Лабораторна робота 2: Перетворення вашої ER-діаграми на схему PostgreSQL

У лабораторній роботі 1 ви побудували ER-діаграму (сутності, атрибути та зв'язки) для вашого проекту. У цій лабораторній роботі ви перетворите цю ER-модель на фактичну реляційну схему. Використовуйте свою ERD як план: кожна сутність стає таблицею, а кожен зв'язок стає зовнішнім ключем (або таблицею об'єднання для багато-до-багатьох). Ви можете переглянути свою ERD зараз, якщо потрібно (наприклад, розділити або об'єднати таблиці, або додати відсутні зв'язки) перед написанням SQL.

Цілі

— Написати SQL DDL-інструкції для створення кожної таблиці з вашої ERD в PostgreSQL.

— Вказати відповідні типи даних для кожного стовпця, вибрати первинний ключ для кожної таблиці та визначити будь-які необхідні зовнішні ключі, обмеження UNIQUE, NOT NULL, CHECK або DEFAULT.

— Вставити зразки рядків (принаймні 3–5 рядків на таблицю) за допомогою `INSERT INTO`.

— Нарешті, протестувати все в pgAdmin (або іншому клієнті PostgreSQL), щоб переконатися, що таблиці та дані завантажуються правильно.

Інструкції

Посилання на гайд з інсталювання необхідних інструменів для лабораторних:

db-intro-course/lectures/2 - ER diagrams/lecture_notes.md at master · ZheniaTrochun/db-intro-course

Docker Compose:

https://github.com/ZheniaTrochun/db-intro-course/blob/master/docker-compose.yml

Переконайтеся, що представлені всі сутності та зв'язки на ERD. Визначте первинні ключі (PK) для кожної сутності. Для зв'язків «один до багатьох» та «багато до багатьох» вирішіть, як їх застосовувати: зазвичай стовпець зовнішнього ключа на стороні «багато» посилається на батьківську таблицю, а зв'язки «багато до багатьох» вимагають *таблиці сполучення* зі складеним первинним ключем.

За бажанням, намалюйте діаграму реляційної схеми. Перед написанням коду ви можете намалювати просту діаграму, що показує таблиці, ключові стовпці та посилання зовнішніх ключів. Це може допомогти спланувати SQL. (Такі інструменти, як draw.io, DbDiagram або pen-and-paper, чудово підійдуть.)

Напишіть оператори `CREATE TABLE` для кожної сутності/таблиці:

1. Визначте назву таблиці та список стовпців з типами даних. Виберіть типи, що відповідають даним (наприклад, `INTEGER` для ідентифікаторів/кількості, `VARCHAR(n)` для короткого тексту, `TEXT` для довшого тексту, `DATE/TIMESTAMP` для дат, `NUMERIC` для точних чисел тощо).
2. Визначте, які стовпці не можуть бути null, і за потреби додайте `NOT NULL`.
3. (Необов'язково) Додайте значення `DEFAULT` для будь-яких стовпців, де значення за замовчуванням має сенс.
4. Включіть будь-які інші бажані обмеження стовпців (наприклад, `UNIQUE`).

Визначте ключі та обмеження. Удоскональте свої оператори `CREATE TABLE` наступним чином:

1. Використовуйте **`PRIMARY KEY`** для вибраного(их) ключового(их) стовпця(ів). У PostgreSQL оголошення стовпця `INTEGER PRIMARY KEY` автоматично означає `UNIQUE NOT NULL`. Кожна таблиця повинна мати один первинний ключ (один стовпець або групу стовпців).
2. Для кожного зв'язку додайте обмеження **`FOREIGN KEY`**. Наприклад, `FOREIGN KEY (user_id) REFERENCES Users(id)` гарантує, що кожен `user_id` у цій таблиці відповідає існуючому `Users(id)`. Це забезпечує цілісність посилань. Якщо ви опустите список стовпців, на які посилаються, PostgreSQL за замовчуванням прийме первинний ключ батьківської таблиці (наприклад, `REFERENCES Users`).
3. Використовуйте **`CHECK`** для застосування простих правил до стовпця (наприклад, `CHECK (ціна > 0)`, щоб переконатися, що ціна додатна). Позначте обов'язкові стовпці як `NOT NULL`, щоб запобігти пропуску даних. Використовуйте `UNIQUE`, якщо стовпець або набір стовпців має бути унікальним. За потреби назвіть обмеження за допомогою `CONSTRAINT` (це необов'язково, але може зробити повідомлення про помилки зрозумілішими).

