

СОДЕРЖАНИЕ

[Введение 3](#_Toc168416458)

[1 Образование и научная работа Шрейдера 8](#_Toc168416459)

[2 История 10](#_Toc168416460)

[3 Теоретическая часть 12](#_Toc168416461)

[3.1 Определения 12](#_Toc168416462)

[3.2 Шрейдер о своей работе 16](#_Toc168416463)

[4 Практическая часть 20](#_Toc168416464)

[4.1 Процесс подсчета 20](#_Toc168416465)

[4.2 Пример подсчета 23](#_Toc168416466)

[5 Другие подходы 24](#_Toc168416467)

[5.1 По Хартли 24](#_Toc168416468)

[5.2 По Шеннону 25](#_Toc168416469)

[Заключение 28](#_Toc168416470)

[Список используемой литературы 30](#_Toc168416471)

ВВЕДЕНИЕ

Современный мир информационных технологий характеризуется стремительным развитием программного обеспечения и увеличением объемов обрабатываемых данных. В условиях ужесточения требований к защите персональной информации, таких как Общий регламент по защите данных (GDPR) в ЕС и Федеральный закон "О персональных данных" в России, анонимизация данных становится критически важным процессом. Она позволяет использовать реальные данные в разработке и тестировании программного обеспечения, минимизируя риски нарушения конфиденциальности.

Анонимизация данных — это процесс преобразования информации таким образом, чтобы исключить возможность идентификации личности или восстановления исходных данных. В разработке программного обеспечения она применяется для тестирования на реалистичных наборах данных без раскрытия персональных сведений. Это особенно важно в таких сферах, как здравоохранение, банковское дело и электронная коммерция, где утечка данных может привести к серьезным последствиям.

Цель данной курсовой работы — разработка генератора анонимизированных реляционных баз данных, который позволит создавать реалистичные, но при этом безопасные с точки зрения конфиденциальности наборы данных для использования в разработке и тестировании ПО.

1. Постановка задачи

Целью курсовой работы является создание консольного приложения для генерации анонимизированных локальных баз данных, которое позволяет формировать тестовые данные, соответствующие заданной структуре, с учётом основных типов данных и ограничений целостности. Приложение должно сохранять реалистичность данных и их пригодность для использования в разработке и тестировании ПО.

Приложение должно обеспечивать возможность работы со следующими типами данных:

* Целые числа;
* Числа с плавающей точкой;
* Строки;
* Даты;
* Числовые интервалы;
* Интервалы дат.

Кроме того, приложение должно учитывать возможность наличия пустого значения (NULL) в полях реляционных таблиц.

Приложение должно учитывать следующие ограничения целостности данных:

* Первичный ключ;
* Внешний ключ;
* Уникальность.

Приложение должно обеспечивать возможность использования в качестве типа данных первичного ключа следующих типов данных:

* Целое число;
* GUID;
* ObjectID.

Приложение должно обеспечивать возможность работы c двумя типами кардинальности связей между реляционными таблицами, а именно «Один к одному» и «Один ко многим». Также должна быть обеспечена возможность задания правила того, является ли обязательным наличие сущности (сущностей) с левой и с правой стороны связи. В случае связи «Один ко многим» приложение должно позволить задать правило, согласно которому будет определяться количество сущностей с правой стороны связи.

Приложение должно поддерживать возможность многопоточного заполнения базы данных, т.е. одновременного (параллельного) заполнения нескольких таблиц.

Список таблиц базы данных, типы данных столбцов таблиц, ограничения целостности, кардинальность связей между таблицами, правила, согласно которым будут генерироваться значения полей таблиц, задаётся в отдельном файле (файлах) формата JSON.

1. Анализ предметной области
   1. Подходы к анонимизации данных

Современные подходы к анонимизации данных направлены на решение задачи защиты конфиденциальности информации. В области защиты данных применяются различные техники анонимизации, каждая из которых имеет свои особенности.

Обобщение предполагает замену точных значений на диапазоны или категории (например, замена значения возраста человека с «24» на «20-30»), что полезно для аналитических отчётов, но мало применимо в тестировании, где требуются конкретные значения.

Псевдонимизация использует замену идентификаторов на псевдонимы с использованием хеширования или таблиц соответствия, что сохраняет риски для конфиденциальности, т.к. позволяет восстановить исходные данные при наличии ключа.

