DOI: https://doi.org/10.36910/6775-2524-0560-2025-61-14
УДК: 004.415.2:004.77
Кошелюк Віктор Андрійович, к.т.н., доцент
https://orcid.org/0000-0002-4136-5087
Корень Віталій Володимирович
https://orcid.org/0009-0001-3732-1639
Тимчук Владислав Володимирович, магістр
Луцький національний технічний університет, м. Луцьк, Україна

АРХІТЕКТУРНІ МОДЕЛІ МУЛЬТИТЕНАНТНОСТІ В СУЧАСНИХ SAAS-СИСТЕМАХ: ПОРІВНЯЛЬНИЙ АНАЛІЗ ТА МЕТОДОЛОГІЯ ВИБОРУ

Кошелюк В. А., Корень В. В., Тимчук В. В. Архітектурні моделі мультитенантності в сучасних SaaS-системах: порівняльний аналіз та методологія вибору. У сучасній індустрії програмного забезпечення, коли доволі сильно домінує модель «Програмне забезпечення як послуга» (або SaaS), мультитенантність стала фундаментальним архітектурним принципом, що дозволяє досягти економічної ефективності та масштабованості. Вибір оптимальної моделі мультитенантності є одним із найкритичніших стратегічних рішень, яке безпосередньо впливає на вартість, безпеку, продуктивність та операційну складність та спроможність системи. Неправильний вибір може призвести до серйозних наслідків, таких як можливий витік даних між орендарями (тенантами), зниження продуктивності через проблему «шумного сусіда» або створення нежиттєздатної бізнес-моделі загалом. Існуюча наукова література детально описує класичні моделі (Silo, Pool), однак часто розглядає їх у контексті монолітних додатків, що є недостатнім для сучасних тенденцій у розробці програмного забезпечення. Зі зростанням складності програмних продуктів виникла потреба у більш гнучкому підході, де різні компоненти в межах одного додатку можуть вимагати різних стратегій залежно від їхніх функціональних та нефункціональних вимог. Ця стаття пропонує систематизований порівняльний аналіз основних архітектурних моделей мультитенантності: від повної ізоляції тенантів (модель Silo) до різних типів спільного використання ресурсів («База даних на тенант», «схема на тенант», «спільна схема на всіх тенантів») та інших, гібридних підходів. Аналіз проводиться за багатьма вагомими критеріями, що включає ізоляцію даних, вартість на тенанта, масштабованість, продуктивність, складність розробки та можливість персоналізації. Наукова новизна роботи полягає у розробці обґрунтованої методології вибору оптимальної стратегії, представленої у вигляді структурованого набору критеріїв, які добре підходять для прийняття рішень. Ці критерії допомагають архітекторам ПЗ узгодити технічні рішення з вимогами бізнесу, такими як ринок збуту, безпека (GDPR, HIPAA), цінова політика та інші вимоги.
Ключові слова: мультитенантність, SaaS, хмарні послуги, програмне забезпечення як послуга, архітектура програмного забезпечення, ізоляція даних, масштабованість, архітектурні шаблони, хмарні додатки, проектування систем.

Kosheliuk V., Koren V., Tymchuk V. Architectural models of multitenancy in modern SaaS systems: a comparative analysis and selection methodology. In the modern software industry, where the "Software as a Service" (SaaS) model is highly dominant, multitenancy has become a fundamental architectural principle for achieving cost-effectiveness and scalability. The selection of an optimal multitenancy model is one of the most critical strategic decisions, directly influencing the system's cost, security, performance, operational complexity, and overall capability. An incorrect choice can lead to severe consequences, such as potential data leakage between tenants, performance degradation due to the "noisy neighbor" problem, or the creation of an unsustainable business model altogether. Existing scientific literature provides detailed descriptions of classic models (Silo, Pool); however, it often considers them within the context of monolithic applications, which is insufficient given current software development trends. With the increasing complexity of software products, a need has emerged for a more flexible approach, where different components within a single application may require different strategies depending on their functional and non-functional requirements. This research offers a systematic comparative analysis of the primary architectural models of multitenancy: from complete tenant isolation (the Silo model) to various types of resource sharing ("Database-per-tenant," "Schema-per-tenant," "Shared Schema for all tenants") and other hybrid approaches. The analysis is conducted across many significant criteria, including data isolation, cost per tenant, scalability, performance, development complexity, and the potential for personalization. The scientific novelty of this work lies in the development of a well-founded methodology for selecting the optimal strategy, presented as a structured set of criteria well-suited for decision-making. These criteria help software architects align technical solutions with business requirements, such as the target market, security regulations (GDPR, HIPAA), pricing models, and other demands.
Keywords: multitenancy, SaaS, cloud services, software as a service, software architecture, data isolation, scalability, architecture design patterns, cloud applications, system design.

