УДК 62-752-62-755: 621.634

А.А. БУРЛАКОВ
Хмельницький національний університет

ОСОБЛИВОСТІ ВИБОРУ ПОСТІЙНОГО СХОВИЩА ДАНИХ ПІД ЧАС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА JAVA-ПЛАТФОРМІ

Проаналізовано потреби ринку розробки програмного забезпечення в сучасних мовах та технологіях програмування. Показано доцільність використання об'єктного підходу при розробці сучасних програмних продуктів. Досліджено різні моделі даних та типи систем управління базами даних. Проведено аналіз ринку сучасних об'єктно-орієнтованих СУБД.

Ключові слова: мова програмування java, база даних, система управління базами даних, розробка програмного забезпечення, моделі даних, об'єктний підхід до розробки.

A.A. BURLAKOV
Khmelnytskyi National University

CHOICE A DATA STORAGE DURING SOFTWARE DEVELOPMENT ON JAVA-PLATFORM

Analyzes the market of modern languages and programming technologies. Established that the Java programming language last six years is the most popular language for software development. We discuss about object-oriented approach in the development of modern software. In the article we can see the various data models and database management systems such as flat, hierarchical, network, document-oriented, relational, object-oriented, object-relational. It shows the advantages and disadvantages of these data models. It was found the advantages of object-oriented databases, which in the design of certain types of applications are the most important We have analyzed the rating object-oriented databases.

Keywords: java, database, database management system, software development, data model, object approach to development.

Якщо переглянути результати аналізу рейтингу різних мов програмування на ринку розробки програмного забезпечення (ПЗ) за 2015 рік та за минулі шість років (рисунки 1–3), проведеного авторитетним аналітичним інтернет-ресурсом dou.ua [1], то можна впевнено сказати, що мова java, та технології на її основі, стабільно займають перше місце в рейтингу і ця тенденція є незмінною за останні шість років.

Такої широкої популярності серед розробників програмного забезпечення мова програмування java набула завдяки своїй кросплатформності (незалежності написаного коду від програмно-апаратного забезпечення, на якому цей код буде виконуватись), безперервному оновленні з врахуванням сучасних тенденцій в програмуванні, наявності «Загальної публічної ліцензії» (GNU) [2]. Сучасні java-технології дозволяють вирішувати переважну більшість задач програмістів, починаючи з розробки невеличких додатків для портативних пристроїв і закінчуючи великими корпоративними системами.

В наш час найбільшої популярності серед java-програмістів набув об'єктно-орієнтований підхід до розробки програмних систем. Цей підхід передбачає наступний спрощений сценарій розробки: виявляється функціонал майбутньої системи (сукупність прецедентів) та актори, які задіяні в його роботі; для кожного прецеденту виявляються об'єкти, які в процесі взаємодії забезпечують логіку виконання прецеденту; створюється спочатку структура системи в рамках кожного варіанта використання; потім розробляється структура всієї системи з врахуванням переваг, що надають архітектурні та дизайнерські шаблонні рішення; відбувається реалізація системи з використанням оптимальних технологій програмування. Але в цьому процесі центральною ланкою є об'єкт з його станами і поведінкою. Серед різних типів об'єктів, що беруть участь в реалізації прецедентів, можна виділити об'єкти, призначення яких полягає в інкапсуляції даних предметної області. Такі об'єкти носять назву об'єкти-сутності (entity) і капсульовані в них дані мають бути узгоджені з постійними сховищами даних (бази даних, файли, тощо). Вибір найбільш оптимального типу сховища та конкретного представника цього типу для java-додатка, стає не тривіальною задачею.

