УДК 004.7

Корецький Ю.І., здобувач,
Савіцький Р.С., аспірант, ст. викладач
Державний університет «Житомирська політехніка»

ПОРІВНЯЛЬНИЙ АНАЛІЗ АРХІТЕКТУРНИХ ПАТЕРНІВ ІЗОЛЯЦІЇ ДАНИХ У МУЛЬТИТЕНАНТНИХ ХМАРНИХ ЗАСТОСУНКАХ

Питання про те, як розмежувати дані між клієнтами в спільній системі, постає перед розробниками SaaS-застосунків набагато раніше, ніж більшість інших архітектурних рішень. Мультитенантність –це не просто технічна деталь реалізації, а фундаментальний вибір, що визначає подальшу форму всього продукту. Від моделі ізоляції залежить і те, як система поводитиметься під навантаженням, і те, наскільки складно буде її супроводжувати через рік. Існуючі підходи, а саме від виділеної бази на кожного клієнта до спільних таблиць із безпекою на рівні рядків (Row-Level Security, RLS), добре описані в загальних рисах [1], проте їх систематичне порівняння в контексті сучасних реляційних СУБД із підтримкою RLS залишається недостатньо опрацьованим.

Тема мультитенантності має помітну дослідницьку історію. Chong et al. [1] запропонували одну з перших класифікацій SaaS-архітектур, де рівень зрілості системи безпосередньо пов'язується зі ступенем спільного використання ресурсів. Пізніше Bezemer та Zaidman [2] детально розібрали технічні труднощі, з якими стикаються команди при переході на мультитенантну модель, зокрема складність міграцій і гарантування ізоляції. Kabbedijk та Jansen [3] узагальнили патерни варіативності на основі аналізу реальних промислових застосунків. Попри це, питання інтеграції RLS-механізмів реляційних СУБД з ORM-шаром практично не розглядалося.

Для порівняння взято три моделі ізоляції, що найчастіше зустрічаються на практиці. Database-per-Tenant дає кожному клієнту повністю ізольований екземпляр бази даних, але при кількох сотнях тенантів перетворюється на операційний тягар [2]. Schema-per-Tenant розміщує всіх у спільній базі, виділяючи кожному власний namespace [3]. Shared Schema з RLS розміщує всі тенанти в одних таблицях, а розмежування закладено в політики самої СУБД [4].

Моделі оцінювалися за чотирма атрибутами якості згідно ISO/IEC 25010: ізоляція тенантів, масштабованість, ресурсоємність інфраструктури та складність супроводу схем. Паралельно проводилося прототипування на стеку PostgreSQL + Prisma ORM.

Schema-per-Tenant добре справляється до кількох тисяч тенантів і при цьому не вимагає складної логіки на рівні запитів. Міграції є проблемними, тому будь-яка зміна схеми має бути застосована до кожної схеми окремо, що при великій кількості тенантів суттєво збільшує час розгортання [3].

Shared Schema з RLS – найбільш масштабована модель і при цьому найменш витратна в обслуговуванні. PostgreSQL забезпечує надійну ізоляцію через RLS-політики безпосередньо на рівні СУБД, що знімає цей обов'язок з прикладного коду. Додаткова складність виникає при використанні Prisma ORM, який не підтримує RLS нативно, тому контекст доводиться встановлювати вручну через Prisma Client Extensions або прямі SQL-виклики.

Модель ізоляції даних – це рішення, прийняте на старті, визначає, як система буде поводитися під навантаженням, скільки коштуватиме її обслуговування і наскільки болісними виявляться міграції. Кожна з трьох розглянутих моделей є обґрунтованою відповіддю на різні набори вимог, але жодна не є універсально кращою.

Для більшості масштабованих SaaS-продуктів модель Shared Schema з RLS є найраціональнішим вибором. При роботі з Prisma ORM це означає явну реалізацію механізму встановлення та скидання RLS-контексту, оскільки нативної підтримки бібліотека не надає.

Список використаних джерел:

1. Chong F., Carraro G., Wolter R. Multi-Tenant Data Architecture. MSDN Library. Microsoft, 2006. URL: https://learn.microsoft.com/en-us/azure/architecture/guide/multitenant/overview (дата звернення: 10.03.2026).

2. Bezemer C.-P., Zaidman A. Multi-tenant SaaS applications: maintenance dream or nightmare? Proceedings of the Joint ERCIM Workshop on Software Evolution, 2010. P. 88–92.

3. Kabbedijk J., Jansen S. Variability in Multi-Tenant Environments: Architectural Design Patterns from Industry. Advances in Conceptual Modeling, Lecture Notes in Computer Science. Springer, 2011. Vol. 6999. P. 151–160.

4. PostgreSQL Global Development Group. Row Security Policies. PostgreSQL 16 Documentation. 2023. URL: https://www.postgresql.org/docs/current/ddl-rowsecurity.html (дата звернення: 10.03.2026).