

Проектирование и реализация логики запросов для базы данных авиаперевозок на базе PostgreSQL и PL/pgSQL

PostgreSQL для администраторов баз данных и разработчиков



**Меня хорошо видно
& слышно?**



Защита проекта

Тема: Проектирование и реализация логики запросов для базы данных авиаперевозок на базе PostgreSQL и PL/pgSQL



Сергей Айдинов

Java – программист
Сертифицированный Spring -
разработчик



План защиты

Цель и задачи проекта

Какие технологии использовались

Что получилось

Выводы

Вопросы и рекомендации



Цель и задачи проекта

Цель проекта: научиться писать простые функции на языке PL/pgSQL

Задачи, выполняемые в рамках работы:

1. Разработка функций на языке PL/pgSQL для автоматизации поиска сведений по заданным поисковым критериям.
2. Использование утилиты liquibase для «накатывания» функций.
3. Проведение тестирования и отладки функций, включая обработку исключительных ситуаций (слишком много данных в выборке и т.д.).
4. Разработка простого бэк - приложения на языке программирования Java для вызова процедур и отображения результатов.
5. Подготовка презентации и демонстрация работы системы на тестовых данных.



Какие технологии использовались

1. Система управления базами данных PostgreSQL
2. Процедурное расширение языка SQL PL/pgSQL
3. Утилита Liquibase
4. Фреймворк Spring Boot
5. Интегрированная среда разработки программного обеспечения IntelliJ IDEA
6. Графический клиент для работы с базами данных Dbeaver



Фактор загрузки самолета

Функция, вычисляющая фактор загрузки самолета (т.е. отношение количества проданных билетов к общему количеству мест в самолете). В качестве параметров функция принимает минимальный и максимальный факторы загрузки и возвращает список рейсов, где фактор загрузки находится в указанных пределах (включительно).
Функция возвращает следующую таблицу:

flight_id	aircraft_code	totally_seats	tickets_sold	passenger_lf	metadata
-----------	---------------	---------------	--------------	--------------	----------

В колонку metadata вставляется всего один JSON (в последней строке), содержащий сведения об имени функции и времени выполнения запроса.

Сложность функции заключается в том, что сведения, необходимые для ее работы, находятся в трех разных таблицах: seats, ticket_flights и flights.

Функция написана в двух вариантах: обычном (работает и имеющимися таблицами базы данных) и оптимизированном. Во втором случае в базе данных создается дополнительная таблица flights_load_factor, содержащая сведения о факторе загрузки, которая заполняется автоматически с помощью функции fill_table_flights_load_factor() при запуске приложения.



Ссылки на функции в репозитории проекта:

passenger_load_factor:

https://github.com/SergeiAidinov/pg_diploma/blob/master/src/main/resources/db/changelog/functions/passenger_load_factor.sql

passenger_load_factor_optimized:

https://github.com/SergeiAidinov/pg_diploma/blob/master/src/main/resources/db/changelog/functions/passenger_load_factor_optimized.sql

fill_table_flights_load_factor:

https://github.com/SergeiAidinov/pg_diploma/blob/master/src/main/resources/db/changelog/functions/fill_table_flights_load_factor.sql

Вызов функций с параметрами загрузки в пределах от 0.6 до 0.7 включительно возвращает 17 753 строку.

Индексы

Построение индекса командой `create index if not exists flight_id_idx ON bookings.ticket_flights USING btree (flight_id)` **существенно** (с нескольких часов до нескольких секунд!!!) сокращает время работы функции `fill_table_flights_load_factor()`, однако примерно в два раза увеличивает время работы функции `passenger_load_factor`: (с 4,6 сек до 9,7 сек).



Поэтому этот индекс целесообразно строить перед обновлением таблицы `flights_load_factor` и затем удалять.

Однако построение индекса для таблицы `flights_load_factor` командой `create index if not exists flights_load_factor_idx ON flights_load_factor USING btree (load_factor)` сокращает работу оптимизированной функции примерно в девять раз: с 45 мс до 5 мс.



Функция создания бронирования

Функция `create_booking()` имитирует создание бронирования. Сложность заключалась в том, что сведения о бронировании необходимо вносить в две таблицы: `tickets` и `bookings`. Поле `ticket_no` в таблице является текстовым, хотя фактически содержит порядковые номера. При этом автонумерация строк отсутствует. Поэтому была создана функция `custom_sequence_generator_tickets_id()`, выполняющая роль «кастомного секвенсора» и увеличивающая номера билетов, используя не только цифры, но и буквы латинского алфавита. Фактически при увеличении номера используется система счисления по основанию 36. Эта функция вызывается триггером, который назначен на таблицу `tickets`.

Функция `create_booking` в качестве аргумента принимает в том числе массив пользовательских типов `passenger_and_ticket_price_type`



Ссылки на репозиторий:

Функция create_booking():

https://github.com/SergeiAidinov/pg_diploma/blob/present/src/main/resources/db/changelog/functions/create_booking.sql

Функция custom_sequence_generator_tickets_id():

https://github.com/SergeiAidinov/pg_diploma/blob/present/src/main/resources/db/changelog/functions/custom_sequence_generator_tickets_id.sql

Тип passenger_and_ticket_price_type:

https://github.com/SergeiAidinov/pg_diploma/blob/present/src/main/resources/db/changelog/types/passenger_and_ticket_price_type.sql

Триггер:

https://github.com/SergeiAidinov/pg_diploma/blob/present/src/main/resources/db/changelog/triggers/ticket_no.sql



Выводы

1. В целом, получилось выполнить все задачи и даже немного больше (liquibase изначально не планировался).
2. В процессе работы над проектом я научился писать простые функции на языке PL/pgSQL, создал кастомный секвенсор, триггер и пользовательский тип.
3. Нашел способ оптимизировать получение экономически значимой информации из базы данных путем создания отдельной таблицы и заблаговременного наполнения ее данными.
4. Опыт работы над проектом оказался для меня довольно полезным, оцениваю его на 8 баллов из 10.
5. Дальше планирую активнее использовать возможности языка PL/pgSQL в своей работе в качестве Java – программиста.



Вопросы и рекомендации



если есть вопросы



если вопросов нет

Спасибо за внимание!



otus