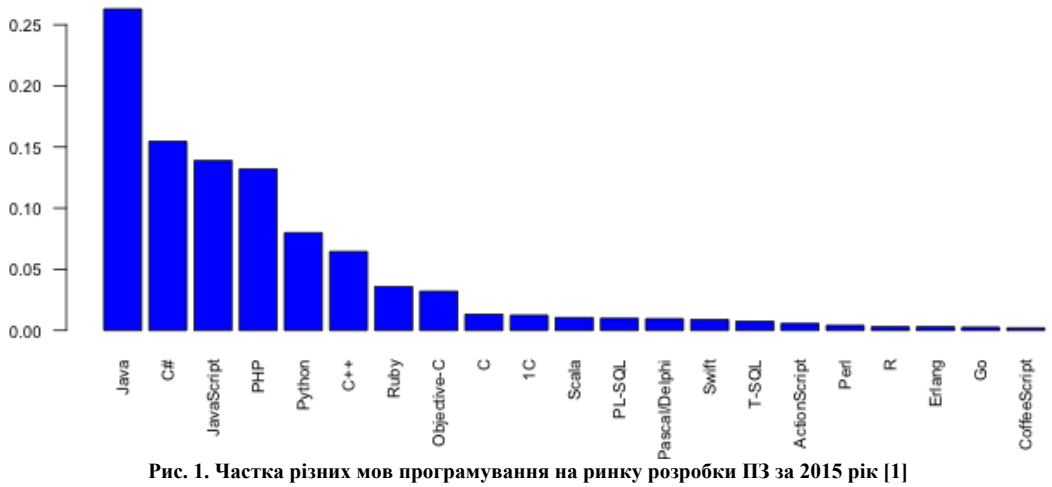

Рис. 1. Частка різних мов програмування на ринку розробки ПЗ за 2015 рік [1]

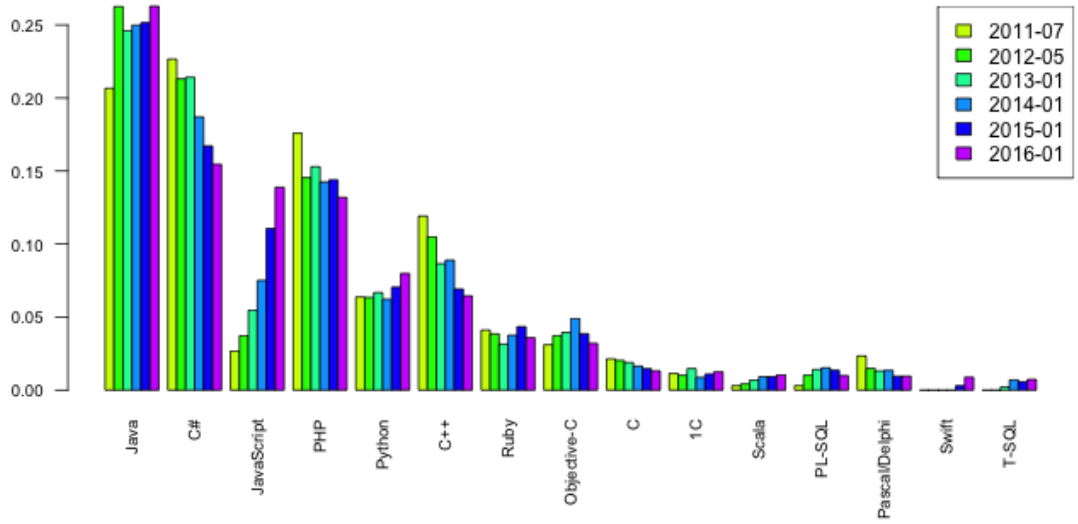

Рис. 2. Динаміка зміни частки різних мов програмування на ринку розробки ПЗ за 2011–2016 роки [1]

