Інфокомунікаційні системи

УДК 004.4’22

Н.І. Калита, В.В. Стрельченко

Харківський національний університет радіоелектроніки, Харків

АВТОМАТИЗАЦІЯ ПРОЕКТУВАННЯ СТРУКТУР БАЗИ ДАНИХ MONGODB

Запропоновано технологію, яка дозволяє на основі XMI-документа або іншого способу подання інформації про предметну область у структурованому вигляді синтезувати структури бази даних MongoDB. Програмна реалізація технології призначена для використання у складі системи автоматизованого проектування MongoDB системним архітектором на етапі проектування інформаційного забезпечення комп’ютеризованих систем управління. Розроблена технологія дозволяє зменшити часові втрати на проектування шляхом формалізації підходу до нього.

Ключові слова: *документо-орієнтована база даних, проектування БД, автоматизований синтез структур БД, САПР, MongoDB, XML, XMI, UML.*

Вступ

Постановка проблеми. Невід'ємною складовою частиною будь-якої інформаційної системи (ІС) є підсистема інформаційного забезпечення (ІЗ), де інформація структурована, організована і зберігається у вигляді бази даних (БД). Серед ієрархічних, мережевих, реляційних та інших найбільш поширеною на теперішній час є реляційна модель. Однак все більший інтерес викликає напрямок розвитку БД – NoSQL (Not Only SQL). NoSQL є, в якійсь мірі, неофіційною узагальнюючою назвою для БД, в яких маніпуляції над даними здійснюються способами, відмінними від застосування SQL (Structured Querying Language).

Отже, назва «NoSQL» ніяк не регламентує модель даних, використовувану для організації БД. При цьому NoSQL БД мають певний набір загальних особливостей:

– відмова від SQL (Structured Query Language): кожна БД керується за допомогою своєї мови спілкування з БД;

– необов'язкова структурованість: структура БД може бути регламентована по-різному залежно від обраної NoSQL БД (починаючи з можливості зберігати дані в будь-якому вигляді до наявності цілком чітких правил попередньої структуризації даних);

– можливість використання такої ж моделі представлення даних, як і в системі, що розробляється;

– відсутність транзакцій: у більшості NoSQL БД атомарність операцій присутня тільки на рівні одного примірника даних (документа, запису, об'єкта), тобто за несуперечливість даних і транзакції відповідає програмний продукт, який розробляється.

MongoDB є однією з найбільш популярних NoSQL БД на теперішній час, накопичено певний досвід вибору структур даних і кращих практик застосування MongoDB в різних предметних областях, але не розроблено універсального підходу і відсутня інформаційна технологія проектування структури MongoDB та її реалізації в рамках системи автоматизованого проектування (САПР) [1].

Мета дослідження – підвищити ефективність та якість розробки структури бази даних MongoDB для заданої предметної області за рахунок автоматизації процесів проектування та оцінювання можливих її варіантів як складових інформаційної технології. Для досягнення цієї мети вирішені задачі аналізу і вибору структури вхідних даних, розроблена технологія автоматизованого синтезу структур БД MongoDB.

Основний матеріал

Задача вибору структури вхідних даних. MongoDB належить до документо-орієнтованих БД, в яких:

1) дані зберігаються в документах;

2) документи легко проектуються на структури і типи мов програмування;

3) вкладені документи зменшують необхідність в об'єднаннях при вибірці даних (з'єднання), а масиви дозволяють організувати звичні зв'язки між документами;

4) динамічна структура даних спрощує реалізацію поліморфізму.

Висока продуктивність MongoDB забезпечується за рахунок вкладеності даних (прискорюється читання), опціонального швидкісного запису (без підтвердження запису) та можливості вмісту в індексах полів і масивів вкладених документів.

Модель даних MongoDB передбачає можливість зберігання множини БД, які можуть містити набір колекцій (collections – аналог сутностей в реляційних БД).

Колекція може містити набір документів.

