]
    new_status = 0 if current_status == 1 else 1
```

В качестве параметра функции используется таблица

Далее запросом *SELECT* определяется текущий статус домашней работы и меняется на противоположный

После того как получили актуальное значение статуса, меняем его на противоположный (переменная *new_status*) и обновляем запись в БД через оператор *UPDATE*

Листинг 11. Функция toggle_complete() для изменения статуса домашней работы

```
    query('UPDATE homeworks SET is_complete = ? WHERE id = ?', (new_status, hw_id))

    status_text = "выполнено" if new_status == 1 else "активно"
    messagebox.showinfo("Успех", f"Задание отмечено как {status_text}!")

    # Обновляем таблицу
    refresh_table(table)
```

Листинг 11 (окончание). Функция toggle_complete() для изменения статуса домашней работы

```
def delete_homework(table):
    """Удаление существующей записи"""
    selected = table.selection()

    if not selected:
        messagebox.showerror("Ошибка", "Запись не выбрана!")
        return

    # Получаем данные выбранной записи
    item = table.item(selected[0])
    values = item['values']
    hw_id = values[0] #ID записи

    # Запрашиваем подтверждение
    confirm = messagebox.askyesno(
        "Подтверждение удаления",
        f"Вы действительно хотите удалить задание?\n\n"
        f"Это действие нельзя отменить!")

    if confirm: #Если подтвердили удаление
        query('DELETE FROM homeworks WHERE id = ?', (hw_id,))
        messagebox.showinfo("Успех", "Запись успешно удалена!")

        # Обновляем таблицу
        refresh_table(table)
```

На вход функции передается таблица, из которой извлекается идентификатор текущей записи

Если запись выбрана, у пользователя запрашивается подтверждение на удаление

В случае положительного ответа запись удаляется

Листинг 12. Удаление записи из базы данных

```
def clean_on_click(event, *entries):
    """Очищает поля ввода по правому клику мыши"""
    clean_entries(*entries)

def on_dbl_click(event, table, subject_entry, homework_entry, mark_entry, date_entry):
    """Заполнение полей при выборе записи (с прямым запросом к БД)"""
    selected = table.selection()

    if selected:
        item = table.item(selected[0])
        values = item['values']
        hw_id = values[0] #ID записи

        data = get_data('SELECT subject, homework, mark, due_date FROM homeworks
          WHERE id = ?', (hw_id,), many=False)

        if data:
            subject, homework, mark, due_date = data
            clean_entries(subject_entry, homework_entry, mark_entry, date_entry)

            subject_entry.insert(0, subject)
            homework_entry.insert(0, homework)
            mark_entry.insert(0, mark if mark else "")
            date_entry.insert(0, due_date if due_date else "")
```

Функция *clean_on_click*() очищает все поля ввода

Функция *on_dbl_click*() передает в поля ввода информацию, полученную с записи, на которую пользователь нажал двойным щелчком мыши

Листинг 13. Вспомогательные функции для работы с таблицей

Добавим в функцию *main*() программный код из листинга 14, это необходимо для того, чтобы «привязать» действия с таблицей к соответствующим функциям.

Обратите внимание: *table.bind* указывается перед *root.mainloop*(); убедитесь, что у вас не дублируется часть кода.

Работа над БД-приложением закончена, итоговый вариант интерфейса представлен на рисунке 8.

4. Заключение

Многолетний опыт обучения бакалавров — будущих учителей информатики по предложенной методике показывает, что у студентов формируются знания и умения по проектированию и созданию многотабличных баз данных, составлению запросов, включающих многосоставные логические условия, в том числе с использова-

ISSN 2221-1993 • **Informatics in School** • 2026 • Volume 25 No 2

```
    # Привязываем события выбора по двойному щелчку на записи
    table.bind('<Double-1>',
               lambda event: on_dbl_click(event, table, subject_entry, homework_entry, mark_entry, date_entry))

    table.bind('<Button-3>',
               lambda event: clean_on_click(event, subject_entry, homework_entry, mark_entry, date_entry))