Постановка проблеми. Протягом останніх двох десятиліть ми стали свідками «зсуву» у розробці готового рішення програмного забезпечення, а саме переходу більшості ПЗ від традиційної локальної моделі «з коробки» до моделей на основі «Програмного забезпечення як послуги» (SaaS). Цей перехід зумовлений вагомими економічними факторами як для постачальників, так і для клієнтів. Для клієнтів SaaS усуває значні початкові капітальні витрати на ліцензії на програмне забезпечення та обладнання, замінюючи їх на попередньо налаштовані рішення. Для постачальників це забезпечує стабільний потік користувачів, спрощує оновлення програмного забезпечення та дозволяє збирати аналітичні дані про використання. Ця модель стала сучасним стандартом видання

різного програмного забезпечення, від систем управління взаємовідносинами з клієнтами (CRM) до інструментів для спільної роботи.

В основі масштабованого та економічно-життєздатного SaaS-додатку лежить архітектура мультитенантності (multitenancy). Мультитенантність – це архітектура, в якій один екземпляр програмного додатку обслуговує кількох клієнтів, яких називають тенантами (tenants), або орендарями. Об'єднуючи ресурси, такі як обчислювальна потужність, пам'ять та сховище, і розподіляючи їх між кількома тенантами, провайдери можуть досягти значної економії на масштабі. Цей спільний доступ до ресурсів є основним механізмом, що знижує вартість підтримки та створює можливість конкуренції на ринку, роблячи модель SaaS привабливою. Таким чином, ефективна реалізація мультитенантності безпосередньо пов'язана з критично важливим практичним завданням побудови сталого та прибуткового бізнесу «у хмарі».

Однак реалізація мультитенантності є серйозним архітектурним викликом. Основна проблема полягає не в тому, чи впроваджувати мультитенантність, а в ефективному проектуванні системи. Вибір конкретної моделі мультитенантності є одним із найважливіших рішень у життєвому циклі SaaS-продукту, що має глибокі та довготривалі наслідки для структури витрат, основ ізоляції даних користувачів (тенантів), продуктивності додатку, масштабованості та операційної складності. Невідповідний вибір архітектури додатку що розробляється, може призвести до катастрофічних наслідків, включаючи порушення безпеки через погано розглянуту ізоляцію даних, сильне падіння продуктивності через проблему «шумного сусіда» (коли високе навантаження одного тенанта негативно впливає на інших) або нестійку модель витрат, що руйнує прибутковість розроблюваного ПЗ. Це рішення виходить за рамки чистої технології і це стає фундаментальним рішенням бізнес-стратегії. Недорога модель зі спільним доступом може дозволити втілення моделі «freemium» (можливість користування ПЗ без додаткових оплат) або низьку ціну для масового ринку, тоді як добре ізольована, дорога модель необхідна для обслуговування корпоративних клієнтів у регульованих галузях, таких як фінанси чи охорона здоров'я, які вимагають суворих гарантій безпеки та відповідності до законів та стандартів (наприклад, GDPR, HIPAA). Таким чином, технічна архітектура безпосередньо розширює, або навпаки, обмежує цільовий ринок компанії, її цінову стратегію та конкурентоздатність.

Аналіз досліджень та публікацій Концепція мультитенантності добре відома як в академічній літературі, так і в галузевій практиці. У фундаментальних роботах визначено основні принципи та архітектурні шаблони. Наприклад, у дослідженні Чонґа, Карраро та Вольтера [1] розглянуто основні моделі тенантності, надавши ранню основу для оцінки компромісів між ізоляцією тенантів та ефективністю. Ця робота, разом із базами знань великих хмарних провайдерів, таких як Microsoft Azure та Amazon Web Services [2], встановила загальну термінологію моделей Silo та Pool, які слугують основою для більшості сучасних реалізацій. Ці ресурси заклали підґрунтя для розуміння мультитенантності як спектру вибору, а не єдиного, монолітного підходу [3].