Документи – це множина елементів «ключ-значення». Документи підтримують динамічну структуру, що дозволяє різним документам однієї колекції містити різні набори полів або структуру, однакові поля можуть містити різні типи даних. Специфікація, яка використовується для візуального представлення документів користувача і для роботи з ними, називається JSON (Javascript Object Notation) [2]. Для серіалізації, передачі даних по мережі і зберігання використовується специфікація BSON (Binary JSON) [3]. BSON надає велику кількість типів даних для зберігання, оскільки саме в цьому форматі дані зберігаються в БД, а JSON – просто спосіб представлення цих даних користувачеві для комфортної роботи.

Проектування структури бази даних в MongoDB не є тривіальним завданням. Для забезпечення продуктивності при великих обсягах даних в процесі проектування необхідно враховувати:

а) зв'язки між даними, що визначає необхідність денормалізації структури документа. Існують наступні варіанти запитів:

– пов'язані дані зазвичай викликаються разом, тобто якщо зчитуються дані, то зчитується і пов'язана з ними інформація, – тоді варто розглянути варіант вкладеної структури документа (embedding), тому що робота з одним документом здійснюється швидше і не вимагає повторного звернення до сервера за даними;

– пов'язані дані разом використовуються не дуже часто, і необхідно зберегти нормалізовану структуру бази даних, щоб уникнути надмірності. У цьому випадку варто розглянути варіант зовнішніх зв'язків, наприклад, за допомогою масивів, якщо зв'язок «багато-до-багатьох»;

б) якщо запити, які часто використовуються, постійно викликають дані з різних колекцій, то необхідно розглянути варіант злиття цих колекцій;

в) індекси, які в MongoDB за властивостями і функціями аналогічні індексам в реляційних БД. Створення індексів для полів або поля збільшує швидкість доступу до цього поля при читанні і уповільнює роботу при частому записі. Іншими словами, якщо у колекції відношення записів до читання мале, то використання індексів рекомендовано, якщо велике – ні;

г) шардінг (Sharding) – горизонтальне розширення, яке дає значне збільшення швидкості роботи з великими масивами даних у БД. При використанні шардінга необхідно вибрати шард-ключ (shard-key), вибір якого матиме великий вплив на ефективність використання функції шардінга в цілому;

д) атомарність операцій. Якщо необхідно змінювати пов'язані дані за одну неперервну операцію, то слід використовувати вкладені документи, оскільки в MongoDB атомарність операції (транзакції) підтримується тільки на рівні одного документу.

Оскільки на даний час на ринку програмних продуктів не представлені засоби автоматизації процесу проектування структур БД в MongoDB, необхідно розробляти інструментарій синтезу варіантів структур БД та їх тестування.

Вхідними даними під час проектування БД є структурована інформація про предметну область. Основним способом моделювання логічної структури реляційної БД є діаграма «сутність-зв'язок» (ERD). При створенні ERD для БД, як правило, намагаються дотримуватися правила приведення структури даних до третьої нормальної форми (3НФ), яка забезпечує відсутність надмірності і суперечливості.

Отже, спосіб представлення предметної області у вигляді ERD передбачає чіткі правила структурування інформації і тому може бути використаний в якості вхідних даних для вирішення задачі синтезу схем БД.

Цей підхід найкраще підходить у випадку, коли вже є готова структура реляційної БД або складена діаграма «сутність-зв'язок».

Однак, коли проект знаходиться на стадіях передпроектного дослідження або є готовий програмний продукт, такий підхід може виявитися неприйнятним внаслідок того, що:

а) більшість програмних проектів розробляються з урахуванням парадигм об'єктно-орієнтованого програмування (ООП), а для представлення успадкування та поліморфізму за допомогою діаграми «сутність-зв'язок» необхідно відступати від правил 3НФ. Отже, інтерпретація такої інформації може бути неоднозначною та нетривіальною, а також може вимагати додаткового часу на розробку діаграми «сутність-зв'язок»;

б) за принципами ООП програми складаються з об'єктів і зв'язків між ними. Розробнику було б легше, за наявності певної структури програмного продукту, просто надати таку структуру для інтерпретації програмою і не витрачати додатковий час на розробку ще однієї діаграми тоді, коли вся необхідна інформація вже структурована;

в) дерева є ще однією структурою, яку не так просто або не так просто наглядно відобразити за допомогою діаграми «сутність-зв'язок».