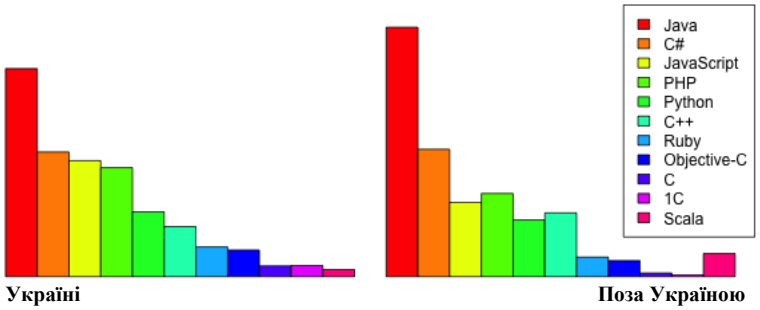

Рис. 3. Розповсюдженість різних мов програмування серед розробників ПЗ в Україні та поза її межами [1]

Метою роботи був аналіз різних типів сховищ даних та ефективність їх використання для моделювання різних видів предметних областей об'єктами-сутностями.

В переважній кількості випадків розроблювані комп'ютерні системи працюють з інформацією, яку забезпечують системи управління базами даних (СУБД). Усі існуючі бази даних можна розділити за їх моделями даних, які описують те, як дані будуть збережені, організовані та яким чином ми можемо ними маніпулювати. До основних моделей баз даних (БД) належать: плоска, ієрархічна, мережева, документо-орієнтована, реляційна, об'єктно-орієнтована, об'єктно-реляційна [3, 4].

Для характеристики вищезазначених моделей даних з точки зору надійності і передбачуваності системи транзакцій, можна використати акронім ACID, який описує: атомарність, узгодженість, ізольованість та довговічність. Згідно з цими принципами ми отримуємо надійну та контрольовану транзакційну систему. На рисунку 4 зображено, які моделі баз даних відповідають вимогам ACID повністю, а які тільки частково.

Рис. 4. Відповідність вимогам ACID моделей баз даних

Як бачимо, вимогам ACID повністю відповідають: реляційна, об'єктно-орієнтована та об'єктно-реляційна моделі бази даних. Інші моделі якщо й і відповідають ACID, то не повністю. Тому ті моделі, які не відповідають вимогам ACID не доречно використовувати для систем, яким потрібна надійна збереженість даних та їх цілісність. Коротко покажемо, що представляють собою моделі даних, які відповідають вимогам ACID [5].

Реляційна модель – це одна із найпопулярніших моделей на ринку баз даних. Головним елементом даної моделі є таблиця. Таблиця представляє собою окремий об'єкт, який має свої атрибути (в таблиці це поля). Кожна стрічка в таблиці це окремий запис. Кожна таблиця має свій первинний ключ, який виражається у вигляді окремого поля таблиці, і містить унікальні ідентифікатори записів. Ці ідентифікатори потрібні для розрізнення записів між собою та організації зв'язків. Зв'язки в реляційній моделі можуть бути трьох видів: "олин-до-одного", "один-до-багатьох" та "багато-до-багатьох". Зв'язок "один-до-одного" реалізується шляхом зв'язку двох первинних ключів двох таблиць. "Один-до-багатьох" реалізується шляхом додавання додаткового поля (вторинний ключ) до таблиці із сторони "багато" – це поле буде зв'язане із первинним ключем таблиці із сторони "одного". Зв'язок "багато-до-багатьох" реалізується шляхом

додавання ще однієї таблиці де будуть зберігатись вторинні ключі для зв'язку із кожною таблицею. Можна відмітити такі сильні сторони реляційної моделі: має найбільш поширене використання на теперішній час; здійснює контроль над транзакціями; присутній багатокористувацький доступ до БД; наявність стандартів; ґрунтується на чітко сформованому математичному апараті; присутній контроль цілісності даних.

