УДК 004.021:004.92

Цюцюра Світлана Володимирівна
Доктор технічних наук, професор, професор кафедри інформаційних технологій,*ORCID: 0000-0002-4270-7405*
Київський національний університет будівництва і архітектури, Київ
Бородавка Євгеній Володимирович
Кандидат технічних наук, доцент, доцент кафедри інформаційних технологій проектування та прикладної математики, *ORCID: 0000-0002-7476-9387*
Київський національний університет будівництва і архітектури, Київ

МЕТОДИ ПРОЕКЦІЇ ОБ'ЄКТНИХ МОДЕЛЕЙ НА СТРУКТУРИ ДАНИХ

Анотація. *Проблема адекватного збереження об'єктно-орієнтованих моделей в реляційні бази даних є актуальною задачею на сьогодні. Є декілька стандартних підходів до вирішення поставленої проблеми. Їх дослідження, аналіз переваг та недоліків, а також сфери можливого застосування наведено в даній статті. Як приклад використовувався фрагмент діаграми класів, що була розроблена для використання в системі автоматизованого проектування на основі життєвого циклу будівельного об'єкта. За результатами дослідження зроблено висновок, що найефективнішою стратегією є комбінація розглянутих методів залежно від поставленої мети.*

Ключові слова: об'єктна модель; об'єктне реляційне перетворення; реляційна база даних; рефакторинг; проекція класів

Вступ

У процесі розробки системи автоматизованого забезпечення життєвого циклу будівельного об'єкта виникла проблема організації моделі структури даних для збереження інформації про будівельний об'єкт на всіх етапах його життєвого циклу. Слідуючи сучасним тенденціям в розробці програмних комплексів, система розроблялася об'єктно-орієнтованою, тому всі основні сутності системи реалізовані у вигляді об'єктів класів. Це дозволяє системі бути стійкою до внесення помилок і простою з точки зору рефакторингу.

Оскільки система реалізована об'єктно-орієнтованою, то для збереження сутностей, якими вона оперує, необхідно використати механізми проекції об'єктних моделей на реляційну базу даних [1].

Аналіз останніх досліджень і публікацій

Основною роботою з даного напрямку є [1], в якій описані всі можливі варіанти проектування об'єктних моделей на реляційні бази даних, а також їх переваги і недоліки.

У роботі [2] розглядається додатковий аспект проектування об'єктних моделей в реляційні бази даних, що пов'язаний з оптимізацією швидкості роботи БД, а також її розміру.

Мета статті

Оскільки розроблювана система має розгалужену ієрархічну структуру класів, то є сенс автоматизувати відображення класів на структуру даних та максимально її оптимізувати. Процес відображення системи класів на реляційну базу даних називається Object Relational Mapping (O/RM), а посередник між об'єктно-орієнтованим середовищем та реляційною базою даних відповідно – Object Relational Mapper (ORM).

Пряме повторення полів класу в таблиці бази даних зазвичай недостатнє для створення адекватного реляційного подання об'єктної моделі. Окрім звичайної інформації, що наявна в класах, необхідно зберігати також інформацію про зв'язки між класами, деяку інформацію про поведінку класу, додаткову інформацію для управління об'єктом (так звана «shadow information»), а також метаінформацію у випадку опису складних класів-списків чи інших складних класів.

Виклад основного матеріалу

Методи проекції об'єктів

Розглянемо для початку просту діаграму класів та її проекцію на реляційну базу даних (рис. 1).

У простій проекції кожен атрибут класу проектується в одне поле таблиці бази даних, що має такий же тип даних. Звичайно так просто ніколи не буває. Зазвичай типи атрибутів класу і полів таблиці можуть відрізнятися, або один атрибут проектуватися в більше, ніж одне поле. Крім того,

між діаграмою класів та її проекцією на базу даних (рис. 1) є деякі відмінності пов'язані з введенням в таблиці БД додаткових полів, що використовуються в якості внутрішніх та зовнішніх ключів.

У результаті проектування об'єктної моделі в реляційну виникає необхідність її рефакторингу відповідно до отриманої реляційної моделі. Оскільки в таблицях БД з'явилася додаткова інформація, що потрібна для підтримки роботи ORM, то UML діаграма класів повинна бути приведена у відповідність до реляційної структури даних. Таким чином в об'єктній моделі з'являться нові атрибути і методи в класах, щоб забезпечити їх максимально просте та ефективне збереження і відновлення з бази даних.