Шифрование данных хотя и обеспечивает высокий уровень защиты, но делает данные непригодными для непосредственного тестирования без этапа дешифрования.

Частичное удаление полей может нарушить целостность базы данных и логику работы приложения.

На этом фоне маскировка данных выделяется как наиболее сбалансированный подход. Маскировка данных — это процесс замены оригинальных значений на искусственно сгенерированные, которые сохраняют все формальные характеристики исходных данных, но при этом исключают возможность идентификации личности. Главное преимущество маскировки заключается в её способности сохранять структуру данных и их пригодность для тестирования. В отличие от шифрования или удаления, этот метод не нарушает целостность базы данных и позволяет разработчикам работать с информацией, которая по всем формальным признакам соответствует реальной. Это особенно важно при тестировании функционала, чувствительного к типам данных и форматам — валидации форм, обработке транзакций или генерации отчётов.

Маскировка также позволяет учитывать ограничения целостности данных. Например, при работе с первичными ключами можно генерировать уникальные значения, а для внешних ключей — создавать согласованные наборы данных, сохраняющие связи между таблицами. Это делает тестовые среды максимально приближенными к реальным условиям эксплуатации системы.

С точки зрения безопасности маскировка обеспечивает необратимое преобразование данных, что соответствует строгим требованиям современных стандартов защиты информации.

* 1. Способы генерации чисел

В контексте генерации тестовых данных принципиальное значение имеет корректная имитация числовых значений, приближенных к реальным данным. Важно отметить, что в практических сценариях числовые данные редко являются "по-настоящему случайными" в строгом математическом смысле. Напротив, они подчиняются определенным вероятностным распределениям, отражающим закономерности предметной области.

В теории вероятностей и математической статистике все случайные величины принципиально разделяются на два класса: дискретные и непрерывные. Это фундаментальное различие определяет не только их математические свойства, но и методы генерации соответствующих значений при создании тестовых данных.

Дискретные случайные величины принимают отдельные, изолированные значения (в контексте разрабатываемого приложения – целочисленные), в то время как непрерывные случайные величины могут принимать любое значение в некотором интервале. В разрабатываемом приложении случайной величиной такого типа будут являться числа с плавающей точкой.

При генерации целочисленных значений (дискретных данных) наиболее часто используются три распределения: равномерное, биномиальное и распределение Пуассона.

Равномерное распределение характеризуется одинаковой вероятностью появления любого значения из заданного диапазона. Применяется в случаях, когда все исходы равновероятны.

Биномиальное распределение описывает количество успехов в серии независимых испытаний с постоянной вероятностью успеха.

Распределение Пуассона применяется для моделирования редких событий в фиксированный промежуток времени. Типичные примеры использования включают генерацию количества запросов к серверу в минуту или число ошибок на странице текста.

Для генерации вещественных чисел (непрерывных величин) наибольшее практическое значение имеют три распределения: равномерное, нормальное и экспоненциальное.

Равномерное непрерывное распределение аналогично дискретному случаю, все значения из заданного интервала имеют равную плотность вероятности. Используется при моделировании величин без выраженных закономерностей.

Нормальное распределение крайне важное для статистики распределение, описывающее множество природных и социальных явлений. Применяется для генерации данных, сосредоточенных вокруг среднего значения с постепенным уменьшением частоты отклонений, таких как рост людей или погрешности измерений.

Экспоненциальное распределение описывает временные интервалы между независимыми событиями, происходящими с постоянной средней интенсивностью. Широко используется при моделировании времени обслуживания запросов, длительности сеансов или сроков эксплуатации оборудования.

