

Реализация программного модуля асинхронной записи векторного слоя в базу данных PostGIS с помощью языка Python в системе QGIS

Вихляев Дмитрий Романович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье рассматривается создание модульной программы в среде QGIS для произведения записи векторного слоя объектов в базу данных PostgreSQL/PostGIS. Программа написана на языке программирования Python в среде QGIS и с поддержкой внутренних модулей системы. Результатом исследования станет готовая программа реализующая перенос геопространственных данных из слоя QGIS в таблицу базы данных PostGIS с использованием асинхронного подхода.

Ключевые слова: QGIS, асинхронное программирование, Python, векторные данные, атрибуты.

Implementation of the software module for inserting a vector layer into the PostGIS database using Python in the QGIS system

Vikhlyayev Dmitry Romanovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article discusses the creation of a modular program in the QGIS environment for writing a vector layer of objects to the PostgreSQL/PostGIS database. The program is written in the Python programming language in the QGIS environment and with support for the internal modules of the system. The result of the study will be a ready-made program that implements the transfer of geospatial data from the QGIS layer to the PostGIS c database table using an asynchronous approach.

Keywords: QGIS, asynchronous programming, Python, vector data, attributes.

1 Введение

1.1 Актуальность

Асинхронная запись векторных слоев из QGIS в PostGIS через Python обеспечивает высокую производительность за счет параллельной обработки данных без блокировки интерфейса пользователя. Это особенно эффективно при работе с большими объемами пространственной информации, где скорость и надежность записи критически важны. Модуль автоматически преобразует геометрии между форматами QGIS и PostGIS, поддерживает различные системы координат и предлагает гибкие настройки пакетной

загрузки. Ключевые сценарии использования включают миграцию крупных датасетов, синхронизацию данных в реальном времени, консолидацию результатов пространственного анализа и построение отказоустойчивых ETL-процессов для геоданных. Решение сочетает глубокую интеграцию с экосистемой QGIS/PostGIS, что позволяет объединить готовые инструменты геопропространственной системы и автоматизацию языка программирования.

1.2 Обзор исследований

Ю.С.Борисова в своей работе создала условные знаки в геоинформационной среде Qgis для изображения сетей коммуникаций на крупномасштабных топографических планах [1]. А.О.Белоусов разработал технологию формирования информационной базы о состоянии земель сельскохозяйственных организаций в геоинформационной системе Qgis [2]. И.Б.Гинзбург использовал автономное приложение при работе с данными дистанционного зондирования земли и векторными кадастровыми слоями [3]. Е.Н.Дремов изучил метод распознавания значений высот на векторных слоях геопропространственных данных [4]. И.Н.Беляева, А.В. Федоров создали векторный слой интерактивной карты из растрового изображения плана здания [5]. С.В.Смирнов, С.С.Наговицын, Ю.В.Русских спроектировали векторные слои для геоинформационных систем [6].

1.3 Цель исследования

Цель исследования – написать модуль реализующий фоновый перенос геопропространственных данных из слоя QGIS в таблицу базы данных PostGIS с полным сохранением геометрии и атрибутов.

2 Материалы и методы

Для реализации используется геоинформационная система Qgis 3.36.1 с встроенным интерпретатором Python. Подключение к удалённой базе данных Psotgis осуществляется по ip-адресу.

3 Результаты и обсуждения

В системе QGIS запуск python скрипта осуществляется синхронно в, том же потоке, что и пользовательский интерфейс. В связи, с чем при долгой операции скрипта, программа зависнет для пользователя. Чтобы предотвратить блокировку необходимо построить систему на классе QgsTask, наследуя функциональность фоновых задач QGIS.