Наприклад, ви можете написати:

```
CREATE TABLE products (
  product_id   SERIAL PRIMARY KEY,
  name         VARCHAR(100) NOT NULL,
  price        NUMERIC(8,2) NOT NULL CHECK (price > 0),
  category_id  INTEGER REFERENCES categories(category_id)
);
```

Тут `product_id` – це первинний ключ (унікальний, не null), а `category_id` – зовнішній ключ, що посилається на таблицю категорій. `CHECK (price > 0)` гарантує, що дозволені лише додатні ціни.

Вставте зразки даних. Після створення кожної таблиці додайте зразки рядків для її тестування. Для цього використовуйте `INSERT INTO`. Надайте щонайменше 3–5 змістовних рядків у кожній таблиці. Переконайтеся, що рядки батьківської таблиці

вставлені перед рядками дочірньої таблиці (щоб існували посилання на зовнішні ключі). Наприклад:

```
INSERT INTO users (user_id, name, email) VALUES
  (1, 'Alice', 'alice@example.com'),
  (2, 'Bob',   'bob@example.com'),
  (3, 'Carol', 'carol@example.com');



INSERT INTO orders (order_id, user_id, amount) VALUES
  (10, 1, 100.00),
  (11, 1,  50.50),
  (12, 2,  75.25);
```

Використання тестових даних допомагає перевірити вашу схему. Заповнення таблиць дозволяє «перевірити вашу схему та зв'язки». Таким чином ви можете перевірити, чи всі вставки виконані успішно (без порушень обмежень).

Відкрийте pgAdmin (або будь-який клієнт PostgreSQL) та підключіться до бази даних. Виконайте оператори `CREATE TABLE` для побудови схеми, а потім виконайте оператори `INSERT`. Виправте будь-які синтаксичні помилки або проблеми з обмеженнями, що виникли. Наприклад, якщо зовнішній ключ не спрацьовує, переконайтеся, що батьківський рядок було вставлено першим. Зробіть запит до таблиць (наприклад, `SELECT * FROM table;`), щоб перевірити, чи дані відображаються належним чином.

Поради

— **Налаштування pgAdmin:** Встановіть PostgreSQL з офіційного сайту або EDB; інсталятор зазвичай містить pgAdmin 4 (графічний інструмент адміністрування). PgAdmin є безкоштовним і широко використовується в курсових роботах. Після встановлення запустіть *pgAdmin 4* з вашої системи (наприклад, у Windows він відображається в розділі «PostgreSQL» у меню «Пуск»). Під час першого запуску вас попросять створити головний пароль (для збережених підключень). Потім у лівому деревоподібному поданні клацніть правою кнопкою миші **Servers** та виберіть **Create → Server...** Введіть ім'я (наприклад, «Local») та використовуйте `localhost` з вашим ім'ям користувача/паролем PostgreSQL для підключення.

— **Запуск SQL:** Після підключення відкрийте **Інструмент запитів** у pgAdmin. Ви можете вставити туди свої оператори `CREATE TABLE` та `INSERT` і виконати їх. Щоразу, коли ви виконуєте SQL, переглядайте панель повідомлень, щоб побачити, чи все вдалося. Якщо ви надаєте перевагу командному рядку, ви також можете скористатися оболонкою `psql` або будь-яким редактором SQL.