Інший спосіб представлення структурованої інформації про предметну область, який задовольняє всі вимоги, які не можуть бути задоволені ERD-діаграмою – Unified Modeling Language (UML) [4]. Уніфікована мова моделювання UML відображає інформацію про предметну область за допомогою діаграми класів (Class Diagram) і надає можливість наочного представлення спадкування та поліморфізму. Також суттєвою властивістю діаграми класів є наявність додаткових типів зв'язків, які спрощують інтерпретацію вхідних даних. Правилами, яких дотримуються при розробці діаграми класів, виступають патерни (pattern) проектування і парадигми програмування, якими керуються при розробці об'єктів і зв'язків предметної області.

У деяких аспектах вони схожі з правилами приведення діаграми «сутність-зв'язок» до третьої нормальної форми: наприклад, об'єкт повинен містити тільки інформацію, яка описує цей об'єкт або ним використовується.

Проведений аналіз дозволяє виділити вихідні передумови для автоматизованого проектування структур MongoDB і способи структурованого подання інформації про предметну область:

1. Якщо у розробника є структура БД у вигляді ERD, з якої він хоче отримати документу структуру, тоді вхідними даними є ERD.

2. Якщо у розробника є готові класи системи або класи та БД, або для системи ще не розроблено нічого з перерахованого, тоді вхідними даними є діаграми класів.

Постановка задачі проектування. Автоматизоване проектування структури MongoDB складається з етапів:

1. Автоматизований синтез схем БД для MongoDB.

2. Проведення автоматизованого оцінювання отриманих структур або структури.

3. Багатокритеріальний вибір найкращої структури або структур на множині згенерованих варіантів.

У даній роботі розглядається перший з наведених етапів.

Автоматизована система синтезу схем БД для MongoDB призначена для:

– автоматизованої генерації структур БД на основі розширюваного механізму правил;

– моніторингу додаткової інформації про синтезовані структури: аномалії модифікації, звернення до надлишкових даних та атомарність доступу до пар об'єктів.

Вхідною інформацією є:

– структурована інформація про предметну область у вигляді діаграми класів UML;

– додаткові обмеження у вигляді XML-документу (eXtensible Markup Language):

1) вкладеність класів;

2) атомарність доступу до пар класів;

3) уточнення зв'язків «один-до-багатьох» зв'язками «один-до-декількох».

Вихідною інформацією є синтезовані колекції MongoDB.

Кількість екземплярів колекцій і максимальна кількість структурних елементів в колекції залежить від предметної області.

Діаграма класів є частиною специфікації UML, в роботі розглядається специфікація UML, v. 2.4.1 [5, 6]. Оскільки вхідними даними є діаграма класів UML, то потрібно вибрати формат передачі даних між середовищем розробки програми та інструментальним засобом автоматизації процесу проектування структури БД MongoDB. Розглянемо стандарт XML Metadata Interchange (XMI) [7].

XMI – стандарт обміну метаданими за допомогою мови XML (eXtensible Markup Language), який може використовуватися для обміну будь-якими метаданими, якщо їх метамодель може бути виражена за допомогою MOF (Meta-Object-Facility – стандарт для розробки, керованої моделями). MOF (метаоб'єктний інструмент) реалізований як чотирьохшарова архітектура [8].

XMI найкраще підходить для передачі даних з діаграми класів, тому що цей стандарт був спеціально розроблений для передачі метаданих, які відповідають моделі MOF (найбільш яскравим прикладом моделі 2-го рівня MOF є метамодель UML: модель, яка описує саму мову UML). У той же час він є досить загальною специфікацією, тому що не призначений для діаграми класів конкретно. UML v. 2.4.1 відповідає XMI v. 2.4.1.

Основною перевагою використання XMI є те, що експорт UML-діаграм у форматі XMI підтримується практично кожним програмним засобом для UML-моделювання.