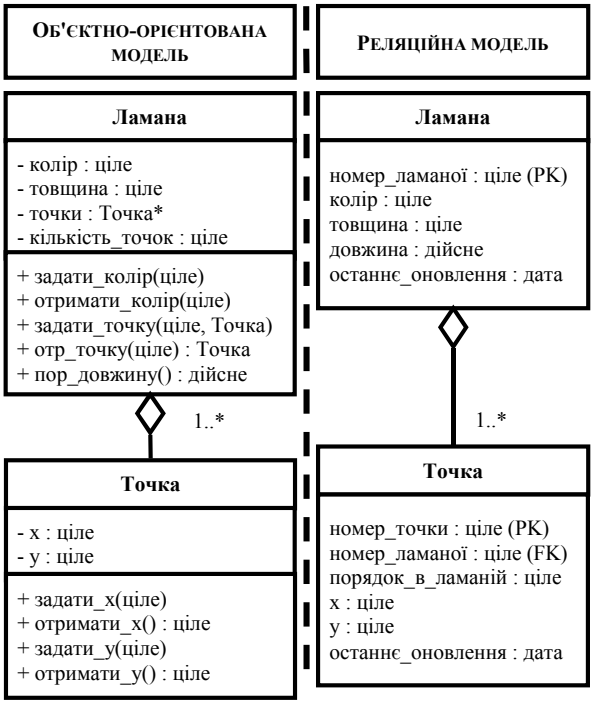

Рисунок 1 – Простий приклад проекції

Тепер розглянемо більш складні ієрархічні об'єктні моделі. Реляційні бази даних не підтримують ієрархії автоматично, тому необхідно мати спеціальні підходи для реалізації запису/зчитування ієрархії класів об'єктної моделі з її реляційного аналогу. Для реалізації проектування ієрархічних об'єктних моделей на реляційні БД є чотири основні підходи:

– проекція ієрархії класів в одну таблицю;

– проекція кожного конкретного класу в окрему таблицю;

– проекція кожного класу в окрему таблицю;

– проекція класів в узагальнену структуру таблиць.

Кожен з наведених варіантів має свої переваги та недоліки і може бути застосований в окремих випадках залежно від обставин. Розглянемо кожен з чотирьох варіантів більш детально на прикладі двох версій простої ієрархії класів (рис. 2).

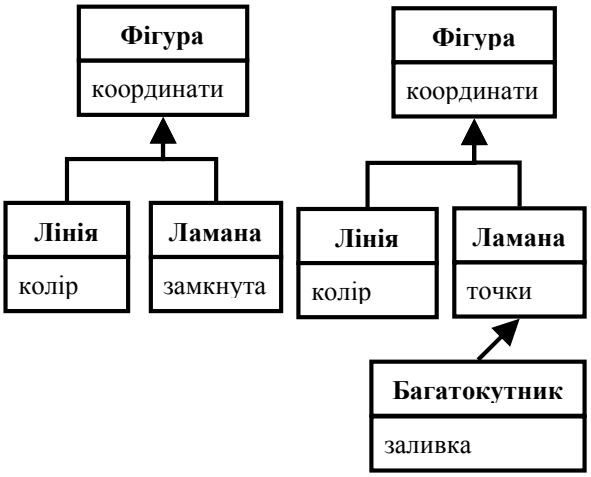

Рисунок 2 – Дві версії простої ієрархії класів

Перша версія ієрархії класів описує абстрактну фігуру, що має деякий загальний набір атрибутів. Як приклад показане поле координат оточуючого прямокутника. Фігура має двох "нащадків" – лінія, для якої в якості поля вказаний колір та ломану, яка має поле-список, що визначає координати точок ломаної.

Друга версія практично аналогічна першій, але з класу «Ломана» відокремлений клас нащадок «Багатокутник», який в якості прикладу має поле «заливка», що визначає колір заповнення багатокутника.

Тепер розглянемо як ці дві версії ієрархії проектуються на реляційну базу даних за допомогою різних підходів.

