Прокуронова Анастасия Юрьевна
старший преподаватель
ФГБОУ ВО «Московский технологический университет»
г. Москва

АВТОМАТИЗАЦИЯ ГЕНЕРАЦИИ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ ПО ПРОЕКТИРОВАНИЮ БАЗ ДАННЫХ КАК ИНСТРУМЕНТ АДАПТИВНОГО ОБУЧЕНИЯ В ВУЗЕ (ПРОЕКТНАЯ СТАДИЯ)

Аннотация: *в статье представлен разрабатываемый подход к автоматической генерации индивидуальных заданий по проектированию баз данных с использованием языка Python. На данный момент создана экспериментальная версия генератора в виде набора скриптов, позволяющая генерировать уникальные варианты схем данных, текстов заданий и эталонных ответов. Описывается архитектура прототипа, его текущие возможности и ограничения. Акцент сделан на том, что проект находится в стадии активной разработки и не доведён до готового приложения (EXE-файла). Обсуждаются планируемые педагогические эффекты (снижение списывания, индивидуализация) при условии последующего внедрения. Отдельно рассматривается вопрос интеллектуальной собственности в вузовской среде, который частично объясняет незавершённость публичного релиза.*

Ключевые слова: *проектирование баз данных, генерация заданий, Python, прототип, адаптивное обучение, стадия разработки.*

Введение

Дисциплины, связанные с проектированием баз данных, традиционно требуют большого числа практических заданий. В условиях массового потока студентов преподаватели сталкиваются с двумя проблемами: сложностью подготовки уникальных вариантов для каждого обучающегося и высоким риском копирования готовых решений. Одним из способов решения является автоматическая генерация заданий.

В рамках собственной педагогической практики автором начата разработка инструмента на Python, который позволит создавать практически неограниченное число вариантов заданий по проектированию БД – от ER-диаграмм до SQL-скриптов. *На текущий момент проект находится в стадии прототипа*, не завершён и не внедрён в учебный процесс. Цель данной статьи – описать концепцию, текущее состояние разработки и планы по дальнейшему развитию, а также обсудить возможные педагогические эффекты, которые могут быть достигнуты при завершении проекта.

Особая оговорка: все наработки, создаваемые автором в рамках трудовой деятельности в вузе, могут признаваться интеллектуальной собственностью образовательной организации. Именно поэтому здесь представлен *описательный*, *а не финальный программный продукт* – завершение и публикация инструмента требуют согласования с администрацией.

1. Текущее состояние проекта.

На сегодняшний день разработана *экспериментальная версия генератора* в виде набора Python-скриптов. Инструмент не собран в EXE-файл, не имеет графического интерфейса и работает в консольном режиме или в среде Jupyter Notebook. Основные функции, которые уже реализованы или находятся в высокой степени готовности:

- генерация текстового описания предметной области (библиотека, магазин, склад и т. д.) со случайными параметрами;

- генерация логической схемы БД: сущности, атрибуты (с типами данных), первичные ключи, связи (один-ко-многим, многие-ко-многим);

- генерация эталонного DDL-скрипта (SQL) для выбранной схемы;

- вывод задания в удобном для печати формате (текстовый файл).

Чего пока нет в текущей версии:

- автоматической проверки студенческих решений (только генерация эталона для ручного сравнения);

- веб-интерфейса или исполняемого EXE-файла;

- интеграции с системами дистанционного обучения (LMS);

- надёжного генератора связей многие-ко-многим (реализован частично, требует доработки).

2. Архитектура и реализация прототипа.

Прототип построен на стандартной библиотеке Python (модули random, dataclasses, pathlib). Отсутствие внешних зависимостей сделано намеренно, чтобы в будущем упростить возможную сборку в EXE с помощью PyInstaller или Nuitka – но на текущий момент эта задача не ставилась.

2.1. Структура модулей.

- domain_generator.py – выбор случайного домена и генерация описания;
- schema_builder.py – построение логической схемы (сущности + атрибуты);
- task_formatter.py – формирование текста задания для студента;
- ddl_exporter.py – создание эталонного SQL-скрипта.

Все модули представляют собой отдельные.py-файлы, запускаемые через командную строку. Например, команда python main.py --complexity medium --output task_123.txt генерирует задание и сохраняет его в файл.

2.2. Пример генерируемого задания.

Вариант №7 (фрагмент):

Предметная область: «Склад запасных частей».Спроектировать базу данных для учёта деталей, поставщиков и поступлений. Сущности: detail (id, name, weight), supplier (id, name, address), delivery (detail_id, supplier_id, quantity, date).

Требования:

- построить ER-диаграмму;
- написать SQL-скрипт создания таблиц с ограничениями PRIMARY KEY, FOREIGN KEY, NOT NULL;
- нормализовать схему до 3НФ (исходная дана уже нормализованной или требует проверки – в зависимости от варианта).

2.3. Ограничения прототипа.

В текущей версии не генерируются:

- составные первичные ключи;
- индексы и представления;

- триггеры и хранимые процедуры.

