УДК 004.65

ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ PLANTUML ДЛЯ ПОКРАЩЕННЯ ПОБУДОВИ ER-ДІАГРАМ

Анатолій Куротич, Леся Булатецька

EXPLORING THE CAPABILITIES OF PLANTUML FOR ENHANCING ER DIAGRAM CONSTRUCTION

Anatolii Kurotych, Lesia Bulatetska,

Lesya Ukrainka Volyn National University, 13 Volya Avenue, Lutsk, 43025, Ukraine

Abstract. *This research analyzes PlantUML's capabilities for building ERD. The shortcomings of PlantUML's basic functionality for creating ERD are described. Additional PlantUML features are presented such as improving the appearance and readability of diagrams by highlighting primary and foreign keys, removing unnecessary elements, and creating legends for user convenience. A plugin module has been developed to enhance the readability of PlantUML code by structuring it into functions and procedures, making diagrams easier to create and maintain.*

Побудова ER-діаграм є важливою складовою розробки програмного забезпечення, оскільки забезпечує структуровану візуалізацію даних. Для цього часто застосовують CASE-інструменти, зокрема PlantUML – інструмент з відкритим кодом для генерації діаграм на основі текстового опису (puml-код) [1–3]. Не зважаючи на широкий функціонал, PlantUML недостатньо задокументований, що ускладнює його практичне використання і потребує подальших досліджень.

У цій роботі розглянуто тип діаграм PlantUML – Entity Relationship Diagram. Відповідно до документації PlantUML [1–3], сутність customer_order, як приклад, може бути описана мовою PlantUML для побудови ER-діаграми, що продемонстровано на рис. 1.

```
@startuml
entity "customer_order" {
  * id: bigserial <<PK>>
  --
  * customer_id: bigint <<FK>>
  * shipping_address: text
  * total_price: numeric
  * ordered_at: timestamp with time zone
  comment_to_order: text
}
@enduml
```

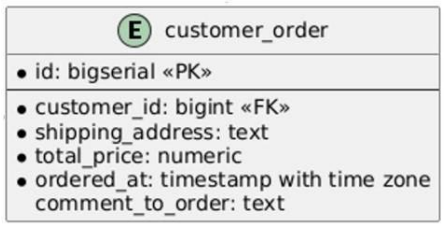

Рис. 1. Puml-код (*a*) та вигляд відображення сутності customer_order (*b*) відповідно до документації PlantUML .

Оскільки "Entity Relationship Diagram" призначена не лише для візуалізації схем реляційних баз даних, її базове подання (рис. 1) має низку недоліків:

- коло з літерою "E" зліва від назви таблиці не має інформативної цінності;
- первинні і зовнішні ключі слабо виділені, зазвичай їх виділяють кольорами чи додатковими символами;
- символ "*", який позначає обов’язковість поля (обмеження NOT NULL) і зображений як чорне коло ліворуч від імені змінної, може бути незрозумілим для користувача діаграми, не знайомого з PlantUML;

- однаковий шрифт як для назви змінної так і для її типу ускладнює сприйняття.

Згідно з дослідженнями [4], підсвічування синтаксису суттєво впливає на зручність роботи з кодом, тому наведені вище недоліки є цілком обґрунтованими.

Для усунення цих проблем при візуалізації сутності у PlantUML використано такі вдосконалення:

- видалена літера (E) біля назви таблиці (за допомогою дерективи "hide circle");
- первинний ключ зображено символом "золотистий ключ";
- зовнішній ключ зображено символом "сірий ключ".

Крім того, для уникнення артефактів рендерингу при створенні типу enum рекомендовано використовувати ключове слово object замість entity і додавати ідентифікатор "(E)" до назви.

Оскільки деякі символи можуть бути незрозумілими кінцевому користувачу, доцільно використовувати легенду для пояснення умовних позначень.

В результаті початковий код PlantUML став більш "навантаженим" і менш читабельним, тому для його спрощення пропонується використовувати вбудовані функції (!function), процедури (!procedure) та бібліотеки, які слід винести в окремі .puml-файли й підключати їх у головний файл командою !include. Такий підхід забезпечує модульність і покращує організацію коду. Щоб використати підключуваний модуль, його слід розмістити на вебресурсі з постійною адресою, наприклад, у репозиторії GitHub [5]. Після чого з'являється можливість підключати цей модуль в будь який PlantUML файл.

В результаті ми отримали наступні переваги:

- основний файл містить лише логіку діаграми, без зайвих деталей, що покращує читабельність коду;
- один модуль можна використовувати у різних діаграмах;
- зміни в модулі автоматично відображаються в усіх пов'язаних діаграмах, що може бути корисно в випадках, коли потрібно виправити помилку;
- стандартизація стилів та процедур для всіх учасників проєкту, спрощує роботу в команді.

Цей підхід дозволяє легко масштабувати наш проєкт і зберігати код зрозумілим та організованим. Оформлення функцій (та процедур) в підключуваний модуль дасть нам перевагу навіть у випадку зображення лише однієї сутності.

Puml-код в лістингах був протестований на офіційній сторінці PlantUML (https://www.plantuml.com/plantuml/uml). Для хостингу PlantUML модуля (файлу) було використано сервіс Github. SQL запити були перевірені на RDBMS PostgreSQL версії 15.0 [6]

Бібліографія

1. Guía de Referencia del Lenguaje PlantUML URL: https://pdf.plantuml.net/ (date of access: 06.05.2025).
2. PlantUML. PlantUML.com. URL: https://plantuml.com/ (date of access: 06.05.2025).
3. Welcome to The Hitchhiker's Guide to PlantUML! – The Hitchhiker's Guide to PlantUML documentation. Crashedmind GitHub. URL: https://crashedmind.github.io/PlantUMLHitchhikersGuide/ (date of access: 06.05.2025).
4. The impact of syntax colouring on program comprehension Proceedings of the 26th Annual Conference of the Psychology of Programming Interest Group (PPIG 2015) 10 p. URL: https://ppig.org/files/2015-PPIG-26th-Sarkar1.pdf
5. *GitHub · Build and ship software on a single, collaborative platform · GitHub*. URL: https://raw.githubusercontent.com/kurotych/sqlant/b2e5db9ed8659f281208a687a344b34ff38129cd/puml-lib/db_ent.puml (дата звернення: 06.05.2025)
6. A. O. Kurotych, L. V. Bulatetska, Optimizing the process of ER diagram creation with PlantUML, CEUR Workshop Proceedings (2025) 47–57. URL: https://ceur-ws.org/Vol-3917/paper12.pdf