Перелік посилань

1. Антонов С. Что такое чат-боты и зачем они нужны? [Електронний ресурс] / Святослав Антонов. – 2018. – Режим доступу до ресурсу: https://informburo.kz/cards/chto-takoe-chat-boty-i-zachem-oni-nuzhny.html.
2. Гатулин Р. Р. Использование мессенджера Telegram для реализации технологии электронного обучения в вузе [Електронний ресурс] / Р. Р. Гатулин, Д. А. Колупаева // Науки об образовании. – 2017. – Режим доступу до ресурсу: https://cyberleninka.ru/article/n/ispolzovanie-messendzhera-telegram-dlya-realizatsii-tehnologii-elektronnogo-obucheniya-v-vuze.
3. Интерактивное обучение [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://studme.org/157663/pedagogika/interaktivnoe_obuchenie.
4. Справочник по Bot API [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://tlgrm.ru/docs/bots/api.
5. Можаров М. С. Использование современных технологий в области интерактивного обучения программированию: тенденции и перспективы [Електронний ресурс] / М. С. Можаров // Вестник ТГПУ. – 2017. – Режим доступу до ресурсу: https://cyberleninka.ru/article/n/ispolzovanie-sovremennyh-tehnologiy-v-oblasti-interaktivnogo-obucheniya-programmirovaniyu-tendentsii-i-perspektivy/viewer.

Вдосконалення методу проєктування вебдодатків на основі об'єктно-реляційного перетворення

Мілер В.М., Орленко В.С.
Хмельницький національний університет

З метою покращення розробки вебдодатків проектувальники намагаються знайти рішення, яке дозволить швидко і ефективно використовувати компонентну базу існуючих бібліотек.

Найпоширенішою платформою для розробки вебдодатків є LAMP [1]. LAMP - абревіатура набору вільного ПЗ з відкритим кодом, в який входять ОС Linux, веб-сервер Apache, СКБД MySQL, та інтерпретатор Perl/PHP/Python - основні компоненти для побудови життєздатного багатоцільового вебсервера

В процесі роботи пропонується метод, який здійснює реляційно-об'єктне та об'єктно-реляційне перетворення (ОРП) таблиць бази даних у копіях класів базових моделей програмного коду. В даний час є розробки подібних систем, таких як Doctrine, Propel [2], але вони мають суттєвий недолік - великий обсяг програмного коду і надлишок реалізованих функцій. Наприклад, Доктрина включає більше 100 класів. Це спричиняє проблему з продуктивністю, оскільки інтерпретатор мови повинен завантажити значну

кількість файлів. Крім того, на об'єкти, отримані за допомогою ОРП, існує ряд обмежень:

1) Вони можуть знаходитися тільки в одній таблиці. Таким чином, проєктування об'єкта додатка зводиться до композиції об'єктів ОРП, що збільшує трудомісткість розробки.

2) Вони знаходиться в єдиній базі даних. Це випливає з попереднього пункту. Неможливо створити об'єкт, розподілений по декільком БД.

3) Мають фіксований набір властивостей. Зокрема, кількість колонок в таблиці має дорівнювати кількості властивостей об'єкта;

4) Ускладнена багатомовна підтримка, оскільки формується єдина таблиця, в якій частина стовпців "відповідає" за різні мови, а після вибірки необхідно фільтрувати дані..

Враховуючи переваги використання систем PDO та виходячи з недоліків існуючих систем, необхідно сформулювати цілі розробленої системи PDO:

1) Адекватне відображення предметів та їх зв’язків. Основною функцією PDO є відображення об’єктів предметної області веб-програми в реляційній структурі бази даних, а також відображення взаємозв’язків об’єктів. Це може призвести до низки проблем. Наприклад, об'єкт може зберігатися в багатьох таблицях безлічі баз даних на різних серверах. Об’єкти можуть бути різними за своєю будовою. Наприклад, нові елементи в процесі розробки можуть бути додані до об'єкта профілю користувача, а нові властивості можуть з'являтися в об'єкті.

2) Підтримка багатомовності. При розробці веб-додатків часто доводиться підтримувати кілька мов. Багатомовна підтримка повинна бути прозорою, тобто розробник повинен бути впевнений, що об’єкт зараз відображається поточною мовою для користувача. Але в той же час система повинна мати достатню гнучкість, щоб змінити мову відображення на таку, яку вимагає розробник.