Об'єктно-орієнтована модель - це модель, в основу якої покладено принципи об'єктно-орієнтованого програмування. Згідно цієї моделі, дані представлені у вигляді класів та об'єктів, зв'язаних між собою зв'язками. Сукупність класів моделює предметну область. Об'єкти реалізуються класами, які є деяким шаблон, по якому будуються об'єкти. Кожен об'єкт характеризується набором характеристик. Кожна із характеристик містить якісь дані, що пасивно характеризують об'єкт. Для роботи з класом в класі міститься набір методів. Кожен метод є набором деяких операцій, які виконуються над об'єктом. Зв'язки між об'єктами забезпечуються завдяки методам та характеристикам, які мають класовий тип даних. Кожному об'єкту, при створенні, надається унікальний ідентифікатор (OID), який ніколи не змінюється підчас існування об'єкта.

Розглянемо, як виглядає процес розробки ПЗ при об'єктному підході, якщо використовувати реляційні та об'єктні моделі даних. Як було описано вище, реляційні БД побудовані на базі таблиць та трьох видів відношень між ними. Таким чином, щоб побудувати деяку структуру бази даних потрібно зібрати дані по предметній області і провести їх нормалізацію. В результаті цього ми отримаємо таку структуру даних, в якій інформація про окремий об'єкт може міститись в декількох таблицях, зв'язаних між собою певними відношеннями (рис. 5). Якщо ж використовувати об'єктно-орієнтований підхід до розробки, то дані організуються у вигляді окремих класів, а в результаті і об'єктів. А тепер уявімо, на скільки важко буде організувати перетворення даних в об'єкти і навпаки, при роботі із реляційною БД. І ще додати до цього витрачений процесорний час на перетворення. На відміну від реляційних БД, дані в об'єктно-орієнтованих БД зберігаються в об'єктах без перетворень, не витрачаючи зайвий раз процесорний час (рис. 6).

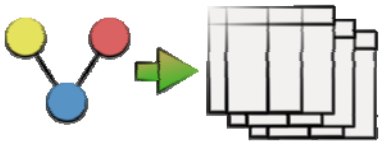

Рис. 5. Збереження даних в реляційних БД

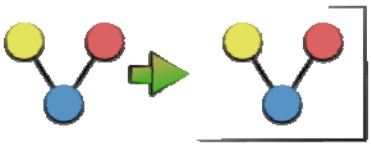

Рис. 6. Збереження даних в об'єктно-орієнтованих БД

На рисунках 7 та 8 показано процес розробки ПЗ з використанням реляційної та об'єктної бази даних відповідно. Як видно з рисунків, робота з об'єктно-орієнтованими БД зменшує час (або кількість етапів) розробки у двічі. Розробка самої БД займає на 40–80% менше часу ніж розробка реляційної БД.

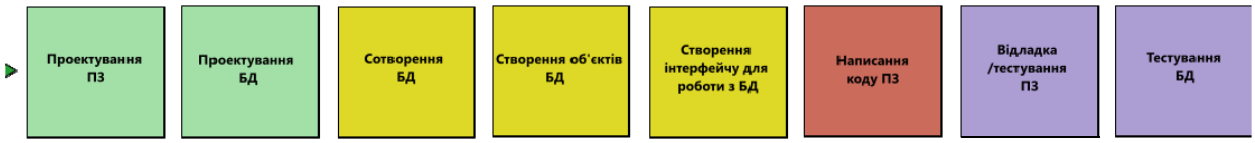

Рис. 7. Процес розробки програмного забезпечення із використанням реляційної БД

Рис. 8. Процес розробки програмного забезпечення з використанням об'єктно-орієнтованої БД

Крім того, об'єктно-орієнтована модель бази даних надає декілька цікавих особливостей, яких не може дати реляційна модель баз даних. До таких особливостей належать: інкапсуляція, наслідування та поліморфізм. В загальному, ці особливості надають можливість розрізняти дані у вигляді деяких сутностей, внутрішній механізм яких закритий від користувача. При розробці об'єктної моделі розробнику доступний механізм наслідування та перевизначення методів. В сумі все це пришвидшує розробку і полегшує модифікацію та експлуатацію об'єктно-орієнтованої бази даних. Таким чином, можна відмітити такі сильні сторони об'єктно-орієнтованої моделі: має найбільш поширене використання на теперішній час; повноцінна підтримка об'єктно-орієнтованого підходу; контроль цілісності даних зі сторони бази даних; інтеграція із об'єктно-орієнтованими мовами програмування без посередників; легке видалення даних; легка модифікація зв'язків між об'єктами; простота масштабування; велика швидкодія.