— **Приклад екрана pgAdmin:** На скріншоті нижче показано запит головного пароля pgAdmin, який з'являється під час запуску (після встановлення).

Використовуйте вибраний пароль або той, що був встановлений під час встановлення. Після розблокування ви зможете перейти до свого сервера та бази даних.

— **Налагодження:** Якщо створення таблиці не вдається, прочитайте помилку – вона часто вказує на те, який рядок неправильний (наприклад, неправильна назва типу даних або відсутня кома). Якщо INSERT не вдається, перевірте зовнішні ключі та обмеження NOT NULL. Пам'ятайте, що додавання первинного ключа автоматично створює унікальний індекс, тому помилки дублікатів ключів означають, що ви вставляєте той самий ключ двічі.

Результати

— SQL-скрипт(и) у звіті з усіма вашими операторами `CREATE TABLE` та `INSERT` по порядку. Ви можете додавати коментарі до SQL-файлу для ідентифікації таблиць.

— **К**оротний письмовий звіт, що підсумовує вашу остаточну схему. Перелічіть кожну таблицю з її стовпцями та ключами, а також поясніть будь-які важливі обмеження або припущення (наприклад, «У таблиці Orders `customer_id` – це зовнішній ключ, що посилається на Customers(customer_id)). За потреби додайте діаграму вашої схеми.

— Доказ того, що таблиці були заповнені (це може бути частина SQL-файлу або скріншоти результатів запиту). Переконайтеся, що кожна таблиця містить щонайменше 3–5 рядків.

— Ви можете додати знімки екрана pgAdmin, що показують ваші таблиці, інструмент запитів або результати `SELECT *` , якщо це корисно (особливо для розділу Оцінка або якщо самі SQL-файли є неоднозначними).

Критерії оцінювання

Ваша лабораторна робота буде оцінена за такими критеріями:
- **Коректність схеми:** усі необхідні таблиці існують з правильними стовпцями та типами даних, що відображають вашу ER-модель.
- **Синтаксис SQL:** оператори `CREATE TABLE` та `INSERT` виконуються без помилок у PostgreSQL.
- **Визначення ключів:** кожна таблиця має відповідний первинний ключ, а зовнішні ключі правильно представляють зв'язки (забезпечуючи цілісність даних).
- **Використання обмежень:** обмеження використовуються змістовно (наприклад, `NOT NULL` для обов'язкових полів, `UNIQUE`, де потрібно, `CHECK` для будь-яких правил даних).
- **Узгодженість даних:** вибіркові дані узгоджуються зі схемою (наприклад, значення зовнішнього ключа відповідають існуючим первинним ключам, немає null-значень, де це заборонено, значення задовольняють умови `CHECK`).
- **Документація:** пояснення схеми є чітким та точним. (Таблиці та ключі повинні бути чітко описані.)

Наведені вище концепції базуються на стандартному проектуванні реляційних схем. Наприклад, у документації PostgreSQL зазначається, що `PRIMARY KEY` забезпечує унікальність та `NOT NULL` для ключового стовпця, а `FOREIGN KEY ... REFERENCES parent_table` запобігає вставці недійсних посилань Крім того, додавання зразків даних за допомогою `INSERT` допомагає перевірити схему. Ці рекомендації керують кроками в цій лабораторній роботі.

Додаткові ресурси:

Посібник для початківців з проектування схеми реляційної бази даних

https://dbschema.com/blog/design/steps-to-design-relational-schema/?srsltid=AfmBOop8uzffqet6y_veZlZcQgyfdgHB_K9-PEE9iLhWNNXzhWEJkbWx

PostgreSQL: Документація: 17: 5.5. Обмеження

https://www.postgresql.org/docs/current/ddl-constraints.html

PostgreSQL - pgAdmin 4

https://www.w3schools.com/postgresql/postgresql_pgadmin4.php