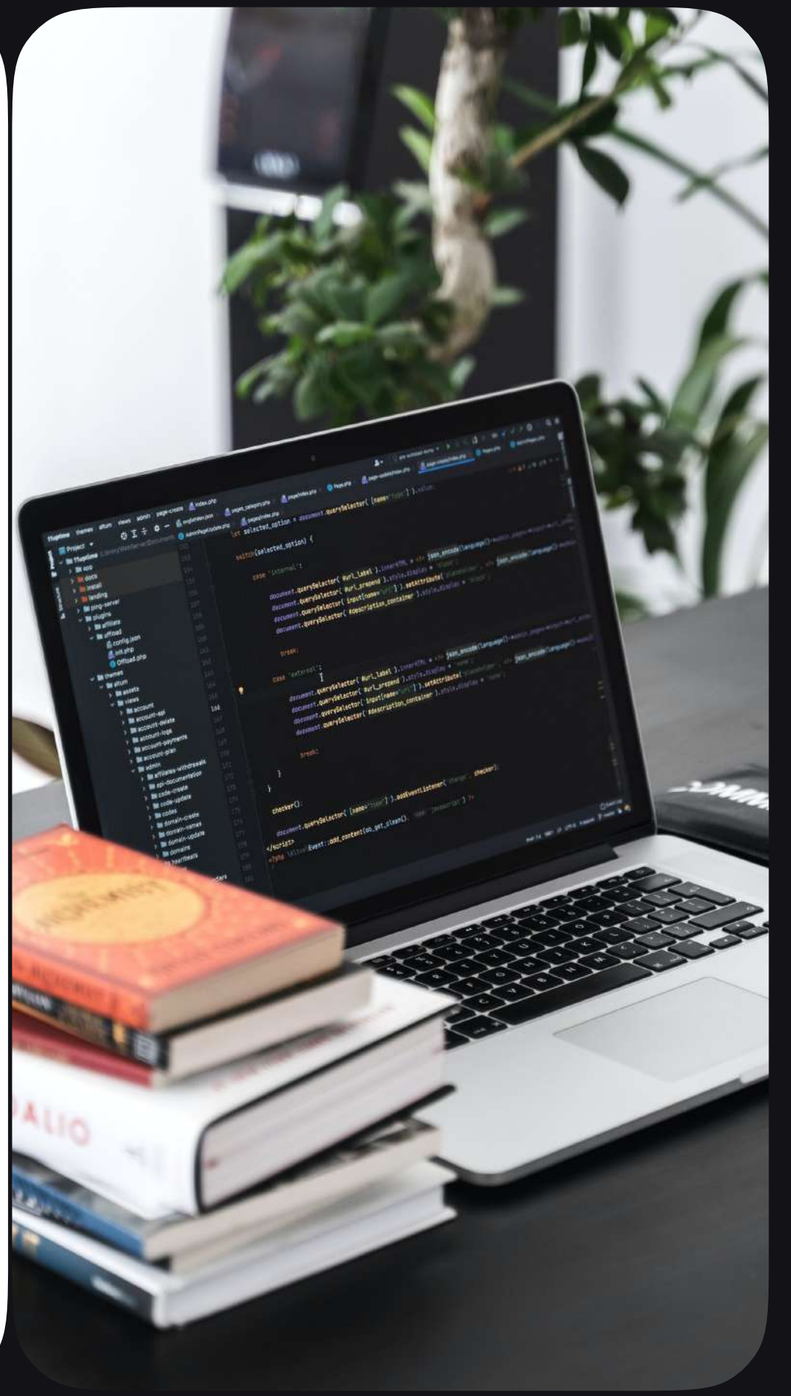
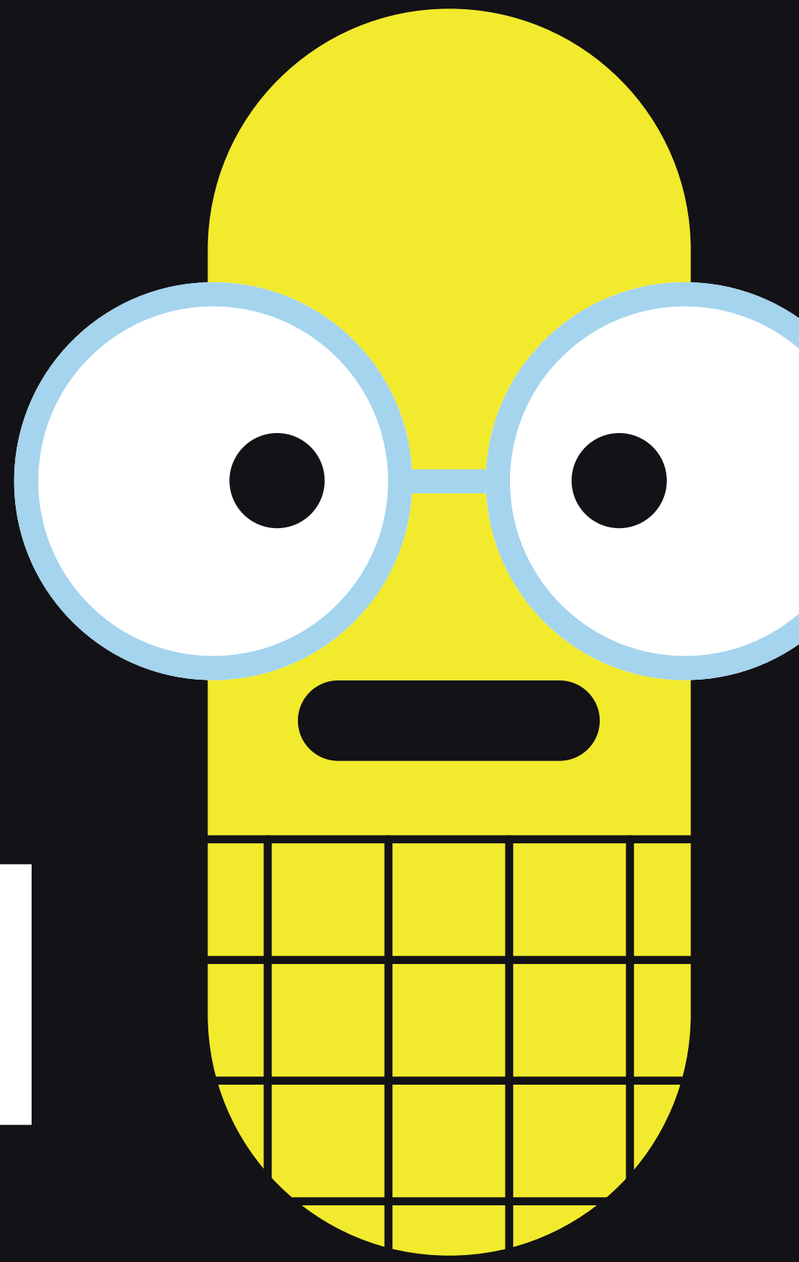
Занятие 7