Об'єктно-реляційна модель представляє собою деяку гібридну модель із більш низькою продуктивністю роботи і об'єднує в собі реляційну базу даних та механізм перетворення даних в об'єкти. Саме механізм перетворення і відрізняє дану модель від реляційних баз даних. Механізм перетворення даних може знаходить як на стороні СУБД, так і на стороні клієнта. В першому випадку користувач буде отримувати дані у вигляді повноцінних об'єктів, які він легко може обробити, в другому випадку змушений

утворювати відображення об’єктів на таблиці сам з використанням більш повільної технології ORM (з англійської – об’єктно-реляційне відображення). Можна відмітити такі позитивні сторони об’єктно-реляційної моделі: більш тісна інтеграція з об’єктно-орієнтованими мовами програмування;, ніж у реляційних моделей; допомагає скоротити час при розробці; достатньо широка поширеність.

Існують деякі об’єктивні причини, що спонукають науковців та розробників ПЗ звертати все більше уваги в сторону використання об’єктних СУБД, незважаючи на таку широку розповсюдженість реляційних та об’єктно-реляційних моделей, а саме:

- обмеження реляційних та об’єктно-реляційних моделей;
- вимоги деяких видів сучасних додатків;
- популярність об’єктно-орієнтованої парадигми.

Спочатку розглянемо обмеження табличних моделей даних:

- погане представлення реальних об'єктів: реляційна модель не може представляти реальний світ в правильний шлях, бо має тільки одну семантику, яка базується на таблицях. Дуже важко іноді зрозуміти реальні зв’язки між об’єктами через «призму» таблиць;
- недостатньо повноцінна підтримка обмежень цілісності і виробничих обмежень (enterprise constraints), бо таблична модель підтримує тільки незначне число обмежень, в той час як виробничі обмеження визначаються галузевими стандартами;
- складність в обробці рекурсивних запитів: існує дуже погана підтримка для обробки рекурсивних запитів;
- при маніпулюванні даними через SQL відсутня обчислювальна повнота [7]. Щоб подолати цю ситуацію, необхідно вбудовувати SQL в конструкції високорівневих мов програмування, таких як Java;
- погана підтримка довготривалих транзакцій: в табличних СУБД транзакції «живуть» недовго і методи та механізми управління паралелізмом погані для довготривалих операцій;
- слабка підтримка еволюції структури: під еволюцією структури передбачається внесення змін до схеми бази даних в додаток під час виконання без переривання виконання;
- поганий навігаційний доступ: існує дуже заслабка підтримка навігаційного доступу в реляційних СУБД.

Наступним блоком розглянемо деякі сучасні види додатків, які не задовольняє таблична модель.

CAD-системи (системи автоматизованого проектування): у додатків такого виду в БД зберігаються дані про елементи будівель, машин, інтегральних мікросхем, а процес проектування табличної бази даних для них може бути дуже тривалим; процес проектування в цих типах додатків не є статичним, бо структура еволюціонує в часі, оновлення повинні бути розмножені; такі додатки вимагають контролю версій і управління конфігурацією; ці додатки вимагають складних (композитних) об'єктів при створенні конструкцій. Наприклад, компоненти літака можуть бути пов'язані з іншими компонентами; в цих додатках існує потреба в підтримці спільної розробки одного проекту.

CAM-системи (системи автоматизованого виробництва): дані таких додатків дуже схожі на дані в CAD-системах, але вимагають дискретного виготовлення: ці додатки повинні реагувати на події реального часу; алгоритми і вимоги замовника мають гнучко реагувати на різні ситуації.