Проекція всієї ієрархії в одну таблицю

При застосуванні цього підходу вся ієрархія класів проектується в одну таблицю і всі атрибути класів стають полями цієї таблиці. Назву таблиці слід обрати за назвою базового класу ієрархії, це дуже проста і зрозуміла манера проектування. Результат використання стратегії проектування всієї ієрархії класів в одну таблицю подано на рис. 3.

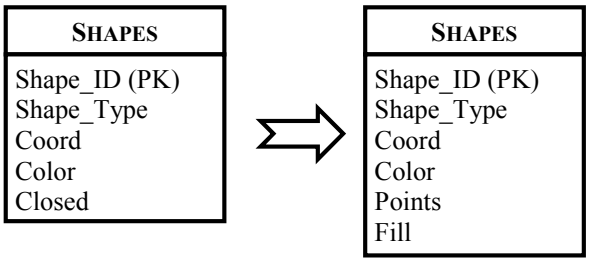

Рисунок 3 – Проекція ієрархії в одну таблицю

Як видно на рис. 3, в таблиці з'явилися два додаткових поля: перше – це унікальний ключ таблиці, а друге – тип фігури, що може приймати значення – лінія, ламана або можливо навіть обидві (уявімо для прикладу, що це можливо).

Але, якщо ми додамо поняття багатокутник в нашу ієрархію, то тип може приймати тепер значення «багатокутник», а в загальному випадку у нас може виникнути необхідність, щоб наша фігура було одночасно і багатокутником і ламаною. У такому випадку у нас будуть множитися ознаки для вмісту поля «тип» і в кінці кінців ми будемо змушені зробити рефакторінг таблиці БД, замінивши одне поле «тип» на три булевих поля «IsLine», «IsPolyline» та «IsPolygon» (рис. 4).

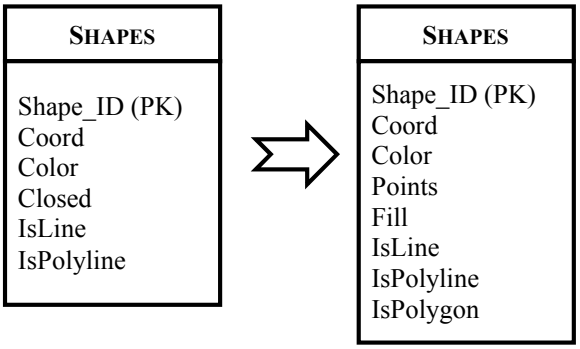

Рисунок 4 – Таблиця БД після ре факторингу

За такого підходу є висока імовірність зростання кількості полів таблиці – тобто розростання таблиці в ширину. Особливо у випадку великої ієрархії з великою кількістю даних. Велика кількість полів не дуже добре впливає на швидкодію бази даних. Також у випадку додавання нових атрибутів у об'єктно-орієнтовану модель зміна реляційної моделі потребуватиме додавання нових полів, що зробить неможливим використання попередніх баз даних без їх фізичного рефакторингу.

Проекція кожного конкретного класу в окрему таблицю

За такого підходу до проектування об'єктної моделі на реляційну базу даних кожен конкретний клас (не абстрактний) проектується в окрему таблицю, куди записуються його власні атрибути, а також успадковані атрибути від базового класу. Приклад реляційної БД для розглянутого підходу подано на рис. 5.

Як ми бачимо на рис. 5, таблиці відповідають класам, а їх поля відповідають власним та успадкованим атрибутам класів. Окрім того, кожна таблиця має первинний ключ. Якщо при такому підході виникне необхідність у появі нового успадкованого класу (в даному прикладі "Багатокутник"), то все, що потрібно зробити, це додати ще одну таблицю для цього класу.

