Санкт-Петербургский национальный исследовательский университет ИТМО

Факультет Программной Инженерии и Компьютерной техники

Информационные системы и базы данных

Курсовая работа

Имиджборд ИТМО

Этап 3

Выполнили:

Беляков Д. В. группа P33131,

Тюрин И. Н. группа P33102

Преподаватель:

Харитонова А. Е.,

Сагайдак А. А.

Санкт-Петербург
2023

Оглавление

Текст задания

Реализовать даталогическую модель в реляционной СУБД PostgreSQL:

- Создать необходимые объекты базы данных.
- Заполнить созданные таблицы тестовыми данными.
- Сделать скрипты для:
  - создания/удаления объектов базы данных;
  - заполнения/удаления созданных таблиц.
- Обеспечить целостность данных при помощи средств языка DDL.
- Добавить в базу данных триггеры для обеспечения комплексных ограничений целостности.
- Реализовать функции и процедуры на основе описания бизнес-процессов (из этапа №1).
- Произвести анализ использования созданной базы данных:
  - выявить наиболее часто используемые запросы к объектам базы данных;
  - результаты представить в виде текстового описания.
- Создать индексы и доказать, что они полезны для вашей базы данных:
  - доказательство должно быть приведено в виде текстового описания.

Выполнение задания

Все необходимые объекты базы данных и SQL-скрипты для управления ими были написаны и находятся в публичном репозитории на Github:
https://github.com/e1turin/itmo-db-and-is/tree/main/coursework.

Там есть как DDL скрипты для создания сущностей в базе и установления связей между ними (`sql/entities` и `sql/entities/references.sql`), так и DML скрипты для добавления/удаления/получения/обновления сущностей в соответствии с предполагаемыми бизнес процессами. То есть, мы предполагаем, что комментарии пользователей не удаляются полноценно, а лишь помечаются флажком “`deleted`”. Это сделано для того, чтобы можно было проводить анализ по удаленным комментариям и в будущем настроить процесс модерации.

Также там размещен код для создания триггеров в базе данных (`sql/triggers`) и хранимых процедур (`sql/functions`).

Для удобства были написаны скрипты для автоматического создания нужных объектов в базе (`sql/build.psql`) и их удаления (`sql/destroy.psql`).

Анализ использования созданной базы данных

Есть учитывать тематику нашего проекта, то самые частые запросы – это создание и получение комментариев, получение топиков и тредов. А также получение реакций к каждому комментарию. Ведь почти на всём сайте есть возможность прочитать и написать комментарии.

Небольшой сниппет кода, который отвечает за получение информации о комментарии:

```
WITH Pics AS (
       SELECT * FROM "Picture_attachments"
       LEFT JOIN "Pictures"
       ON "Pictures".picture_id = "Picture_attachments".picture_id
       WHERE "Picture_attachments".comment_id = 7
), Vids AS (
       SELECT * FROM "Video_attachments"
       LEFT JOIN "Videos"
       ON "Videos".video_id = "Video_attachments".video_id
       WHERE "Video_attachments".comment_id = 7
)
SELECT * FROM "Comments"
LEFT JOIN "Reaction_sets"
  ON "Comments".reactions_id = "Reaction_sets".r_set_id
LEFT JOIN Pics
  ON "Comments".comment_id = Pics.comment_id
LEFT JOIN Vids
  ON "Comments".comment_id = Vids.comment_id
WHERE "Comments".comment_id = 7;
```

На самом деле таких запросов в бизнес логике не будет, а будут отдельные подзапросы с клиента — это добавит гибкости на клиенте и снимает с базы нагрузку на соединение подзапросов.

И пример кода по добавлению комментариев:

```
INSERT INTO "Comments" (thread_id, content, user_id)
VALUES (1, 'Privet, Misha!', 7),
       (1, 'Privet, SuperHot!', 8),
       (1, 'Privet, SuperHot!', 8),
       (1, 'There is nothing', 6);
```

После вставки комментария должны отрабатывать триггеры на увеличение популярности треда и добавление реакций.

Индексы

Для ускорения выполнения запросов были созданы следующие индексы:

```
CREATE INDEX "Users_isu_index" ON "Users" USING btree(isu_id);
CREATE INDEX "Users_username_substring_index" ON "Users" USING GIN(username);
CREATE INDEX "Users_username_index" ON "Users" USING hash(username);

CREATE INDEX "Comments_deleted_index" ON "Comments" ((1)) WHERE deleted;
CREATE INDEX "Comments_user_id_index" ON "Comments" USING btree(user_id);
CREATE INDEX "Comments_creation_date_index" ON "Comments" USING btree(creation_date);
CREATE INDEX "Comments_thread_id_index" ON "Comments" USING btree(thread_id);
CREATE INDEX "Comments_content_substring_index" ON "Comments" USING GIN(content);
CREATE INDEX "Comments_title_substring_index" ON "Comments" USING GIN(title);

CREATE INDEX "Poll_answers_poll_id_index" ON "Poll_answers" USING hash(poll_id);
CREATE INDEX "Polls_comment_id_index" ON "Polls" USING hash(comment_id);

CREATE INDEX "Trash_comment_id_index" ON "Trash" USING hash(comment_id);
CREATE INDEX "Trash_recycle_date_index" ON "Trash" USING btree(recycle_date);
```

Данные индексы почти все рассчитаны на джоины с другими таблицами и фильтрации “удаленных” комментариев. Так, например, индексируется атрибут `deleted` в Comments и `recycle_date` в Trash, чтобы можно было быстрее фильтровать запросы и хешируются внешние ключи таблиц (возможно стоит использовать btree или другие индексы), чтобы можно было быстро соединять нужные кортежи. Еще добавлены инвертированные индексы для быстрого поиска по тексту комментария и его заголовка. Для разных attachments индексы специально прописывать не надо, так как там FKs входят в состав PK.

Стоит сказать, что для других индексов применения пока не нашлось, тем не менее, в процессе работы сервиса можно будет выявить тенденции в данных и узкие места в базе данных, и тогда уже добавить новые индексы и/или удалить ненужные.

Вывод

В ходе выполнения задания разработали скрипты для создания и изменения базы данных для ранее представленных бизнес задач. Во время работы выявились недостатки в ранее разработанных ER-моделях: потребовалось добавлять новые атрибуты сущностям комментария, треда, помойки, без которых логика запросов стала бы чрезвычайно сложной. Недостатки были исправлены. Были изучены принципы написания хранимых функций, процедур и триггеров в PostgreSQL. Необходимые для бизнес логики процедуры и триггеры были добавлены в базу данных.