Значна частина досліджень була зосереджена на рівні організації структури зберігання даних, оскільки він є найскладнішим і найкритичнішим аспектом ізоляції орендарів. Ретельно аналізуються три основні моделі тенантності баз даних: «База даних на тенант», «Схема на тенант» та «Спільна схема» зі стовпцем-дискримінатором. Наявні дослідження порівнювали ці підходи, виявляючи чіткий компроміс. Модель «База даних на тенант» пропонує найсильнішу ізоляцію та найпростіше управління даними для кожного тенанта (наприклад, резервне копіювання та відновлення), але несе найвищі витрати ресурсів і може бути обмежена кількістю баз даних, які може розмістити сервер. У той час, модель «Спільної схеми» є найбільш економічно ефективною та щільною з точки зору ресурсів, але вносить значну складність на рівні додатку для забезпечення сегрегації даних і несе найвищий ризик витоку даних між тенантами при неправильній реалізації.

Однією з найчастіше цитованих проблем у мультитенантних системах є проблема «шумного сусіда», коли діяльність одного тенанта використовує велику кількість спільних ресурсів, обмежуючи можливу витрату ресурсів для інших. Це питання було предметом численних досліджень. Дослідження виявили його першопричини у конкуренції за процесорний час, пам'ять, об'єм мережевого обміну даними та блокування баз даних. Запропоновані рішення варіюються від впровадження політик управління ресурсами та обмеження запитів на рівні додатку до використання передових алгоритмів планування та функцій розподілу ресурсів, доступних у сучасній хмарній інфраструктурі та системах баз даних. Ці дослідження підкреслюють, що в будь-якій моделі зі спільними ресурсами передбачуваність продуктивності не може бути гарантована без явних стратегій пом'якшення.

Метою роботи є систематизація та порівняльний аналіз основних архітектурних моделей мультитенантності з використанням аналізу за багатьма критеріями з метою розробки обґрунтованої методології вибору оптимальної стратегії в контексті сучасних хмарних SaaS-додатків. Це включає не лише переоцінку класичних моделей, але й їх інтеграцію в процес прийняття рішень, який включає в себе розгляд головних критеріїв реалізації програмного забезпечення.

Виклад основного матеріалу. В основі архітектури практично будь-якого сучасного ПЗ на базі моделі програмного забезпечення як послуги (SaaS) лежить фундаментальне питання: як ефективно та безпечно обслуговувати кількох клієнтів, або орендарів чи «тенантів», за допомогою єдиної програмної платформи. Це рішення, відоме як вибір моделі тенантності (tenancy model), і є одним із найважливіших архітектурних рішень, яке визначає не лише технічну реалізацію, але й економічну модель, операційну складність та потенціал масштабування продукту. Від даного вибору залежить здатність системи забезпечувати належний рівень ізоляції даних, продуктивності та можливості персоналізованого налаштування для кожного клієнта, водночас оптимізуючи витрати на інфраструктуру та управління процесами.

Вибір архітектурної моделі тенантності завжди є компромісом між двома цілями: максимальною ізоляцією ресурсів та максимальною ефективністю їх використання. На одному кінці спектру знаходиться повна ізоляція, що гарантує найвищий рівень безпеки та можливості персонального налаштування, але за рахунок значних інфраструктурних та операційних витрат. На іншому – повне об'єднання ресурсів, що дозволяє досягти значної економії та спростити управління, але висуває на передній план складні завдання щодо забезпечення безпеки даних та уникнення взаємного впливу («проблема шумного сусіда»). Не існує універсального чи правильного підходу, оскільки оптимальна стратегія залежить від бізнес-вимог, вибору клієнтів, нормативних зобов'язань та очікуваного масштабу. Тому розглянемо ключові моделі, що лежать у цьому спектрі, починаючи з найпростішої та найбільш ізольованої.

Модель Silo. У моделі Silo кожному тенанту надається повністю незалежний, виділений стек інфраструктури. Це включає власні сервери, приватну базу даних та, можливо, інші виділені, власні ресурси, такі як кеш або керування чергами. Архітектурно це найпростіша модель для уявлення, оскільки вона фактично передбачає розгортання окремої копії всього додатку для кожного нового клієнта.