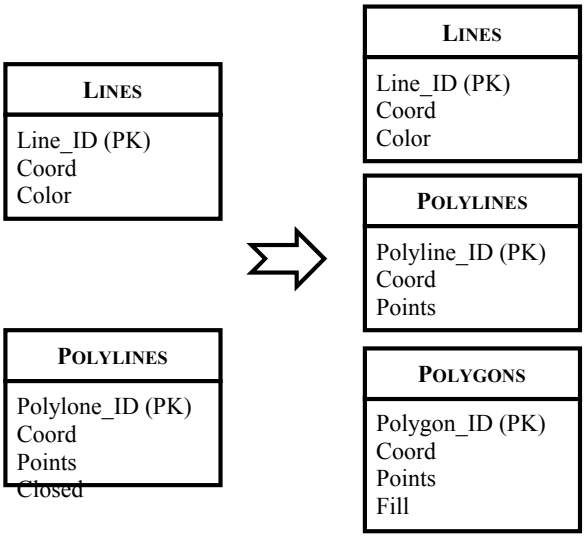

Рисунок 5 – Проекція конкретних класів в окремі таблиці

Проекція кожного класу в окрему таблицю

За такого підходу кожен клас (включаючи абстрактні) проектується в окрему таблицю зі всіма своїми атрибутами. У результаті дані кожного об'єкта будуть знаходитися як мінімум в двох таблицях (рис. 6).

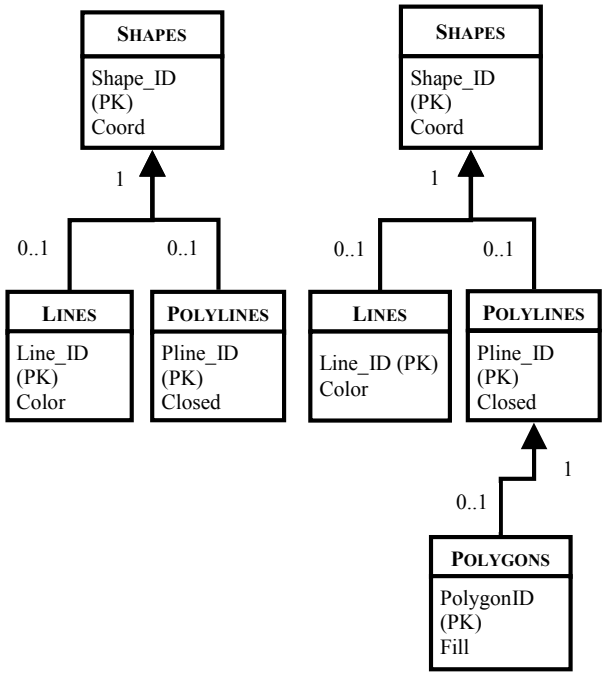

Рисунок 6 – Проекція кожного класу в таблицю

Оскільки таблиці практично повторюють класи, то відповідно для того, щоб завантажити із БД екземпляр класу нащадку необхідно також завантажити дані з таблиці батьківського класу.

Таблиця - ***Порівняння стратегій проектування***

Стратегія	Переваги	Недоліки	Сфера застосування
Одна таблиця на ієрархію	Простий підхід Легко додавати нові класи, для цього необхідно лише додати нові поля в таблицю. Підтримка поліморфізму простою зміною типу рядка. Швидкий доступ до даних, оскільки вся інформація в одній таблиці	Зміцнення зв'язків всередині ієрархії класів, оскільки всі вони знаходять в тісних зв'язках в таблиці. При цьому зміна в одному класі може вплинути на таблицю, що в свою чергу може вплинути на інші класи в ієрархії. Потенційне марнування простору в БД. Ускладнюється визначення типу, оскільки накладання типів стає значним. Таблиця може швидко розростися для великих ієрархій	Це добра стратегія для простих і невеликих ієрархій класів, де відсутнє або невелике накладання типів даних
Одна таблиця для кожного конкретного класу	Хороша продуктивність доступу до даних кожного класу	Коли модифікується клас, то необхідно модифікувати таблицю цього класу і таблиці всіх дочірніх класів. Якщо один з об'єктів змінить роль, то доведеться копіювати дані між таблицями. Важко підтримувати об'єкти, що мають декілька ролей	Коли зміна типів та їх перекриття трапляються дуже рідко
Одна таблиця для класу	Легка для розуміння стратегія, оскільки це є проектуванням один-до-одного. Просто підтримується полімофізм, оскільки різні об'єкти знаходяться в різних таблицях. Дуже легко модифікувати батьківські класи і додавати нові дочірні класи, оскільки потрібно модифікувати/додавати лише одну таблицю. Розмір даних збільшується пропорційно до збільшення кількості об'єктів	Багато таблиць в БД, одна на кожен клас плюс таблиці для підтримки БД. Потенційно матиме більш тривалий час доступу до даних, оскільки необхідно здійснювати читання об'єкта з декількох таблиць	Коли має місце значне перекриття типів даних або зміна типів звичайна справа
Узагальнена схема	Працює дуже добре, коли БД інкапсульована за допомогою спеціалізованих фреймворків. Може бути легко розширена за допомогою метаданих для підтримки проектування зв'язків. Дуже гнучка і дає прості можливості змінювати типи атрибутів, що зберігаються	Дуже досконала техніка, що може бути важкою для застосування на перших етапах. Працює на невеликих об'ємах даних, оскільки для читання одного об'єкта необхідно прочитати багато рядків БД. Необхідно створювати адміністративний застосунок для керування метаданими	В застосунках, де немає великої кількості даних або там, де швидкість доступу до даних не критична і може бути здійснене попереднє завантаження даних в кеш