Специфікація UML v. 2.4.1 [5, 6] є досить об'ємною і складною, а її імплементація повинна бути якомога точніша. Такою реалізацією може служити UML2 в Java, яка імплементує UML 2.x. При тому UML2 не надає засобів моделювання UML діаграм – вона була розроблена для того, щоб бути основним будівельним блоком при створенні середовищ моделювання.

Таким середовищем може виступати, наприклад, MDT/Papyrus для Eclipse IDE (integrated development environment).

Синтез структур БД в MongoDB. На основі проведеного аналізу UML і способів обміну UML-діаграмами запропонована технологія автоматизованого синтезу структур БД (рис. 1), яка складається з:

1) виділення об'єктів з вхідного файлу;

2) інтерпретації виділених об'єктів.

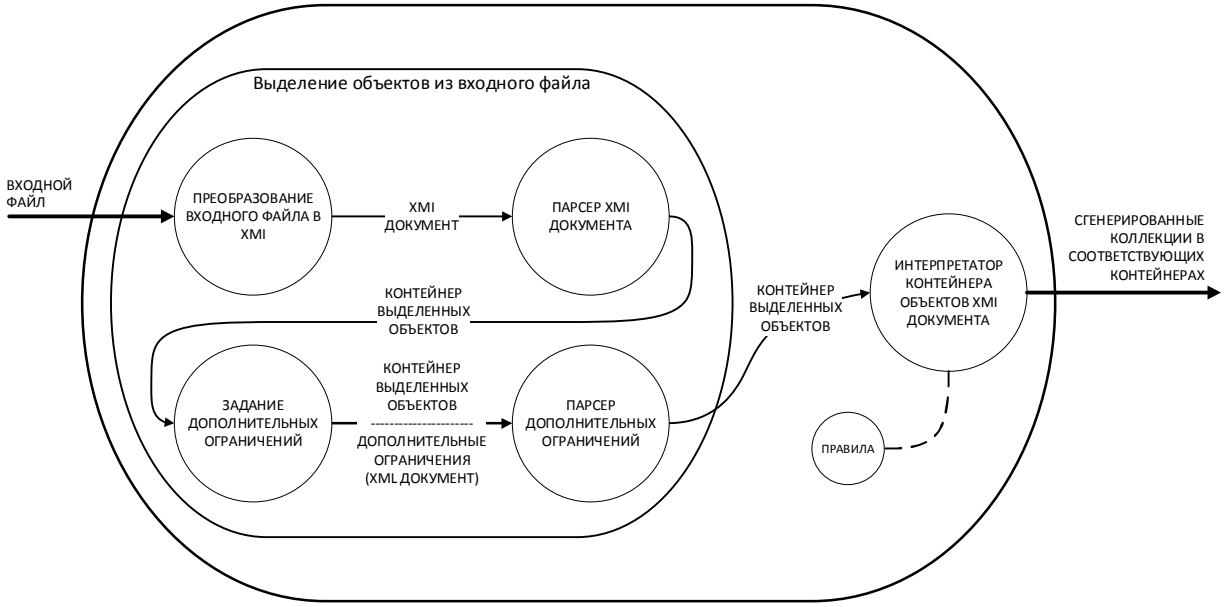

Рис. 1. Інформаційна технологія синтезу структур MongoDB

Виділення об'єктів з вхідного файлу складається з задач:

– перетворення вхідного файлу в XMI: якщо XMI документ або інший документ з структурованою інформацією про предметну область надається користувачем не в стандартному виді, а в представленні, придатному для інтерпретації, тоді виконується перетворення вхідного файлу та подання його в регламентованому вигляді;

– парсинг XMI документу (від «parse» – розбір): розбір XMI-документу і наповнення контейнера виділеними об'єктами;

– завдання користувачем додаткових обмежень, які не можуть бути описані засобами діаграм класів UML. Додаткові вихідні дані у вигляді окремого XML-документу обумовлені таким:

1) зберігання додаткових обмежень в самому XMI-документі виключає його подальшу валідацію;

2) при отриманні XML можна відразу визначити, чи правильно складений цей файл як синтаксично, так і відносно XML Schema Definition (XSD).