Работа с БД. Таблицы в PyQT



Теория



Подключение PyQt к базе данных SQL

Подключение приложения к связующей базе данных и получение приложения для создания, чтения, обновления и удаления данных, хранящихся в этой базе данных, является обычной задачей в программировании.

- Связующие базы данных обычно организованы в набор таблиц или **отношений**.
- Данная строка в таблице называется **записью** или **кортежем**, а **столбец** — **атрибутом**.

Создание соединения с базой данных

Подключение ваших приложений к физической базе данных SQL — важный шаг в процессе разработки приложений баз данных с помощью PyQt.

Для успешного выполнения этого шага потребуется общая информация о том, как настроена ваша база данных.

SQLite 3

SQLite 3 — хорошо протестированная система баз данных с поддержкой всех платформ и минимальными требованиями к конфигурации.

SQLite позволяет вам читать и писать непосредственно в базы данных на вашем локальном диске без необходимости в отдельном серверном процессе. Это делает его удобным вариантом для обучения разработке приложений для баз данных.

Еще одним преимуществом использования **SQLite** является то, что библиотека поставляется с **Python**, а также с **PyQt**, поэтому вам не нужно ничего устанавливать, чтобы начать с ней работать.

В **PyQt** вы можете создать соединение с базой данных с помощью **QSqlDatabase** класса. Этот класс представляет соединение и предоставляет интерфейс для доступа к базе данных.

Чтобы создать соединение, просто добавьте **.addDatabase()** на **QSqlDatabase**. Этот статический метод принимает драйвер SQL и необязательное имя соединения в качестве аргументов и возвращает соединение с базой данных:

```
QSqlDatabase.addDatabase(
    driver, connectionName=QSqlDatabase.defaultConnection
)
```

```
QSqlDatabase.addDatabase(
    driver, connectionName=QSqlDatabase.defaultConnection
)
```

Первый аргумент, драйвер, является обязательным аргументом, который содержит строку, содержащую имя драйвера SQL, поддерживаемого PyQt.

Второй аргумент, **connectionName**, является необязательным аргументом, который содержит строку с именем соединения. Имя соединения по умолчанию имеет значение **QSqlDatabase.DefaultConnection**, которое обычно содержит строку «**qt_sql_default_connection**».

Если у вас уже есть соединение с именем **connectionName**, то это соединение удаляется и заменяется новым соединением, а **.addDatabase()** возвращает только что добавленное соединение с базой данных обратно вызывающему объекту.