* 1. Способы генерации интервальных значений

При генерации тестовых данных, включающих временные периоды или числовые диапазоны, особую важность приобретает корректное моделирование отношений между интервалами. Фундаментальную основу для работы с такими отношениями составляет интервальная алгебра Аллена – формальная система, описывающая все возможные варианты взаимного расположения двух временных или числовых интервалов. Разработанная Джеймсом Алленом в 1983 году, эта алгебра находит широкое применение в планировании задач, анализе временных рядов и, что особенно важно для нашей задачи, в генерации тестовых данных. Алгебра Аллена определяет 13 базовых отношений между двумя интервалами A и B, которые полностью исчерпывают все возможные варианты их взаимного расположения на временной или числовой оси, однако При детальном анализе обнаруживается, что некоторые из этих отношений являются либо зеркальными отображениями друг друга, либо могут быть выражены через комбинации других. Для задач генерации тестовых данных мы можем существенно сократить базовый набор, оставив только 6 ключевых отношений, сохранив при этом всю выразительную мощность алгебры:

* Перекрытие: начало A до начала B, конец A после начала B, но до конца B
* Обратное перекрытие: зеркальное отношение к перекрытию
* Непосредственное следование: конец A точно совпадает с началом B
* Обратное непосредственное следование: зеркальное отношение к непосредственному следованию
* Включение: A полностью содержится внутри B
* Обратное включение: B полностью содержится внутри A

1. Практическая часть

В рамках практической реализации генератора анонимизированных баз данных алгоритм работы программы строится на последовательном выполнении трех ключевых этапов, обеспечивающих формирование корректных и согласованных тестовых данных.

Первый этап предполагает анализ исходных данных и создание специализированных генераторов значений для каждой колонки таблиц на основе правил, заданных в конфигурационном файле. На этом этапе программа анализирует типы данных, требуемые распределения, допустимые диапазоны значений, кратность связей между таблицами.

Второй этап заключается в непосредственной генерации данных. Используя созданные на предыдущем шаге генераторы, программа заполняет таблицы синтетическими данными, соблюдая заданный объём записей, статистические закономерности и взаимную согласованность данных в связанных таблицах.

Третий этап посвящен применению ограничений целостности. Программа добавляет в сформированные таблицы ограничения первичных ключей (PRIMARY KEY), внешних ключей (FOREIGN KEY), ограничения уникальности (UNIQUE) и обязательные поля (NOT NULL).

Реализация каждого этапа будет подробно рассмотрена в следующих разделах с приведением диаграмм, описывающих структуру приложения и алгоритмы.

* 1. Анализ исходных данных

Данный этап целиком посвящен анализу исходных данных и подготовке к непосредственной генерации анонимизированных данных. Как было описано в разделе 1, исходные данные и правила записаны в нескольких файлах формата json.

Первый из них должен описывать объект класса DatabaseSchema, данный класс является корневым элементом модели, описывающей схему базы данных и содержит список таблиц, представленных объектами класса Table. Класс Table содержит поле name строкового типа – имя таблицы, значение которого должно быть уникальным среди всех таблиц, и список столбцов, представленных объектами класса Column. Данный класс содержит строковое поле name – имя столбца, значение которого должно быть уникальным в пределах таблицы, а также поле type строкового типа, описывающее тип данных столбца. Структура и отношения вышеописанных классов приведены диаграммой классов на рисунке 1.

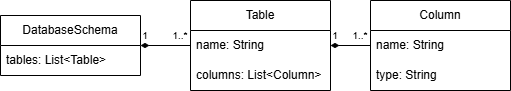


Рисунок 1 – Диаграмма классов схемы базы данных

Во втором файле формата JSON задаются правила, по которым будут генерироваться значения в столбцах, не являющихся первичными или внешними ключами. Все правила хранятся в одном экземпляре корневого класса RuleSet. В данном классе для каждого из допустимых в приложении типа данных содержится список правил, касающихся правил генерации значений в столбцах этого типа данных.

Для каждого типа данных правила генерации значений этого типа описаны собственным классом, однако все классы правил реализуют общий интерфейс Rule, имеющий методы getTableName() и getColumnName() для доступа к именам таблицы и столбца, правила генерации значений в котором описывает правило. Также в интерфейсе имеется метод getNullChance() для удобного доступа к значению шанса, что очередное сгенерированное значение будет пустым. Кроме того, в интерфейсе содержится метод toGenerator(), предназначенный для удобного и быстрого преобразования правила в готовый генератор значений данного типа. На рисунке 2 представлена диаграмма классов для набора правил генерации значений. Классы для правил конкретных типов будут подробно описаны далее в этом разделе.

