ТЕХНИЧЕСКИЕ НАУКИ

УДК 004.654

Заполнение баз данных большим объемом случайной информации

А. Н. Музыченко
Донской государственный технический университет (г. Ростов-на-Дону, Российская Федерация)

Аннотация. Рассмотрена проблема заполнения базы данных тестовой информацией при разработке или для демонстрации клиенту. Описаны основные ситуации, которые возникают при необходимости заполнить базу данных большим объемом рандомной информации. Выделены основные способы такого заполнения баз. Для каждой ситуации выделены наиболее и наименее подходящие методы.

Ключевые слова: заполнение базы данных, язык программирования PHP, язык запросов SQL, заполнение базы данных вручную

Filling Databases With a Large Volume of Random Information

Anna N. Muzychenko
Don State Technical University, (Rostov-on-Don, Russian Federation)

Abstract. The article deals with the problem of the need to fill in the database with test data during its development or for demonstration to the client. The main situations that may arise when it is necessary to fill in databases with a large volume of random values are described. The paper provides the main ways of filling in databases with large amounts of random data. For each situation, the most and least suitable methods are highlighted.

Keywords: filling in a database, PHP programming language, SQL query language, filling a database manually

Введение. При создании базы данных (БД) у разработчика может не быть подходящих наборов тестовых значений. Это серьезная сложность, если нужно проверить работоспособность программного продукта или представить его заказчику. Важно знать, не будет ли сбоев по времени у базы с большим объемом данных и несколькими записями в каждой таблице [1]. Эти проблемы обусловили актуальность рассматриваемой темы.

Чаще всего для решения описанных выше задач разработчики берут в открытых источниках аналогичные базы данных с известными значениями либо пользуются рандомайзерами, которые можно найти в сети или написать самостоятельно [2–3]. Однако это подходит не для любого решения. Не всегда можно легко найти наполненную базу, а случайно сгенерированная информация способна нарушать логику базы данных и созданного на ее основе приложения.

Цель данной работы — показать успешную реализацию такой задачи. Необходимо разделить базы данных и способы их заполнения на группы и выбрать для каждого типа оптимальный способ заполнения.

Основная часть

Сравнение баз данных и способов их наполнения. Рассмотрим три основные ситуации.

1. Известно, что в открытом доступе есть примеры таких же или аналогичных заполненных баз данных. В этом случае нужно перенести информацию из одной базы в другую. Это самый простой вариант.

2. Случайные данные нужно создать самостоятельно. При этом в базе мало связей между таблицами. А если они и есть, то не отличаются четкой логикой. В качестве примера можно привести базу данных с информацией о продажах магазина, содержащей таблицы «Покупатель», «Продавец», «Товар» и «Покупка». В данном случае ссылки будут на идентификаторы первых трех таблиц. Можно использовать любой язык программирования и получить скрипты, выполнение которых позволит заполнить таблицы. Для этого создаются или скачиваются файлы с именами, фамилиями, номерами телефонов, товарами, а оттуда уже случайным образом выбираются значения. Скрипт основной таблицы следует создавать в зависимости от количества записей в связанных таблицах. Можно сначала рандомно выбрать данные в многомерный массив, а из него создать текстовые файлы для внесения данных с SQL-скриптами (от англ. structured query language — язык структурированных запросов).

3. В базе данных много связанных таблиц, и записи в них логически зависят друг от друга (например, по времени). Допустим, товар имеет срок годности, и его нужно ежедневно перепроверять. К тому же приходится

учитывать новые поставки товара. Дополнительной сложностью будет система статусов заказов — резервирование товара, расчеты, данные о том, забрал ли клиент заказ, и другие сведения о наличии, запросах и реализации товаров. В таком случае только часть таблиц заполнятся методами, описанными выше, в пунктах 1 и 2. С остальными таблицами, вероятно, возникнут сложности. Представим три выхода из ситуации.

Первый способ заполнения таких таблиц — вручную. Это самый трудозатратный вариант. Допустим, человек заполняет таблицы вручную и его скорость — 5 реалистичных записей в минуту. В базе данных 25 таблиц, и в каждую нужно внести от 500 записей до 10 тыс. (в среднем по 5 тыс.). Значит, для заполнения базы данных нужно создать в среднем 125 тыс. записей. В день человек работает 8 часов с перерывом на 1 час. 7 часов — это 420 минут, то есть примерно 2,5 тыс. записей в день. Расчеты показывают, что на это понадобится около двух месяцев монотонной работы.

Второй способ рассмотрим на примере PHP (от англ. hypertext preprocessor — препроцессор гипертекста). Так называется один из скриптовых языков программирования. Запросы формируются с помощью PHP и отправляются в базу. Полученные из нее данные анализируются, и в базу идет запрос на добавление новых записей [4]. Этот вариант может показаться очень простым. SQL выбирает нужную информацию, которую невозможно было бы так легко получить из структуры данных (например, из многомерного массива), а на языке PHP они обрабатываются в определенных циклах, функциях и возвращаются в базу данных. Представим, что для создания 125 тыс. реалистичных взаимосвязанных записей нужно обратиться в базу данных примерно 20 тыс. раз (учитывая, что запрос делается до и после обработки). Это создаст сильную нагрузку и приведет к тому, что запрос может вовсе не выполниться, т. к. реализуется цикл, представленный на рис. 1.