Вызов функции **.addDatabase()** добавляет соединение с базой данных в список доступных соединений. Этот список представляет собой глобальный реестр, который PyQt поддерживает за кулисами, чтобы отслеживать доступные соединения в приложении. Регистрация соединений с помощью значимого имени соединения позволит вам управлять несколькими соединениями в приложении базы данных.

Описание	Описание
`.setDatabaseName(name)`	Устанавливает имя базы данных в name, которое представляет собой строку, представляющую допустимое имя базы данных
`.setHostName(host)`	Устанавливает имя хоста в host, которое представляет собой строку, представляющую допустимое имя хоста
`.setHostName(host)`	Устанавливает имя пользователя в username, которое представляет собой строку, представляющую допустимое имя пользователя
`.setHostName(host)`	Устанавливает пароль в password, который представляет собой строку, представляющую действительный пароль

Чтобы создать соединение с базой данных SQLite с помощью QSqlDatabase, откройте интерактивный сеанс Python и введите следующий код:

```
>>> from PyQt5.QtSql import QSqlDatabase
>>> con = QSqlDatabase.addDatabase("QSQLITE")
>>> con.setDatabaseName("contacts.sqlite")

>>> con

>>> con.databaseName()
'contacts.sqlite'

>>> con.connectionName()
'qt_sql_default_connection'
```

Этот код создаст объект подключения к базе данных, используя «**QSQLITE**» в качестве драйвера соединения и «**contacts.sqlite**» в качестве имени базы данных соединения. Поскольку вы не передаете имя соединения в **.addDatabase()**, вновь созданное соединение становится вашим соединением по умолчанию, имя которого «**qt_sql_default_connection**».

В случае баз данных SQLite имя базы данных обычно является именем файла или путем, включающим имя файла базы данных.
Вы также можете использовать специальное имя «:memory:» для базы данных в памяти.

Использование различных SQL-драйверов

Имя	Система управления базами данных
QDB2	IBM Db2 (version 7.1 and above)
QIBASE	Borland InterBase
QMYSQL/MARIADB	MySQL or MariaDB (version 5.0 and above)
QOCI	Oracle Call Interface
QODBC	Open Database Connectivity (ODBC)
QPSQL	PostgreSQL (versions 7.3 and above)
QSQLITE2	SQLite 2 (obsolete since Qt 5.14)
QSQLITE	SQLite 3
QTDS	Sybase Adaptive Server (obsolete since Qt 4.7)

Библиотека предоставляет богатый набор драйверов SQL, которые позволяют использовать различные типы систем управления базами данных в соответствии с вашими конкретными потребностями.

Столбец **Имя** драйвера содержит строки идентификаторов, которые необходимо передать в .add Database() в качестве первого аргумента для использования связанного драйвера. В отличие от драйвера SQLite, при использовании другого драйвера вам может потребоваться установить несколько атрибутов, таких как databaseName, hostName, userName, и password, чтобы соединение работало правильно.

Открытие соединения с базой данных

Как только у нас есть подключение к базе данных, нужно открыть это соединение, чтобы иметь возможность взаимодействовать с нашей базой данных. Для этого надо вызвать функцию .open(), которая:

1. **.open()** открывает соединение с базой данных с использованием текущих значений соединения
2. **.open(username, password)** открывает соединение с базой данных с использованием предоставленных имени пользователя и пароля.

Оба варианта возвращают **True**, если соединение успешно. В противном случае они возвращают **False.**

Если соединение не удается установить, то можно позвонить **.lastError ()**, чтобы получить информацию о том, что произошло. Эта функция возвращает информацию о последней ошибке, сообщенной базой данных.

Выполнение SQL-запросов с помощью PyQt

Конструктор **QSqlQuery** имеет несколько вариаций, рассмотрим:

1. QSqlQuery(query, connection) создает объект запроса, используя строковый SQL-запрос и соединение с базой данных. Если вы не указали соединение или если указанное соединение является недопустимым, то используется соединение с базой данных по умолчанию. Если запрос не является пустой строкой, то он будет выполнен сразу же.
2. QSqlQuery(connection) создает объект запроса с помощью соединения. Если соединение недопустимо, то используется соединение по умолчанию.

Можно создавать объекты **QSqlQuery** без передачи каких-либо аргументов конструктору. В этом случае запрос будет использовать соединение с базой данных по умолчанию, если таковое имеется.

Чтобы выполнить запрос, нужно вызвать **.exec()** для объекта запроса. Рассмотрим использование **.exec()** двумя различными способами:

1. **.exec(query)** выполняет строковый SQL-запрос, содержащийся в запросе. Он возвращает **True**, если запрос был успешным, и в противном случае возвращает **False**.
2. **.exec()** выполняет ранее подготовленный SQL-запрос. Он возвращает **True**, если запрос был успешным, и в противном случае возвращает **False**.