Основною перевагою моделі Silo є її неперевершена безпека та ізоляція даних. Оскільки ресурси не є спільними, немає можливості витоку даних між тенантами або взаємного впливу на продуктивність. Ця повна ізоляція робить її вибором за замовчуванням для додатків, що обслуговують коштовних корпоративних клієнтів у суворо регульованих галузях (наприклад, фінанси, охорона здоров'я, уряд), де відповідність стандартам, таким як HIPAA або GDPR, не підлягає обговоренню, а вимоги щодо ізоляції даних повинні суворо дотримуватися. Крім того, ця модель пропонує найвищий ступінь кастомізації (персоналізації), оскільки виділене середовище тенанта можна змінювати або розширювати, не впливаючи на інших клієнтів. Однак недоліки є серйозними. Вартість на одного тенанта надзвичайно висока через відсутність об'єднання ресурсів, що робить її економічно невигідною для більшості ринків B2C (Бізнес-Клієнт) або SMB (Бізнес малого або середнього масштабу). Управління операціями також є дуже складним, оскільки середовище кожного тенанта потрібно надавати, спостерігати та оновлювати індивідуально. Підключення тенанта зазвичай є повільним, ручним процесом, що перешкоджає швидкому масштабуванню.

Модель Pool представляє протилежний підхід, де один екземпляр додатку та його базова інфраструктура обслуговують кількох тенантів одночасно. Економічні переваги цієї моделі є значними завдяки максимальному використанню ресурсів. Однак вона вносить критичний виклик забезпечення ізоляції даних та управління розподілом ресурсів у спільному середовищі. Стратегія досягнення цього, особливо на рівні даних, визначає окремі підмоделі.

Стратегія «База даних на тенант». Ця стратегія є компромісом між моделлю Silo та повністю спільними моделями. Хоча тенанти ділять сервери додатків, кожному з них призначається окрема, приватна база даних у межах спільного сервера баз даних або хмарного сервісу зберігання даних (наприклад, Amazon RDS, Azure SQL Database). Цей підхід забезпечує сильну ізоляцію даних, оскільки база даних одного тенанта не впливає на інших. Він також спрощує завдання управління даними для конкретного тенанта, такі як виконання резервного копіювання або відновлення даних для одного клієнта. Однак він зберігає частину вартості та складності моделі Silo. Кожна база даних

споживає базову кількість пам'яті та обчислювальних ресурсів, що призводить до вищих фінансових витрат. Крім того, більшість серверів для зберігання та керування даними мають практичні або ліцензійні обмеження на кількість баз даних, які вони можуть розміщувати, що може обмежувати масштабованість додатку за кількістю тенантів.

Стратегія «Схема на тенант». У цьому підході всі тенанти ділять один екземпляр бази даних, але дані кожного тенанта розміщуються у власному приватному наборі таблиць, організованих у виділену схему (або простір імен, залежно від системи баз даних). Ця модель досягає кращого балансу між ізоляцією та вартістю [4]. Вона пропонує хорошу логічну ізоляцію даних і все ще дозволяє відносно прості операції для конкретного тенанта (наприклад, модифікації схеми, експорт даних). Накладні витрати на ресурси на одного тенанта нижчі, ніж у моделі «База даних на тенант», оскільки накладні витрати самого рушія бази даних є спільними. Основним недоліком є те, що всі тенанти ділять ті самі базові обчислювальні ресурси, пам'ять та ємність потоку даних сервера баз даних, що збільшує ризик проблеми «шумного сусіда» [5]. Не всі системи баз даних пропонують надійну підтримку безпеки та управління ресурсами на рівні схеми, що може додати складності в реалізації.

Стратегія «Спільна схема, спільна база даних». Це найпоширеніша і найбільш економна щодо використаних ресурсів модель мультитенантності. Всі тенанти ділять одну базу даних і один набір таблиць. Для розділення даних до кожної таблиці, що містить дані конкретного тенанта, додається стовпець-дискримінатор, зазвичай названий «tenant_id». Кожен запит до бази даних, що виконується додатком для читання, запису, чи видалення даних – повинен містити умову «WHERE tenant_id = ?», щоб гарантувати, що він працює лише з даними поточного тенанта.