Проекція класів в узагальнену структуру таблиць

Цей підхід також називають проекція з використанням метаданих. Даний підхід не залежить від ієрархії конкретних класів і є універсальним для будь-якого випадку. На рис. 7 показана логічна структура для опису будь-якої ієрархії класів за допомогою опису значень атрибутів.

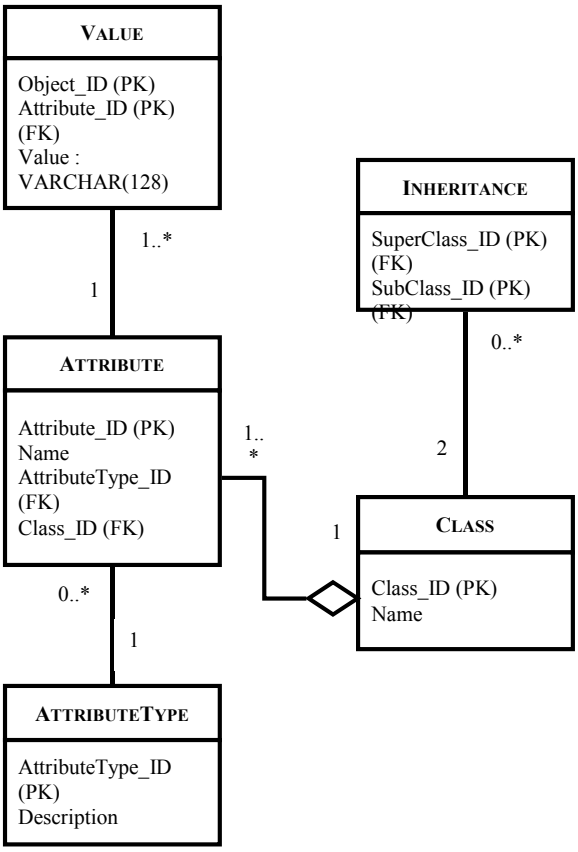

Рисунок 7 – Проекція класів в узагальнену структуру таблиць

Подана схема показує лише опис ієрархічних зв'язків і може бути доповнена асоціативними зв'язками. Значення кожного атрибуту зберігається в таблиці «Value». В такому випадку клас із десяти атрибутів буде описаний десятьма рядками в цій таблиці. Таблиця «Attribute Type» містить опис всіх використовуваних типів атрибутів: ціле, дійсне, гроші, дата і т. п. Самі атрибути при цьому описані у відповідній таблиці «Attribute». Таблиця «Class» містить перелік всіх класів, що використовуються. Таблиця «Inheritance» описує зв'язки успадковування між класами – фактично вибудовує ієрархію класів за допомогою понять «батьківський клас» та «клас-нащадок». Єдине обмеження за такого підходу – це відсутність множинного успадковування.

Тепер розглянемо як модель із прикладу на рис. 1 проектується в узагальнену структуру таблиць. Для збереження класу «Точка в просторі» (*T3DPoint*) потрібно мати три записи в таблиці «Value» для збереження координат X, Y, та Z відповідно. Стільки ж записів для цього класу буде і в таблиці «Attribute», а в таблиці «Class» буде міститися один запис, що описує цей екземпляр класу.