3) Розроблені можливості пошуку. Система повинна підтримувати складні запити, наприклад, "знайти електронний лист першого менеджера компанії, який розмістив останній продукт".

В рамках даного дослідження було запропоновано систему об'єктно-реляційного відображення «Активна модель», яка вирішує вищевказані завдання. Для здійснення реляційно-об'єктних -перетворень необхідно на стороні об'єктної моделі реалізувати декларативний опис реляційної моделі. Для цього зараз використовуються мови опису об'єктів (*ODL - Object Definition Language*). Вони призначені для таких потреб:

1) Визначення схеми бази даних.

2) Забезпечення універсальності опису схеми бази даних. Можна змінити базу даних на іншу, тоді як опис на ODL не зміниться. ODL використовується як "загальний знаменник" при описі схеми бази даних.

3) Трансформація типів даних, тобто генерування типів зі схеми бази даних певною мовою програмування, з якої планується доступ до неї.

У свою чергу, мови сімейства OQL (Object Query Language) використовуються для реалізації RO-перетворень. Це мови запитів об’єктів, декларативні мови доступу до бази даних, подібні до мови SQL для баз даних. Стверджується, що вирази в OQL на 90% сумісні з синтаксисом оператора select з SQL'92 [2]. Відмінності між OQL та SQL полягають у тому, що вхідними даними запитів OQL є об'єкти, а не таблиці. Конструкція select-from-where використовується для написання запиту, як у SQL. Результатом запиту, як правило, є набір об’єктів - набір. Тоді цей набір можна перетворити на список, масив. Набір може оброблятися в циклі та отримувати окремі його елементи - об'єкти бази даних повністю або набір значень з бази даних у вигляді будь-якого типу мови програмування. Таким чином, взаємна модельна трансформація складається з інтерпритації OQL. В існуючих системах (наприклад, в Doctrine) модельне перетворення включає в себе такі операції як парсинг OQL-запиту, валідацію, кешування, перетворення OQL в SQL, виконання SQL, отримання «сирих» даних, гідрацію (перетворення сирих даних в об'єктний вид). З цим пов'язані значні витрати обчислювальних потужностей сервера, тому в даній роботі пропонується ряд змін, спрямованих на збільшення продуктивності операцій перетворення:

1) Зведення ODL до діалекту основної мови розробки. Для досліджуваної платформи LAMP − це PHP.

2) Визначення кінцевої множини перетворюваних типів і їх відображень.

3) Використання діалекту основної мови платформи для здійснення OQL-операція, а інтерпретатора платформи − для інтерпретації OQL команд.

4) Визначення спеціальних типів таблиць і їх ODL.

Діалект МП PHP, який використовується в якості ODL, буде називатися AMDL (ActiveModel Definition Language), а діалект, який використовується в якості OQL − AMQL (ActiveModel Query Language). Результат модельної трансформації на стороні об'єктної моделі буде називатися AM (ActiveModel).

Перед визначенням AMDL і AMQL-діалектів необхідно визначити структури, які піддаються РО перетворенню. Як відомо, інформація в РСУБД організована у вигляді множини таблиць (сукупності схем відносин і даних). У даній роботі пропонується класифікація схем на три типи: T_C − звичайна (classic), T_P-схема додаткових полів (x-properties), T_F − схема прапорів (x-flags) і T_{FT} − схема прапорів (x-flags-temporary) з темпоральною валідацією.

Реляційна модель задається наступним чином. Нехай $A_1, A_2, ..., A_n$ імена атрибутів. Кожному імені атрибута A_i відповідає допустима множина значень, які може приймати атрибут A_j. Це множина значень D_i називається доменом атрибуту A_i, $i = 1, n$. За визначенням, домени є непорожніми кінцевими або зліченними множинами. Поняттю домену D_i відповідає множина значень, що знаходяться в стовпці A_i розглянутої таблиці [4].

Схемою відношення $R\{A_{R1}, A_{R2}, ..., A_{Rn}\}$ називається кінцева множина імен атрибутів $\{A_{R1}, A_{R2}, ..., A_{Rn}\}$, причому атрибут A_i, приймає значення з множини D_{Ri} (i = l, 2, ..., n), де n - розмірність відношення.

Об'єктна модель задається наступним чином. Об'єктом O називається представлення сутності предметної області, яке використовується при моделюванні. $A\ \{A_{O1}, A_{O2}, ..., A_{On}\}$ є множиною атрибутів об'єктної моделі. Класом C називається загальна сутність, яка може бути визначена як сукупність елементів (реалізацій класу). Клас є родовою ознакою об'єктів.