CASE (Засоби автоматизації розробки програм): ці додатки управляють даними протягом різних фаз життєвого циклу розробки ПЗ; розробка може тривати екстремально довго і передбачає спільну роботу; є необхідність підтримувати залежності між компонентами.

Network Management Systems (Системи моніторингу мережі): ці системи використовуються для таких завдань, як управління шляхами мережі, проблемними ситуаціями і питаннями планування мережі.

Інші види додатків, такі як: Office Information Systems (Інформаційні системи автоматизації офісних задач), Multimedia Systems (Системи підтримки мультимедіа), Digital Publishing (Цифрове видавництво), GIS-системи (Geographic information Systems або Геоінформаційна система).

І, на останок, популяризація об’єктно-орієнтованої парадигми. Вона в кращий спосіб моделює реальні життєві ситуації навколишнього світу завдяки таким аспектам як: абстракція, інкапсуляція, наслідування, поліморфізм, композицію. Так, абстракція передбачає конструювання моделі об’єкта реального світу в такий спосіб, що виділяються і програмуються тільки ті властивості і поведінка оригіналу, які потрібні саме для виконання задач в даній предметній області. Інкапсуляція вимагає конструювати об’єкт у вигляді «капсули», яка приховує від інших об’єктів свій внутрішній вміст для забезпечення його цілісності. Механізм наслідування дозволяє утворювати програмні об’єкти з автоматично унаслідуваними від своїх «предків» наборами властивостей, поведінки та типів. Поліморфізм представляє собою механізм динамічного зв’язування, завдяки якому забезпечується «слабкий» зв’язок між програмними об’єктами через те, що вони взаємодіють один з одним через посилання більш абстрактного типу, ніж сам об’єкт, що приймає участь у цьому зв’язку. Композиція дозволяє створювати програмні об’єкти, які складаються з інших об’єктів без обмежень в рівні вкладеності (наприклад об’єкт автомобіля складається з об’єктів двигуна, колес, керма, тощо. Двигун, в свою чергу, складається із інших об’єктів, що імітують вузли та деталі двигуна).

Завдяки тому, що об’єктні СУБД є все більш затребувані на ринку розробки ПЗ, варто зупинитись на їх сучасній архітектурі. Існують наступні різновиди архітектури: «клієнт-серверна» (Client-server), та вбудована (Embedded).

Клієнт-серверна СУБД має окреме програмне забезпечення для роботи із БД і розміщується на одному сервері разом з нею. Запити до бази даних обробляються централізовано. Перевагами клієнт-серверних СУБД є зручність централізованого управління, висока надійність, безпека та доступність. В наш час архітектури об'єктно-орієнтованих клієнт-серверних СУБД поділяються на три підвиди:

- Object Server. В цьому випадку забезпечується середовище для розподіленої обробки між клієнтом і сервером. Як правило, сервер відповідає за функції управління об'єктно-орієнтованою БД, а за транзакції та інтерфейс з мовою програмування відповідає клієнтська частина (рис. 9).

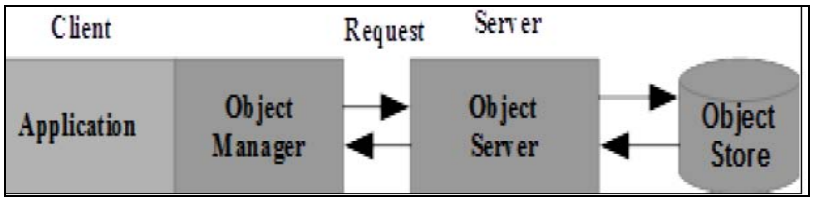

Рис. 9. Архітектура Object Server клієнт-серверної об'єктно-орієнтованої СУБД

- Page Server. У цій моделі клієнт-серверної архітектури клієнт відповідальний за роботу БД більшість часу. Вторинне управління зберіганням і реагуванням на запити здійснюється сервером. Сторінка може містити безліч комплексних об'єктів або звичайних об'єктів (рис. 10).