Далі розглянемо ієрархічну структуру подану на рис. 2. В цьому прикладі нас цікавить заповнення таблиці «Inheritance». Ця таблиця є ключовою для проектування спадковості класів. Кожен запис в ній описує пару «батько-нащадок», використовуючи для цього унікальні ідентифікатори класів. Інші таблиці заповнюються аналогічно до попереднього прикладу.

Випадок множинного успадкування детально розглядати не будемо, оскільки для нього застосовуються точно такі ж моделі, що і для звичайних випадків.

Порівняння різних стратегій проектування

Жодна з описаних стратегій не є ідеальною для будь-якого випадку в проектуванні. Іноді варто почати проектування використовуючи одну стратегію, а в подальшій роботі виконати рефакторінг до іншої моделі. Переваги та недоліки, а також сфери найкращого застосування кожної із моделей наведені в таблиці.

Висновки

Оскільки жодна з моделей не є універсальною, то при виборі стратегії необхідно зважати на предметну область її застосування. У загальному випадку в одному проекті можна використовувати комбінацію цих стратегій для різних частин ієрархії класів.

Список літератури

1. Ambler S.W. Mapping Objects to Relational Databases: O/R Mapping In Detail [Електронний ресурс]. – http://www.agiledata.org/essays/mappingObjects.html.

2. Borodavka Y.V. Binary Data Packing Method for Database Optimization [Текст] / Y.V. Borodavka, M.I. Tsiutsiura // International Journal of Science and Research. – 2014. – Volume 3, Issue 11. – C. 903 – 905.

3. Connolly T.M. Database Systems: A Practical Approach to Design, Implementation and Management / T.M. Connolly, C.E. Begg. – Addison Wesley; 5 edition, 2009. – 1400 с. – ISBN: 978-0321523068.

4. Amber S.W. The Elements of UML(TM) 2.0 Style / S.W. Amber. – New-York: Cambridge University Press, 2005. – 188 с. – ISBN: 978-0521616782.

5. Borodavka Y.V. Product lifecycle management in construction / Y.V. Borodavka, S.L. Pechenov // Східноєвропейський журнал передових технологій. – 2010. – №6/3(48). – С. 31-34.

6. .Borodavka Y.V. Building Model Conception [Текст] / Y.V. Borodavka, M.I. Tsiutsiura // Proceedings of the International Scientific and Practical Conference WORLD Science "Science and Education – Our Future". – 2014. – Issue №3, November 24-26, 20014, Abu Dhabi, UAE. – С. 53 –55.

7. Стадник М. ORM и с чем его едят [Електронний ресурс]. – http://mikhailstadnik.com/few-words-about-orm.

8. Гамма Э. Приемы объектно-ориентированного проектирования / Э. Гамма, Р. Хелм, Р. Джонсон – Питер, 2001 – 368 с.

9. Грейди Буч. UML. Руководство пользователя / Грейди Буч, Джеймс Рамбо, Айвар Джекобсон – М: ДМК, 2000. – 432 с.

10. Конолли Т. Базы данных: проектирование, реализация и сопровождение. – М.: Диалектика, 2000. – 1120 с.

11. Ли К. Основы САПР (CAD/CAM/CAE). – СПб.: Питер, 2004. – 560 с.

Стаття надійшла до редколегії 30.12.2014

Рецензент: д-р техн. наук, проф. С.Д. Бушуєв, Київський національний університет будівництва і архітектури, Київ.

Цюцюра Светлана Владимировна
Доктор технических наук, профессор, профессор кафедры информационных технологий, *ORCID: 0000-0002-4270-7405*
Киевский национальный университет строительства и архитектуры, Киев
Бородавка Евгений Владимирович
Кандидат технических наук, доцент, доцент кафедры информационных технологий проектирования и прикладной математики, *ORCID: 0000-0002-7476-9387*
Киевский национальный университет строительства и архитектуры, Киев