Эти возможности запланированы на следующих этапах разработки. Также остаётся проблема «осмысленности» генерируемых доменов – случайно выбранные атрибуты иногда дают нелогичные сочетания (например, у детали – поле «цвет», но деталь техническая может его не иметь). Требуется доработка семантических правил.

3. Планируемое использование в учебном процессе (*гипотетически*).

Поскольку проект не завершён и не внедрён, ниже описывается *желаемый сценарий*, который может быть реализован после окончания разработки и решения вопросов с авторским правом вуза.

3.1. Интеграция с дисциплинами.

После приведения инструмента в рабочее состояние предполагается его использование в следующих курсах:

- «*Проектирование БД*» – генерация уникальных вариантов для лабораторных работ;

- «*Автоматизация и управление БД*» – на основе сгенерированной схемы студенты пишут триггеры и процедуры;

- «*Инструментальные средства ИС*» – студенты могли бы дорабатывать самого генератора (как учебное задание).

3.2. Ожидаемые педагогические эффекты.

Если проект будет доведён до конца, можно прогнозировать:

- снижение вероятности копирования заданий (каждый вариант уникален);

- экономию времени преподавателя на подготовку вариантов (генерация 30 вариантов занимает < 3 секунд);

- повышение мотивации студентов за счёт работы над «своим» кейсом.

Однако подчеркнём: *это лишь гипотезы*, так как экспериментальное внедрение не проводилось.

3.3. Вопрос готового ПО (*EXE-файл*).

Следует пояснить, почему генератор не представлен в виде EXE-файла. Причин несколько:

- техническая незавершённость – некоторые функции (например, проверка решений) пока не реализованы;

- политика вуза – исполняемый файл, созданный сотрудником в рабочее время, может считаться собственностью организации. Автор намерен сначала согласовать формат распространения;

- целесообразность – для внутреннего использования достаточно скриптов на Python, не требующих компиляции.

Таким образом, на данный момент инструмент существует в виде исходного кода и *не предназначен для тиражирования* за пределы разработчика.

3.4. Статус разработки: «*проект в дальнейшей разработке*».

Сознательное торможение завершения проекта связано с опасением, что полностью готовый продукт автоматически перейдёт в собственность вуза, ограничив возможность автора использовать его в дальнейшем или публиковать как свою разработку. Поэтому представленная статья описывает *концепцию и работающий прототип*, *но не финальное решение*. Это положение отражено в названии раздела и в заключении.

4. Перспективы и направления доработки.

Для перехода прототипа в стадию готового педагогического инструмента необходимо выполнить следующие шаги (в порядке приоритета):

- доработать генерацию связей многие-ко-многим с автоматическим созданием промежуточных таблиц;

- реализовать модуль автоматической проверки SQL-скриптов студента (сравнение структуры, использование sqlparse или кастомного парсера);

- создать простой графический интерфейс на Tkinter или веб-интерфейс на FastAPI;

- решить вопрос о сборке EXE (после согласования с вузом);

- провести пилотное апробирование на одной учебной группе (с разрешения кафедры).

Кроме того, перспективным видится использование больших языковых моделей (LLM) для генерации более реалистичных описаний предметной области по заданной схеме – это позволит сделать задания ещё интереснее для студентов.

Заключение

В статье представлен текущий статус разработки генератора индивидуальных заданий по проектированию БД на Python. Создан прототип в виде набора скриптов, который позволяет генерировать уникальные варианты заданий и эталонные SQL-скрипты. Проект не завершён, не имеет EXE-версии и не внедрялся в учебный процесс. Основные ограничения – отсутствие автоматической проверки, графического интерфейса и доработка некоторых типов связей.

Подобная «недоделанность» является осознанной – автор учитывает нормы об интеллектуальной собственности в вузе и планирует завершить разработку только после чёткого согласования прав на результат. Тем не менее, концепция могут быть полезны коллегам как отправная точка для собственных экспериментальных решений.

В дальнейшем, после устранения указанных недостатков и получения разрешения от администрации, планируется опубликовать готовый инструмент в открытом доступе (включая EXE-версию) и провести его педагогическую апробацию.

Список литературы

1. Дейт К.Дж. Введение в системы баз данных / К.Дж. Дейт. –8-е изд. – М.: Издательский дом «Вильяме», 2005. – 1328 с. : ил. – Пер. с англ. – Парал. тит. англ. EDN QMORMN

2. Коннолли Т. Базы данных проектирование, реализация и сопровождение, теория и практика / Т. Коннолли, К. Бегг. – 3-е изд. – М. [и др.] : Вильямс, 2018. – 1439 с. ил., табл.; 25. – Пер. с англ. Р.Г. Имамутдиновой, К.А. Птицына. – ISBN 978–5-8459–2020–1.

3. Мирошниченко М.А. Цифровая трансформация: российские приоритеты формирования цифровой экономики : монография / М.А. Мирошниченко. –

Краснодар : Кубанский государственный университет, 2021. – 224 с. EDN WQFJGW