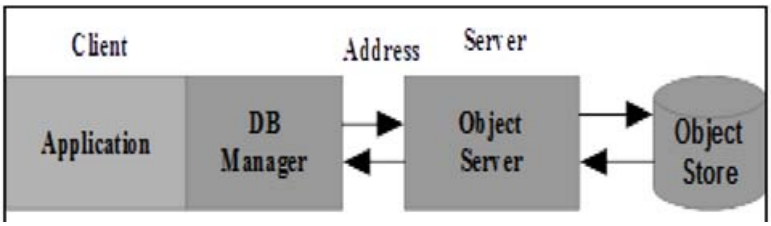

Рис. 10. Архітектура Page Server клієнт-серверної об'єктно-орієнтованої СУБД

- Database Server. При такому підході, клієнт просто передає запит на сервер, отримує результати і передає їх в додаток. Переважна частина обробки здійснюється на сервері (рис. 11).

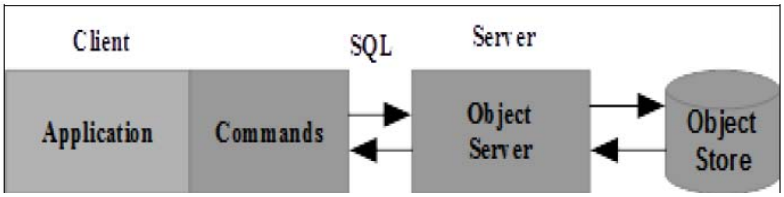

Рис. 11. Архітектура Database Server клієнт-серверної об'єктно-орієнтованої СУБД

Вбудована СУБД може використовуватись як одна із складових частин програмного продукту, який з нею працює. Може реалізовуватись у вигляді деякої бібліотеки, яку можна підключити до програмного продукту. Перевагами вбудованої СУБД є простота, швидкодія, відсутність потреби в адміністраторі БД.

При проведенні аналізу ринку об'єктних СУБД нас будуть цікавити ті представники цього типу, які підтримують зберігання java-об'єктів. Наведемо короткий список найпопулярніших представників у порядку спадання рейтингу [9].

Об'єктно-орієнтована СУБД Dv4o (або Db4object). Розробляється корпорацію Versant. Вона є вбудованою СУБД, але також може використовуватись як клієнт-серверна СУБД. Робота з даною базою даних є максимально простою. До складу вашого програмного продукту має увійти бібліотека, яка буде реалізовувати усі механізми роботи із файлом бази даних. Основні можливості: динамічний розвиток схеми; зберігання будь-яких об'єктів однією стрічкою коду; підтримка кешування об'єктів; компактність; наявність режиму «тільки для читання»; підтримка запитів на LINQ, NQ, QbE, SODA; підтримка паралельного доступу; підтримка відміни транзакції.

Вбудована об'єктно-орієнтована СУБД Perst від компанії McObject. Має такий перелік основних можливостей: компактність; наявність експорту та імпорту даних в формат XML; шифрування бази даних; швидке відновлення без log-файлів; підтримка кешування; динамічний розвиток схеми; наявність механізму стискання бази даних; підтримка управління транзакціями через саме програмне забезпечення; присутність багатокористувацького доступу; підтримка LINQ, JSQL, SQL. Однією із цікавих особливостей даної СУБД є те, що ви може керувати механізмом управління транзакціями із вашого програмного продукту.

СУБД ObjectDB. Вона може бути використана в режимі клієнт-сервер і як вбудована. На відміну від інших об'єктних СУБД, ObjectDB не надає свій власний API і є кросплатформною. СУБД обробляє більшість змін класів прозоро для програміста, включаючи додавання і видалення постійних полів, зміну типів постійних полів і зміну ієрархії класів.