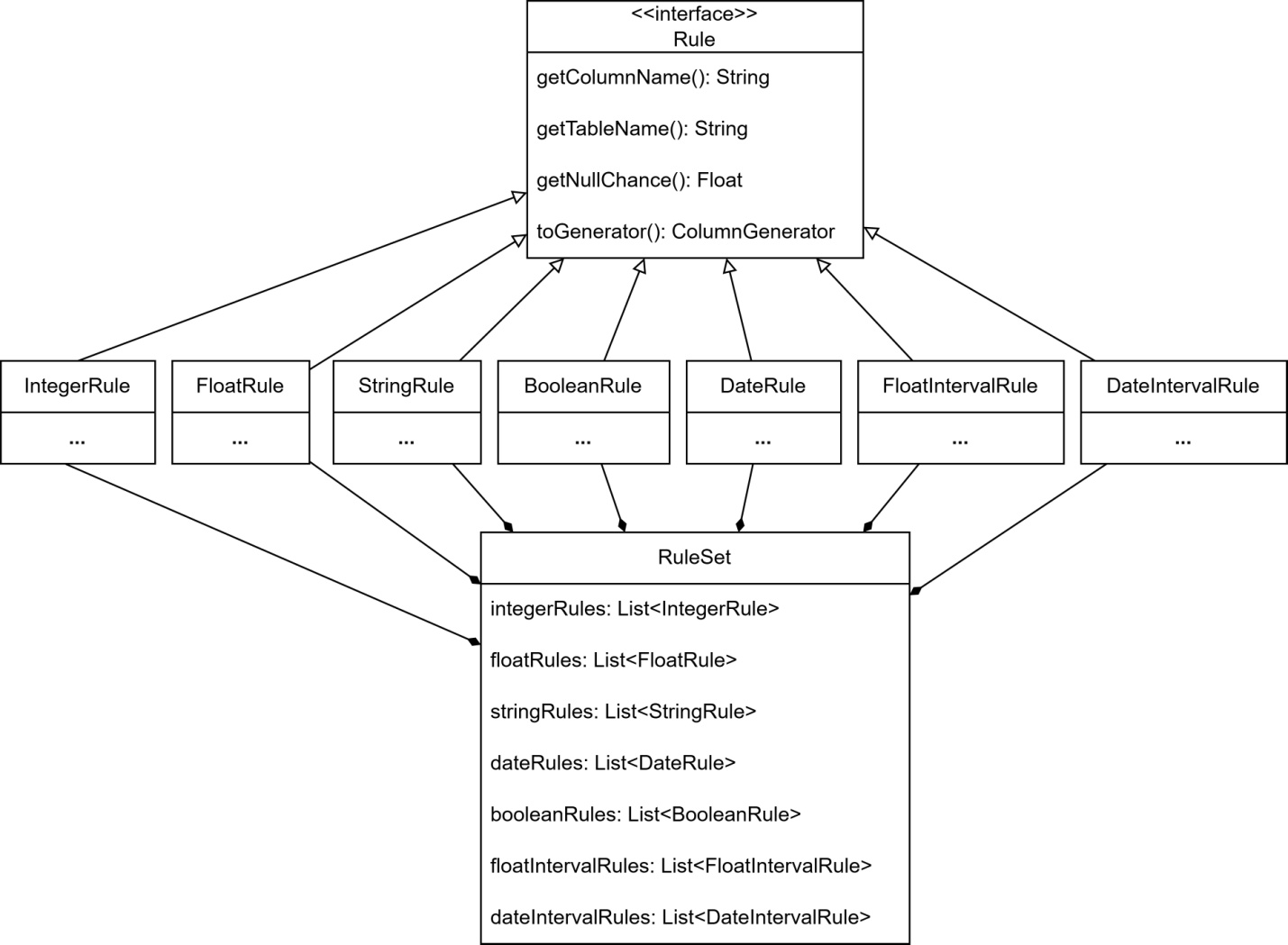


Рисунок 2 – Диаграмма классов набора правил генерации

Важно отметить, что все классы правил генерации данных содержат поля строкового типа columnName и tableName, а также поле типа числа с плавающей точкой nullChance. Значения, хранящиеся в этих полях, возвращаются в соответствующих трёх методах, унаследованных от интерфейса Rule. Далее при описании классов правил для конкретных типов данных эти три поля не будут упоминаться.