– парсингу додаткових обмежень: розбір XML-документа, що містить додаткові обмеження.

У компоненті системи, який відповідає за розбір (парсинг) XMI документу, розглядаються тільки ті статичні компоненти мови UML, які несуть в собі потрібну нам інформацію: класи, абстрактні класи, інтерфейси, асоціації, агрегації, композиції, узагальнення, реалізації та реалізації інтерфейсів. Всі перераховані вище компоненти в XMI поміщаються в дочірній елемент Model – Packaged Element (тег packagedElement). Варто відзначити, що Package та packagedElement є packageableElement, отже, packagedElement може міститися як безпосередньо в Model, так і групуватися всередині різних пакетів (Package).

Програмний комплекс «Підсистема автоматизації процесу проектування структур БД MongoDB» включає модулі:

1) модуль перетворення вхідного файлу в XMI-документ;

2) завдання додаткових обмежень;

3) парсер XMI-документу;

4) парсер додаткових обмежень;

5) інтерпретатор контейнера об'єктів XMI-документу.

Обов'язковими для реалізації автоматизованого синтезу є модулі (3) і (5), виконання інших залежить від повноти і способу представлених вхідних даних.

В якості парсера XMI-документу використовується Document Object Model (DOM) – об'єктна модель документів, що надається Java API. Модель DOM не накладає обмежень на зміст документу. Будь-який документ відомої структури за допомогою DOM може бути представлений у вигляді дерева вузлів, кожен вузол якого містить елемент, атрибут, текстовий, графічний або будь-який інший об'єкт. Вузли пов'язані між собою зв'язками «батько-нащадок» [9]. Основний елемент XMI, який аналізується (packageableElement), може розміщуватися як безпосередньо в Model, так і групуватися в Package(s).

Отже, перед початком розбору необхідно визначити, які саме елементи потрібно аналізувати, переглянувши XMI документ на наявність різних елементів типу Package.

В роботі вибір елементів для аналізу проводиться за такими правилами:

1) якщо в Model немає жодного елемента Package, то аналізуються всі дочірні елементи Model;

2) якщо в Model присутні елементи Package, то аналізуються дочірні елементи першого Package, який зустрічається.

Для зберігання вмісту XMI-документу в Java були розроблені спеціальні класи-контейнери, в яких може зберігатися вся інформація, необхідна для вирішення задачі (рис. 2, а).

Інтерпретатор контейнера об'єктів XMI-документу – це програмний модуль, який здійснює перетворення об'єктів в структури БД на основі механізму правил. Механізм правил засновано на 2-х об'єктах: стан (State) і дія (Action) (рис. 2, б, в). Об'єкт Rule об'єднує їх в те, що називається Правило (рис. 2, г).