    # Запуск окна
    root.mainloop()
```

Листинг 14. Привязка «кликов» по таблице

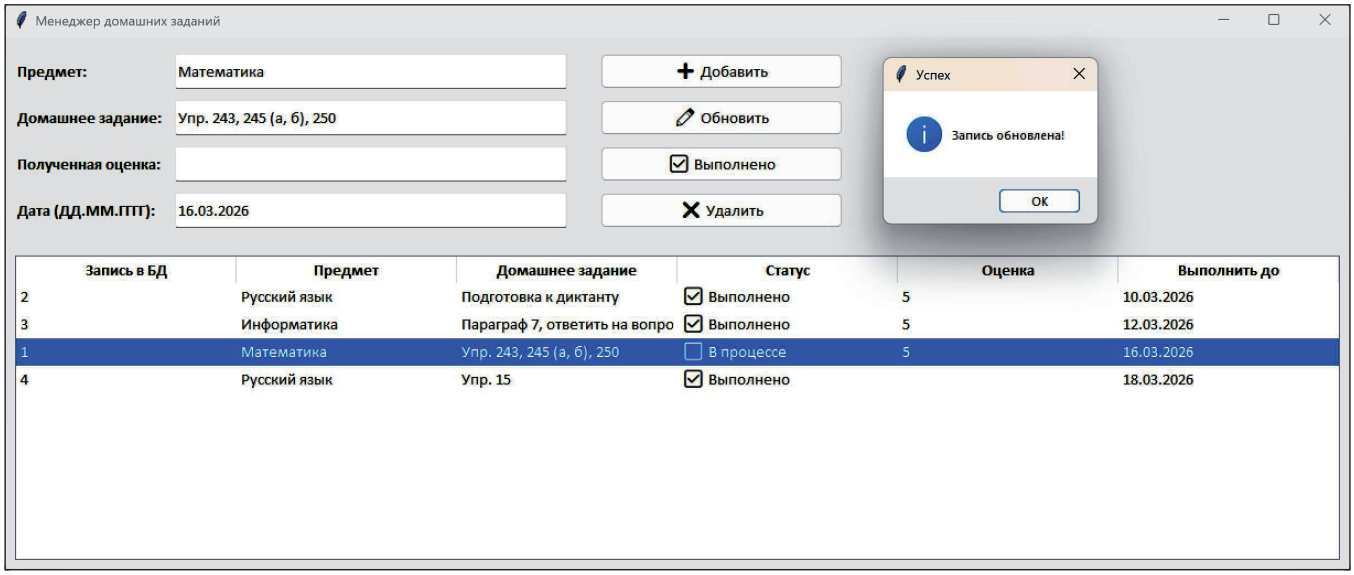

Рис. 8. Финальная версия БД-приложения с демонстрацией обновления записи

нием функций агрегаций. Студенты успешно осваивают создание учебных БД-приложений (демонстрационных примеров) на различных языках программирования (Object Pascal, Python). Отметим, что студенты также использовали эту методику для обучения учащихся профильных классов во время производственной учебной практики в школе.

Финансирование

Исследование выполнено за счет внутреннего гранта РГПУ им. А. И. Герцена (проект № 77-ВГ).

Список источников

1. *Абдулвелеева Р. Р., Зубарева Я. А., Абдулвелеев Р. И.* Диагностика результатов обучения по теме «База данных» // Современные тенденции развития и перспективы внедрения инновационных технологий в машиностроении, образовании и экономике. 2016. Т. 2. № 1. С. 328–332. EDN: WXTYKL.

2. *Баранова Е. В., Симонова И. В.* Развитие профессиональных компетенций бакалавров по направлению педагогического образования в области информатики в условиях цифрового образования // Известия Российского государственного педагогического университета им. А. И. Герцена. 2018. № 190. С. 116–124. EDN: PKYBWW.

3. *Беленкевский Д. С., Симонова И. В.* Модель содержания обучения и организация проектной деятельности студентов педагогического образования при освоении технологий разработки баз данных с использованием языка Python // Региональная информатика (РИ-2024). Материалы XIX Санкт-Петербургской международной конференции (Санкт-Петербург, 23–25 октября 2024 года). СПб.: Санкт-Петербургское Общество информатики, вычислительной техники, систем связи и управления, 2024. С. 236–238. EDN: KSALAV.

4. *Блудов И. В.* Особенности табличных выражений SQL и их соответствие с концепциями реляционной модели данных // Труды Института системного программирования РАН. 2013. Т. 24. С. 417–436. EDN: PIPJYJ.

5. *Васюков О. Г.* Сравнительный анализ современных СУБД, распространенных на предприятиях Самары с целью выбора СУБД для изучения в высшей школе по отдельной дисциплине // Международный научно-исследовательский журнал. 2014. № 6-2 (25). С. 47–50. EDN: SHOTCJ.

6. *Вдович С. А., Панова Н. Ф.* Выбор программного средства при обучении технологии баз данных // Тенденции развития науки и образования. 2021. № 70-1. С. 46–49. DOI: 10.18411/lj-02-2021-11. EDN: CBEXFB.

7. *Газуль С. М.* Базы данных: основы реляционной модели данных и СУБД MariaDB: учебное пособие. СПб.: Санкт-Петербургский государственный экономический университет, 2024. 77 с. EDN: IJWPLU.

8. *Ергалиев Е. Н.* Использование проектно-исследовательского метода при обучении студентов технического вуза базам данных // Universum: психология и образование. 2017. № 1 (31). С. 12–16. EDN: XIJZRR.

9. *Жалолов О. И., Хаятов Х. У.* Понятие SQL и реляционной базы данных // Universum: технические науки. 2020. № 6-1 (75). С. 26–29. EDN: IKQGCE.

10. *Левченко И. В., Садыкова А. Р., Абушкин Д. Б., Карташова Л. И., Кондратьева В. А., Моисеев В. П.* Особенности подготовки по программированию будущих учителей информатики // Вестник Российского университета дружбы народов. Серия: Информатизация образования. 2021. Т. 18. № 4. С. 337–346. DOI: 10.22363/2312-8631-2021-18-4-337-346. EDN: LRQEZJ.

11. *Лысенко А. О.* Общие подходы по изучению темы: базы данных // Экономика и социум. 2015. № 1-3 (14). С. 1004–1009. EDN: UBHBMN.

ISSN 2221-1993 • **Информатика в школе** • 2026 • Том 25 № 2

12. *Пирогов В. Ю.* Некоторые особенности преподавания языка управления базами данных // Мир науки. Педагогика и психология. 2018. Т. 6. № 6. С. 1–9. EDN: VVHXYS.

13. Приказ Минобрнауки России от 17.05.2012 № 413 (ред. от 12.02.2025) «Об утверждении федерального государственного образовательного стандарта среднего общего образования» (Зарегистрировано в Минюсте России 07.06.2012 № 24480). https://mrnvmu.mil.ru/upload/site133/document_file/RnuqAW77ML.pdf

14. *Саидов Ж. Д. Ў. Л.* Компетентностный подход при обучении работе с базами данных в системе высшего образования // Проблемы современного образования. 2022. № 6. С. 253–265. DOI: 10.31862/2218-8711-2022-6-253-265. EDN: AOBUGK.