На рисунке 3 приведена диаграмма классов для правил генерации численных типов данных: IntegerRule и FloatRule. В обоих классах содержится поле distributionType, задающее тип распределения (дискретного в случае IntegerRule и непрерывного в случае FloatRule), а также поле params, являющееся массивом чисел с плавающей точкой и задающее параметры случайного распределения.

# class_diagram_float_int_rule.jpg

Рисунок 3 – Диаграмма классов для правил целых численных значений

Правило генерации строковых данных – StringRule – содержит в себе список экземпляров класса WordRule, каждый элемент которого является правилом для генерации отдельного слова, данный список хранится в поле wordRules. Также внутри класса имеется поле строкового типа separator, задающее, какой последовательностью символов будут разделены сгенерированные слова.

Класс WordRule задаёт правило генерации отдельного слова и имеет поле allowedCharacters, представляющее собой набор символов, разрешённых для использования в сгенерированных значениях. Полями distributionType и distributionParams, по аналогии с классом IntegerRule, задаётся дискретное распределение для генерации случайного значения длины слова. На рисунке 4 приведена диаграмма классов для правила генерации строковых значений.

# class_diagram_string.jpg

Рисунок 4 – Диаграмма классов для правил генерации строковых значений

Правило генерации значения даты описывается классом DateRule. Внутри содержатся два поля startDate и endDate, задающие границы интервала, внутри которого будет генерироваться случайная дата. В данных полях записаны строки формате ISO-8601[1]. На рисунке 5 представлена диаграмма классов для данного вида правила.

# class_diagram_date.jpg

Рисунок 5 – Диаграмма классов для правил генерации дат

Правила генерации значений интервалов чисел с плавающей точкой задаются в классе FloatIntervalRule. В классе имеется поле relations – список экземпляров класса FloatIntervalRelation, каждый из которых определяет то, в каком отношении с описанным в полях start и end (в классе FloatIntervalRule данные два поля имеют тип числа с плавающей точкой, в классе DateIntervalRule – строковый тип) интервалом обязан быть генерируемый интервал. Отношение является одним из правил интервальной алгебры Аллена. Каждое из содержащихся в поле relations правил должно выполняться для сгенерированного интервала одновременно. Диаграмма классов для правил генерации интервальных значений изображена на рисунке 6.

# class_diagram_interval_rules.jpg

Рисунок 6 – диаграмма классов для правил генерации интервалов

Третий файл формата JSON содержит экземпляр класса ConstraintSet и описывает, к каким столбцам таблиц необходимо применить ограничения.

Ограничения первичного ключа задаются в поле primaryKeys, являющимся списком объектов класса PrimaryKey. Каждый из них содержит строковые поля columnName и tableName, указывающие, к какой таблице и столбцу необходимо применить ограничение

Поле foreignKeys содержит список объектов класса ForeignKey и указывает, к каким таблицам и столбцам необходимо применить ограничение внешнего ключа. В данном классе имеются поля строкового типа sourceTableName и sourceColumnName, указывающие на столбец, являющийся ссылкой, а также targetTableName и targetColumnName, определяющие столбец, на который указывает данная ссылка. Кратность связи задаётся полями sourceDistributionType и sourceDistributionParams в виде дискретного случайного распределения. Поле sourceZeroChance задаёт, для какой доли записей с левой стороны связи не окажется сущностей с правой стороны, т.е. ссылка будет содержать пустое значение. Поле targetZeroChance определяет, на какую долю записей с правой стороны не будут указывать никакие записи с левой стороны.

Таким способом можно описать не только связь типа «Многие к одному», но и связь типа «Один к одному», указав в качестве случайного распределения равномерное с параметрами 1 и 2 (оно всегда будет генерировать число 1).

Ограничения уникальности описаны в поле uniques, являющемся списком объектов класса Unique. Внутри данного класса содержатся только поля columnName и tableName, указывающие на таблицу и столбец, к которому необходимо применить ограничение.

Диаграмма классов, описывающая набор ограничений, приведена на рисунке 7.

# class_diagram_constraints.jpg

Рисунок 7 – диаграмма классов для набора ограничений

Ограничения обязательных столбцов (NOT NULL) задаются согласно данным, описанным во втором файле: ограничение накладывается на каждый столбец, которому соответствует правило, в поле nullChance которого указано значение 0.

* 1. Генерация Данных

ЗАКЛЮЧЕНИЕ

----

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

---