Економічна ефективність цієї моделі є її найбільшою перевагою, що дозволяє досягти найвищої щільності тенантів та найнижчої вартості на тенанта [6]. Однак вона вносить значну складність у розробку та експлуатацію. Ризик витоку даних є найвищим, оскільки одна помилка в програмуванні (наприклад, забута умова «WHERE») може розкрити дані одного тенанта іншому. Проблема «шумного сусіда» також є найгострішою в цій моделі, оскільки всі тенанти конкурують за ті самі ресурси бази даних на рівні таблиць. Нарешті, кастомізації для конкретного тенанта, такі як додавання власного поля, надзвичайно важко реалізувати, не впливаючи на всіх інших тенантів.

Гібридна модель «Міст». Bridge модель – це не окремий технічний патерн, а скоріше стратегічне поєднання інших моделей. У цьому підході SaaS-провайдер пропонує різні моделі тенантності для різних рівнів клієнтів. Наприклад, стандартний або безкоштовний тарифний план може розміщувати тенантів у економічно ефективному пулі зі спільною схемою. Преміум або корпоративні клієнти, які платять вищу ціну і вимагають більших гарантій безпеки та продуктивності, можуть бути розміщені в приватній моделі «База даних на тенант» або навіть у повністю виділеному Silo.

Ця модель надає практичне, бізнес-орієнтоване рішення, яке узгоджує архітектурні витрати з цінністю для клієнта та його готовністю платити. Вона дозволяє провайдеру обслуговувати різноманітний ринок за допомогою одного продукту. Основною проблемою є значне збільшення операційної складності. Провайдер повинен створювати та підтримувати інструменти для управління середовищем, включаючи автоматизований моніторинг та міграцію тенантів між різними моделями (наприклад, коли клієнт оновлює свій тарифний план). Для систематизації процесу вибору, архітектурні моделі можна оцінити за узгодженим набором критеріїв. Основні критерії для порівняння включають: ізоляція та безпека даних, вартість на тенанта, масштабованість (як за кількістю тенантів, так і за навантаженням на тенанта), продуктивність та конкуренція за ресурси, складність розробки та підтримки, та можливість персонального налаштування для тенанта. Результати цього аналізу зведені в Таблиці 1.

Табл. 1. Порівняльний аналіз моделей мультитенантності

Архітектурна модель	Ізоляція даних	Вартість на тенанта	Масштабованість	Конкуренція за ресурси	Складність розробки	Можливість кастомізації
Silo	Повна	Дуже висока	Низька (кількість тенантів) / Висока (на тенанта)	Відсутня («шумний сусід» усунений)	Низька	Висока
Pool: База даних на тенант	Висока	Висока	Середня	Низька	Середня	Середня

Pool: Схема на тенант	Середня	Середня	Висока	Середня	Середньо-висока	Низька
Pool: Спільна схема	Низька	Дуже низька	Дуже висока	Висока (значний ризик «шумного сусіда»)	Висока	Дуже низька
Гібридна модель	Змінна	Змінна	Висока	Змінна	Дуже висока	Змінна

Параметри аналізу, представлені в таблиці, виявляють фундаментальні компроміси, властиві архітектурі мультитенантності. Найбільш значущим шаблоном є зворотна залежність між вартістю та ізоляцією. Модель Silo забезпечує ідеальну ізоляцію за найвищу ціну, тоді як модель «Спільної схеми» пропонує найнижчу вартість за рахунок мінімальної вбудованої ізоляції. Цей компроміс стосується не лише витрат на інфраструктуру: важливим наслідком, який часто ігнорується в попередніх аналізах, є перенесення складності. Моделі, які здаються дешевшими з точки зору інфраструктури, як-от підхід «Спільної схеми», перекладають величезний тягар складності на логіку додатку. Розробники повинні бути суворо дисциплінованими, щоб забезпечити, що кожна взаємодія з базою даних враховує орендаря, а для пом'якшення впливу на продуктивність потрібні складні засоби контролю на рівні додатку. Це може значно збільшити час розробки, зусилля на тестування та підтримку програмного забезпечення, що розробляється, потенційно впливаючи на оцінку потрачених ресурсів на розробку проекту.