Для начала нужно инициализировать в основном потоке все объекты, включая получение исходного векторного слоя и формирование строки подключения к базе данных. Эти операции обязательно выполняются до запуска фонового потока в соответствии с требованиями архитектуры QGIS (рис.1).

```

from datetime import datetime
from qgis.core import (QgsTask, QgsProject, QgsVectorLayer,
                       QgsFeature, QgsDataSourceUri, QgsApplication)

# Запуск
db_params = {
    'host': 'ip-address',
    'port': '5432',
    'database': 'name-db',
    'username': 'dblogin',
    'password': 'dbpass'
}

class ImportTask():
    def __init__(self, source_layer_name, db_params):
        super().__init__("Import to PostGIS", QgsTask.CanCancel)
        self.source_layer_name = source_layer_name
        self.db_params = db_params
        self.total_added = 0
        self.errors = []

        # Получаем данные в главном потоке ДО запуска задачи
        layers = QgsProject.instance().mapLayersByName(self.source_layer_name)
        if not layers:
            raise Exception(f"Layer {source_layer_name} not found")
        self.source_layer = layers[0]

        # Формируем URI заранее
        self.uri = QgsDataSourceUri()
        self.uri.setConnection(
            db_params['host'], db_params['port'],
            db_params['database'], db_params['username'], db_params['password']
        )
        self.uri.setDataSource('public', 'ReconstructionGrid', 'polys', '', 'id')

```

Рис. 1. Импорт и инициализация переменных

Важно отметить, что на данной фазе не осуществляется подключение к СУБД, а лишь подготавливаются параметры. После чего управление передаётся менеджеру задач QGIS для асинхронного запуска (рис.2).

```

try:
    task = ImportTask("gfs_grid", db_params)
    QgsApplication.taskManager().addTask(task)
except Exception as e:
    print(f"Не удалось запустить задачу: {e}")

```

Рис. 2. Пример запуска задачи

После активации задачи в фоновом режиме через запуск метода «run» устанавливается соединение с базой данных через PostGIS-провайдер. Все данные необходимо провести через валидаторы, чаще всего ошибки вызываются из-за несоответствия типа геометрии и системе координат.

Для каждого объекта проверяется корректность геометрии. Создаётся новый экземпляр записи с полями, соответствующими структуре целевой таблицы. Происходит копирование исходной геометрии.

Для примера выбраны поля внешнего ключа и времени создания и обновления записи. Поле reconstructionSourceId получает фиксированное значение. Поля createdAt и updatedAt автоматически устанавливаются в текущую временную метку.

Сформированные объекты накапливаются в буфере. Чтобы не вызывать большого скопления памяти, устанавливается значение порога равной 500 записей. Выполнение групповой операция вставки в базу данных происходит через одну транзакцию, оптимизируя расходы на сетевые вызовы и операции ввода-вывода. Полный пример метода представлен на рисунке 3.

```
def run(self):
    try:
        # Используем отдельный экземпляр слоя для провайдера в потоке
        target_layer = QgsVectorLayer(self.uri.uri(), 'target', 'postgres')
        if not target_layer.isValid():
            self.errors.append("Invalid target table")
            return False

        provider = target_layer.dataProvider()
        fields = target_layer.fields()

        BATCH_SIZE = 500
        features_buffer = []

        features_iterator = self.source_layer.getFeatures()
        count = self.source_layer.featureCount()

        for i, feature in enumerate(features_iterator):
            if self.isCanceled():
                return False
            geom = feature.geometry()
            if not geom or geom.isEmpty():
                continue
            new_feature = QgsFeature(fields)
            new_feature.setGeometry(geom)
            # Заполнение атрибутов
            field_names = fields.names()
            if 'reconstructionSourceId' in field_names:
                new_feature.setAttribute('reconstructionSourceId', 1)
            now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            if 'createdAt' in field_names:
                new_feature.setAttribute('createdAt', now)
            if 'updatedAt' in field_names:
                new_feature.setAttribute('updatedAt', now)

            features_buffer.append(new_feature)

            if len(features_buffer) >= BATCH_SIZE:
                success, added_features = provider.addFeatures(features_buffer)
                if not success:
                    self.errors.append(f"Batch insert failed near index {i}")
                    return False
                self.total_added += len(added_features)
                features_buffer = []

            self.setProgress(int((i / count) * 100))
        # Финальный пакет
        if features_buffer:
            success, added_features = provider.addFeatures(features_buffer)
            if success:
                self.total_added += len(added_features)
        return True
    except Exception as e:
        self.errors.append(str(e))
        return False
```

Рис. 3. Метод асинхронного запуска задачи QGIS python API

После каждой итерации обновляется индикатор прогресса, рассчитываемый как процент от общего числа объектов.