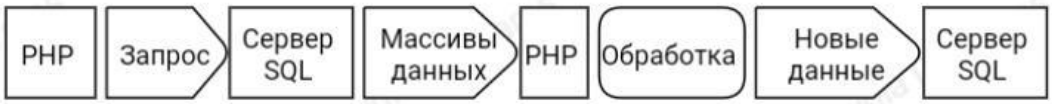

Рис. 1. Цикл взаимодействия PHP с сервером SQL

Взаимодействие PHP с сервером SQL создает большую нагрузку. Поэтому данный способ нельзя считать надежным. Кроме того, он не дает возможности предположить время выполнения.

Третий подход предполагает заполнение случайными данными на стороне SQL с помощью функций, рекурсий, процедур, триггеров. Этот способ сложнее, потому что SQL — язык запросов, а не программирования, и не располагает полным инструментарием для разработки. Но даже с помощью SQL можно написать нужные скрипты. Это будет надежный и оптимальный по времени способ. Заполнение базы большим объемом связанных данных может занять всего несколько часов (меньше, чем полдня). У человека на это ушло бы два месяца (на продумывание и заполнение вручную).

В табл. 1 для описанных выше ситуаций приводится сравнение способов достижения результата по времени (как быстро заполнится база) по шкале от 1 до 5, где 5 — самый быстрый для заполнения способ, 1 — самый медленный, который может и не сработать из-за ограничений по времени.

Таблица 1

Оценка скорости заполнения баз данных

Ситуация	Способ			
	Поиск аналогичных БД	1-й. Заполнение вручную	2-й. Рандомные записи на PHP	3-й. Рандомные записи на SQL
1. Есть такие же БД	5	2	2	2
2. БД несложная, связей мало	3	2	5	3
3. БД сложная, много логически связанных таблиц, в т. ч. по времени	1	2	1	5

В табл. 2 для вышеописанных ситуаций сравниваются скорости реализации способов достижения результата (как быстро тот или иной способ выполнит процесс заполнения базы данных) по шкале от 1 до 5, где 5 — самый быстрый вариант, 1 — самый медленный, при котором затраты на исполнение будут очень большими, но необоснованными.

Таблица 2

Оценка скорости реализации способов заполнения баз данных

Ситуация	Способ			
	Поиск аналогичных БД	1-й. Заполнение вручную	2-й. Рандомные записи на PHP	3-й. Рандомные записи на SQL
1. Есть такие же БД	5	1	3	1
2. БД несложная, связей мало	2	1	5	1
3. БД сложная, много логически связанных таблиц, в т. ч. по времени	1	1	2	5

Просуммировав значения и переведя результат в проценты, получим результирующую табл. 3. Здесь 100 % — наиболее подходящий способ заполнения БД, а 20–30 % — наименее подходящий.

Таблица 3

Оценка скорости исполнения способов заполнения баз данных, %

Ситуация	Способ			
	Поиск аналогичных БД	1-й. Заполнение вручную	2-й. Рандомные записи на PHP	3-й. Рандомные записи на SQL
1. Есть такие же БД	100	30	50	30
2. БД несложная, связей мало	50	30	100	40
3. БД сложная, много логически связанных таблиц, в т. ч. по времени	20	30	30	100

Заключение. Итоги научной работы позволяют сделать следующие выводы.

1. Если в сети есть подходящие заполненные БД, оптимальное решение — копировать готовые данные.

2. Для несложных БД с небольшим количеством связей между таблицами логично написать рандомайзер на любом из языков программирования (или использовать существующий генератор данных) и задействовать это решение для формирования запросов.

3. Для сложных БД с обширными связями между таблицами и взаимно зависимыми записями (логически и по времени) следует сформировать рандомные записи на языке запросов SQL.

4. В любом случае заполнение вручную не подходит для объемных БД.

Библиографический список

1. Коннолли Т., Бегг К., Страчан А. *Базы данных. Проектирование, реализация и сопровождение. Теория и практика*. 3-е изд. Москва: Вильямс; 2003. 1440 с.

2. Костычев Е.А., Омельченко В.А., Зеленов С.В. Нацеленная генерация данных для тестирования приложений над базами данных. *Труды института системного программирования РАН*. 2011;(20):253–268.

3. Кошелев О.В. Шаблоны проектирования в программировании. *Cyberleninka*. URL: https://cyberleninka.ru/article/n/shablony-proektirovaniya-v-programmirovanii (дата обращения: 26.09.2023).

4. Бинус П.И., Шушарина И.С. Функциональные возможности системы автоматического заполнения баз данных. В: *Материалы VII ежегодной всерос. межвуз. науч.-практ. конф. «Информационные технологии в науке и образовании. Проблемы и перспективы»*. Пенза: Изд-во ПГУ; 2020. 123–125.

Об авторе:

Музыченко Анна Николаевна, магистрант кафедры «Институт перспективного машиностроения» Донского государственного технического университета (344003, РФ, г. Ростов-на-Дону, пл. Гагарина, 1), a_m.0000@bk.ru

About the Author:

Anna N. Muzychenko, Master's degree student of the Institute of Advanced Mechanical Engineering Department, Don State Technical University (1, Gagarin Sq., Rostov-on-Don, 344003, RF), a_m.0000@bk.ru