Versant (VOD – Versant object database) – це клієнт-серверна БУБД, що розроблена корпорацією Versant. Має такі основні можливості: безшовне розподілення даних по декількох базах даних; висока швидкість збору даних; корпоративний рівень відмовостійкості; динамічний розвиток схеми; загальні логічні ідентифікатори об'єктів (LOID); дворівнева архітектура кешування даних; підтримка LINQ. Для даного програмного продукту було спеціально розроблено мову для запитів VQL (Versant Query Language). Але крім цього, запити можуть бути створені і за допомогою технології LINQ. Дана об'єктна СУБД використовується в дуже великій кількості галузей, таких як: телекомунікації, фінанси, оборона, урядові структури, медицина та ін.

Objectivity/DB – комерційна клієнт-серверна об'єктно-орієнтована СУБД компанії Objectivity, Inc. Має такі основні можливості: розподілена архітектура; динамічний розвиток схеми; експорт даних в XML; підтримка 32- та 64-розрядних архітектур; підтримка LINQ, PQE, EJPQL; підтримка паралельних запитів. Особливістю даної СУБД є те, що вона може організовувати розподілену архітектуру. При чому, забезпечує єдине логічне представлення 65000 баз даних, що суттєво спрощує роботу із цією архітектурою. Кожна із баз даних може займати мільйони терабайт. Objectivity/DB використовується в великих наукових центрах як CERN для накопичення великої кількості даних по результатам досліджень з прискорення елементарних частинок. Також дана СУБД використовувалась в експерименті BaBar, де зберігалось більше пентабайта інформації.

Таким чином, в роботі було проведено аналіз вимог ринку розробки програмного забезпечення в сучасних технологіях програмування. Показано, що протягом вже майже десяти років технології програмування на java-платформі залишаються самими затребуваними. Привели доцільність використання об'єктного підходу при розробці сучасних програмних продуктів, досліджено різні моделі даних та типи систем управління базами даних, що використовуються на ринку ПЗ. Показано, що застосування об'єктно-орієнтованих СУБД дозволяє значно (до 80%) скоротити терміни проектування баз даних; проведено аналіз сучасного ринку об'єктних СУБД.

Література

1. Рейтинг языков программирования // Аналітичний IT-ресурс [Електронний ресурс]. – Режим доступу : http://dou.ua/lenta/articles/language-rating-jan-2016/

2. General Public License // Вікіпедія [Електронний ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/GNU_General_Public_License.

3. Case-технологии. Современные методы и средства проектирования информационных систем // Аналітичний IT-ресурс [Електронний ресурс]. – Режим доступу : http://citforum.ru/database/case/index.shtml..

4. Системы управления базами данных // Аналітичний IT-ресурс [Електронний ресурс]. – Режим доступу : http://citforum.ru/database/dblearn/index.shtml.

5. Бурлаков А.А. Використання об'єктних систем управління базами даних при розробці програмного забезпечення / А.А. Бурлаков // Збірник тез доповідей VII Міжнародної науково-технічної конференції "Актуальні проблеми комп'ютерних технологій 2014". – Хмельницький, 2014. – С. 37–54.

6. Database System: A Practical Approach to Design, Implementation and Management, by T. Connoly and C. Begg., 5th Edition, 2010, 1400 p.

7. Bertino E., Negri M., Pelagatti G. and Sbattella L. "Object-Oriented Query Languages: The Notion and the Issues", IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 3, 1992.

8. Top object databases // Аналітичний IT-ресурс [Електронний ресурс]. – Режим доступу : http://www.decidesoftware.com/top-object-databases/

9. Системы управления базами данных // Вікіпедія [Електронний ресурс]. – Режим доступу : http://ru.wikipedia.org/wiki.

Рецензія/Peer review : 5.4.2016 р. Надрукована/Printed : 6.6.2016 р.
Рецензент: д.т.н., проф., Марченко В.Л.