На всём протяжении выполнения непрерывно отслеживается флаг отмены операции. При его активации процесс немедленно прекращается с корректным высвобождением ресурсов (рис.4).

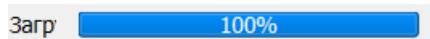


Рис. 4. Статус загрузки пользовательского интерфейса QGIS

После окончания фоновой обработки автоматически вызывается метод завершения. Он работает в контексте основного потока приложения, обеспечивая безопасный доступ к элементам интерфейса. При возникновении ошибок или отмене операции выводится отчёт, включающий перечень всех зафиксированных проблем с указанием этапа их возникновения (рис.5).

```
def finished(self, result):  
    if result:  
        print(f"Импорт завершен! Реально добавлено объектов: {self.total_added}")  
    else:  
        print(f"Ошибка импорта: {self.errors}")
```

Рис. 5. Определение нужных объектов и добавление в новый слой

На этой фазе также происходит неявное высвобождение выделенных ресурсов, закрытие соединений с БД, очистка буферов памяти, завершение работы с провайдерами данных. В конце отображается статус операции.

4 Выводы

В результате исследования была написана программа для вставки объектов геометрического векторного слоя объектов расположенных в пространстве памяти QGIS в базу данных PostGIS. Приведены примеры дополнения атрибутов и установки их значений в процессе записи. Описан способ асинхронного вызова задачи без блокировки пользовательского интерфейса.

Библиографический список

1. Борисова Ю.С. Создание условных знаков в геоинформационной среде Qgis для изображения сетей коммуникаций на крупномасштабных топографических планах // В сборнике: Актуальные вопросы землепользования и управления недвижимостью. Сборник статей III Национальной научно-практической конференции. Екатеринбург, 2021. С. 52-61.
2. Белоусов А.О. Технология формирования информационной базы о состоянии земель сельскохозяйственных организаций в геоинформационной системе Qgis // В сборнике: Рациональное использование земельных ресурсов в условиях современного развития АПК. Сборник материалов Всероссийской (национальной) научно-практической конференции. Тюмень, 2021. С. 15-21.
3. Гинзбург И.Б. Использование автономного веб-приложения при работе с данными дистанционного зондирования земли и векторными кадастровыми слоями // Альманах мировой науки. 2015. № 1-1 (1). С. 79-

80.

4. Дремов Е.Н. Метод распознавания значений высот на векторных слоях геопространственных данных // В сборнике: Оптико-электронные приборы и устройства в системах распознавания образов, обработки изображений и символьной информации. Распознавание - 2017. Сборник материалов XIII Международной научно-технической конференции. 2017. С. 134-136.
5. Беляева И.Н., Федоров А.В. Создание векторного слоя интерактивной карты из растрового изображения плана здания // В сборнике: Актуальные проблемы развития науки и современного образования. Сборник материалов Международной научно-практической конференции. 2017. С. 77-79.
6. Смирнов С.В., Наговицын С.С., Русских Ю.В. Проектирование векторных слоев как начальная стадия создания геоинформационных систем // Социально-экономическое управление: теория и практика. 2019. № 2 (37). С. 72-75.
7. Серебренников И.В. Построение архитектуры программного комплекса на основе Qgis // В сборнике: Наука, знания, интеллект. Сборник статей III Международного научно-исследовательского конкурса. Пенза, 2023. С. 47-50.