Висновки та перспектива подальших досліджень. У статі систематично розглянуто основні архітектурні моделі для реалізації мультитенантності в SaaS-системах. Основне спостереження полягає в тому, що не існує універсальної «найкращої» моделі, яку слід використовувати при розробці програмного забезпечення. Оптимальний вибір – це залежне від контексту рішення, яке вимагає обґрунтованої оцінки конкуруючих технічних та бізнес-вимог. Модель Silo пропонує максимальну безпеку та можливість налаштування під любий ринок для роботи. Різні моделі Pool пропонують спектр компромісів між економічною ефективністю та ізоляцією, причому модель «Спільної схеми» представляє найвищий ризик і найвищу винагороду. Запропонований порівняльний аналіз схем надає структуровану методологію для навігації по цих компромісах, забезпечуючи відповідність обраної архітектури ринку продукту, його бізнес-моделі та регуляторному ландшафту.

Сфера мультитенантних систем постійно розвивається разом із ширшим ландшафтом хмарних обчислень. Існує кілька перспективних напрямків для майбутніх досліджень. По-перше, необхідне глибше дослідження характеристик продуктивності та вартості шаблонів мультитенантності в безсерверних архітектурах (наприклад, AWS Lambda, Azure Functions), оскільки природа цих платформ вносить нові виклики та можливості для управління ресурсами. По-друге, коли програмне забезпечення вже знаходиться у етапі підтримки, можна застосувати аналіз даних ресурсоємності для пом'якшення проблеми «шумного сусіда», завдяки виведення проблемних тенантів у власні схеми, що може ізолювати їх від інших тенантів. По-третє, використання нових технологій, таких як конфіденційні обчислення та шифрування, може бути досліджене як засіб забезпечення сильної криптографічної ізоляції даних навіть у повністю спільному середовищі бази даних, потенційно вирішуючи основну проблему безпеки моделі «Спільної схеми». Також, є потреба в узагальнених фреймворках для автоматизації складного життєвого циклу орендаря, включаючи безшовне підключення та міграцію між моделями тенантності при зміні тарифного плану.

Список бібліографічного опису

1. Chong F., Carraro G., Wolter R. Multi-Tenant Data Architecture. Microsoft Corporation. 2006. URL: https://surl.li/iqnhyy (дата звернення: 29.08.2025).
2. Silo, Pool, and Bridge Models - SaaS Lens. URL: https://surl.li/hvpmtm (дата звернення: 02.09.2025).
3. Multitenant SaaS Patterns - Azure SQL Database | Microsoft Learn. 2025. URL: https://surl.li/nyiqdi (дата звернення: 03.09.2025).
4. A. Aulbach et al. Multi-tenant databases for software as a service: schema-mapping techniques. SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. 2008. P. 1195-1206.
5. Krebs, R., Momm, C., & Kounev, S. A Tenancy-Aware Performance Evaluation Approach for Database-as-a-Service Systems. Proceedings of the 8th ACM/SPEC International Conference on Performance Engineering. 2017. P. 15-26.
6. Schad, J., Dittrich, J., & Quiané-Ruiz, J. A. Runtime optimization of multi-tenant database schemas. Proceedings of the VLDB Endowment. 2012. Vol. 5, No. 11. P. 1475-1486.

References

1. Chong F., Carraro G., Wolter R. Multi-Tenant Data Architecture. Microsoft Corporation. 2006. URL: https://surl.li/iqnhyy (date of access: 29.08.2025)
2. Silo, Pool, and Bridge Models - SaaS Lens. URL: https://surl.li/hvpmtm (date of access: 02.09.2025).
3. Multitenant SaaS Patterns - Azure SQL Database | Microsoft Learn. 2025. URL: https://surl.li/nyiqdi (date of access: 03.09.2025).
4. A. Aulbach et al. Multi-tenant databases for software as a service: schema-mapping techniques. SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. 2008. P. 1195-1206.
5. Krebs, R., Momm, C., & Kounev, S. A Tenancy-Aware Performance Evaluation Approach for Database-as-a-Service Systems. Proceedings of the 8th ACM/SPEC International Conference on Performance Engineering. 2017. P. 15-26.
6. Schad, J., Dittrich, J., & Quiané-Ruiz, J. A. Runtime optimization of multi-tenant database schemas. Proceedings of the VLDB Endowment. 2012. Vol. 5, No. 11. P. 1475-1486.