МЕТОДЫ ПРОЕКЦИЙ ОБЪЕКТНЫХ МОДЕЛЕЙ НА СТРУКТУРЫ ДАННЫХ

Аннотация. *Проблема адекватного сохранения объектно-ориентированных моделей в реляционные базы данных является актуальной задачей на сегодняшний день. Существует несколько стандартных подходов к решению поставленной проблемы. Их исследование, анализ преимуществ и недостатков, а также сферы возможного применения проведены в данной статье. В качестве примера использовался фрагмент диаграммы классов, которая была разработана для использования в системе автоматизированного проектирования на основе жизненного цикла строительного объекта. По результатам исследования сделан вывод, что наиболее эффективной стратегией является комбинация рассмотренных методов в зависимости от поставленной цели.*

Ключевые слова: объектная модель; объектное реляционное преобразование; реляционная база данных; рефакторинг; проекция классов

Tsiutsiura Svitlana
Doctor of Technical Science, Professor, professor of Information technologies department, *ORCID: 0000-0002-4270-7405*
Kyiv National University of Construction and Architecture, Kyiv
Borodavka Yevgeniy
Doctor of Philosophy, Docent, associate professor of Information technologies of Design and applied mathematics department, *ORCID: 0000-0002-7476-9387*
Kyiv National University of Construction and Architecture, Kyiv

METHODS OF OBJECT MODELS MAPING INTO DATA STRUCTURE

Abstract. *The problem of adequate object-oriented model storing in relation databases is actual for nowadays. There are several approaches to resolve this problem. In this paper we try to study, analyze and compare each of knowing method for the sake of choosing best of its. Fragment of class diagram, developed for computer aided design system based on building lifecycle, was used as an example. The methods, which were considered: map hierarchy to a single table, map each concrete class to its own table, map each class to its own table and map classes to a generic table structure. For every method, the advantages and shortcomings were given and composed to comparing table. As a result of research we make conclusion about most effective mapping method. According to the advantages and shortcomings no one methods cannot be treated as most effective. For enterprise applications development the combination methods using is most suitable according to a resolving task.*

Keywords: object model; object relational mapping; relation database; refactoring; classes mapping

References

1. Ambler, S.W. Mapping Objects to Relational Databases: O/R Mapping In Detail [electronic source]. – http://www.agiledata.org/essays/mappingObjects.html.

2. Borodavka, Y.V. (2014). Binary Data Packing Method for Database. International Journal of Science and Research. 3(11), 903 – 905.

3. Connolly, T.M. & Begg, C.E. (2009). Database Systems: A Practical Approach to Design, Implementation and Management. Addison Wesley; 5 edition, 1400. ISBN: 978-0321523068.

4. Amber, S.W. (2005). The Elements of UML(TM) 2.0 Style. New-York: Cambridge University Press, 188. ISBN: 978-0521616782.

5. Borodavka, Y.V. (2010). Product lifecycle management in construction / Y.V. Borodavka, S.L. Pechenov // East-european journal od advanced technologies, 6/3(48), 31-34.

6. Borodavka, Y.V. & Tsiutsiura, M.I. (2014). Building Model Conception. International Scientific and Practical Conference WORLD Science "Science and Education– Our Future", №3, November 24-26, 20014, Abu Dhabi, UAE. pp. 53– 55.

7. Stadnik, M. ORM and with what it eats [electronic source]. – http://mikhailstadnik.com/few-words-about-orm.

8. Gamma, E., Helm, R. & Johnson, R. (2001). Object-oriented design examples. Piter, 368.

9. Booch, G., Rumbaugh, J. & Jacobson, I. (2001). UML. User guide. Moscow, Russia: DMK 2000, 432.

10. Connolly, T. (2000). Databases: design, implementation and management. Moscow, Russia: Dialektika, 1120.

11. Lee, K. (2004). Principles of CAD/CAM/CAE Systems. SPt., Russia: Piter, 560.

Посилання на публікацію

APA *Tsiutsiura, S., & Borodavka, Ye. (2015). Methods of Object Models Mapping into Data Structure. Management of Development of Complex Systems, Issue 21, P. 92 – 98 [in Ukrainian].*

ГОСТ *Цюцюра С.В. Методи проекцій об'єктних моделей на структури даних [Текст] / С.В. Цюцюра, Є.В. Бородавка // Управління розвитком складних систем. – 2014. - № 20. – С. 92 - 98.*