Після визначення операцій перетворення типу даних можна визначити набір операцій перетворення. Об'єктно-реляційна система відображення перетворює набір кортежів (записів, рядків) даних, що складають набір значень атрибутів, у форму, з якої можуть взаємодіяти функції мови програмування PHP. У запропонованому методі (і системі ORP "Активна модель") набір кортежів відображається на екземплярі класу. Об'єктом класу "Активна модель" є сукупність відносин, отриманих в результаті дії природного зв'язку, застосованих до безлічі кортежів (обов'язково по одному для кожного зв'язку) та набору функцій (поведінки) над сукупність відносин, успадкованих від класу програмного забезпечення. Натуральним зв’язком є операція SQL NATURAL JOIN, яка повертає відношення, яке містить усі можливі кортежі (K), які є комбінаціями двох (або більше) кортежів, що належать до двох (або більше) заданих відносин, за умови, що в комбінованих кортежах є однакові значення в одному (або декількох) загальних для вихідних атрибутів (і ці загальні значення з’являються в результуючому кортежі рівно один раз).

Основна відмінність методики «Активна модель» від інших методик в тому, що n (кількість відношень, які використовуються для відображення) може бути більше 1, тоді, як для існуючих методик об'єктно-реляційного відображення на платформі LAMP (маються на увазі методики Doctrine, Propel, Yii Active Record) $n = 1$. З цього випливає, що реалізація моделі даних, подібної до «Активної моделі», вимагає створення в них n моделей. Відповідно, вартість пам'яті, необхідної для зберігання моделей даних, збільшиться за рахунок збільшення кількості зразків класів.

Перелік посилань

1. Майк Кон. Scrum: Гибкая разработка ПО. / Майк Кон. — Изд-во: Диалектика-Вильямс, 2016. — С. 576.

2. Doctrine OPvM [Електронний ресурс]. – Режим доступу: URL: http://doctrine-project.org (дата звернення: 04.08.2017).

3. Роберт Мартин Гибкая разработка программ на Java и C++. Принципы, паттерны и методики. /Роберт С. Мартин, Джеймс Ньюкирк, Роберт Косс - Изд-во: Диалектика-Вильямс, 2016. - 704с.

4. Джулій В.М. Методи та алгоритми розробки web-додатків / В.М. Джулій, Ю.О. Гунченко, Д.В. Чешун // Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка. – К.: ВІКНУ, 2017. – Вип. № 56. – С.107-115

Дослідження проблем ідентифікації об'єктів в базах даних

Мозолюк В.О., Джулій В.М.
Хмельницький національний університет

На даний момент СУБД широко використовуються в організації сучасних інструментальних, промислових, аналітичних та інформаційних систем. Однак такий бурхливий розвиток інформаційних технологій баз даних поставило також ряд нових проблем і визначило напрямки подальших досліджень у цій області. Не припиняюча робота дослідників та аналітиків відноситься до питань оптимізації виконання запитів і структур зберігання даних, новітніх способів виконання реляційних операцій, організації пошуку, і багато інших моментів, що визначають результативність роботи СУБД. Програмне забезпечення на даний момент розвивається в умовах швидкого зростання обчислювальних потужностей, апаратних можливостей, швидкості доступу до пам'яті, обсягу пам'яті, пропускної здатності та надійності каналів передачі даних. Все більшого значення набувають засоби, що забезпечують взаємодію в розподіленої системі функціонування інформаційних систем.

Розглянемо більш докладно основні напрямки розвитку сучасних баз даних і СУБД:

1. Стандартизація мови SQL. У сучасних СУБД на даний момент основною мовою написання запитів і доступу до баз даних є мова SQL (Structured Query Language). Міжнародний стандарт даної мови розроблений в 1989 році, і більшість виробників СУБД призвели свої системи у відповідність даному стандарту. Потрібна постійна актуалізація мови SQL до мінливих вимог сучасних програмних продуктів та апаратних засобів.

2. Використання мультипроцесорних організацій. Промислові комерційні СУБД реалізуються на основі архітектури "клієнт-сервер". При даній організації всі операції над базами даних виконуються на сервері, що володіє достатньою продуктивністю і набором обчислювальних ресурсів. Після появи мультипроцесорних симетричних апаратних архітектур в