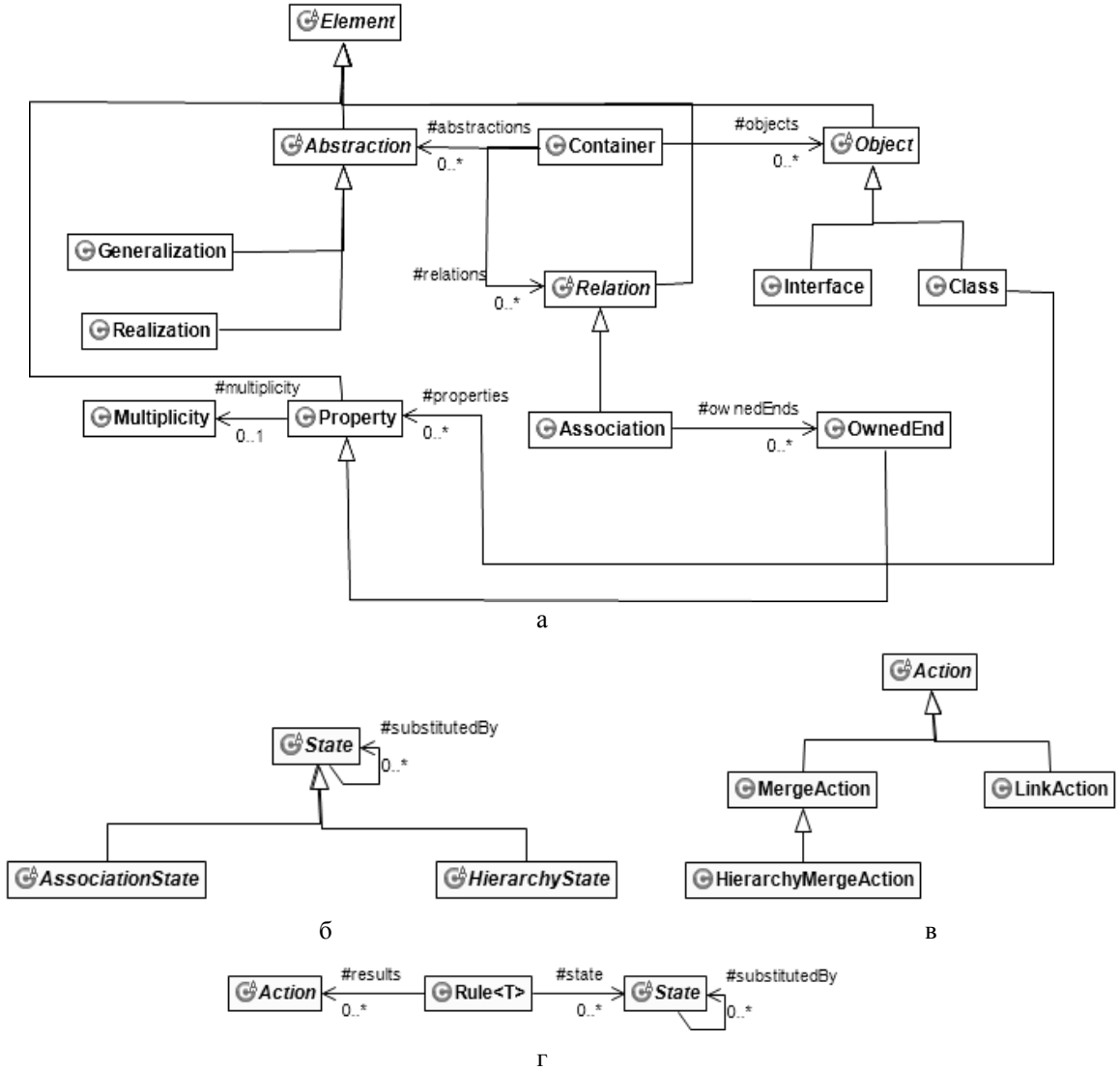

Рис. 2. Елементи системи у поданні згенерованих діаграм класів:
а – класи-контейнери вмісту XMI-документу у «згорнутому» вигляд,
б – ієрархія State, в – ієрархія Action, г – зв'язок між Rule, Action та State

У кожної ієрархії об'єктів і асоціацій є своя множина станів State. Для того, щоб оцінити відповідність правила об'єкту, який розглядається, порівнюються набори їх станів. Якщо правило підходить даному об'єкту, то необхідно застосувати до нього набір Action, що знаходяться в Rule. Action бувають 2-х основних типів: злиття (MergeAction) і посилання (LinkAction). Оскільки для кожного розглянутого об'єкта може бути декілька результатів інтерпретації, то складається дерево структур, яке зберігається в Інтерпретаторі.

Механізм правил є розширюваним у 2-х розуміннях:

а) при додаванні в програму нових об'єктів, які успадковують абстрактний клас State, вони можуть бути в подальшому використані для побудови більш комплексних правил;

б) правила в програмі мають цілком чітку структуру, тому для створення нових правил можуть використовуватися зовнішні дані (наприклад, спеціально створений файл). Для реалізації такого підходу була розроблена XML-схема, яка дозволяє ство-

рювати XML-документи, за допомогою яких можна формувати нові правила шляхом додавання XML-файл у певну директорію.

Для того, щоб користувачеві було легше створювати нові правила, під час запуску створюється XML-схема, яка містить назву всіх допустимих State.

Висновки

Сформульовано три основні етапи автоматизованого проектування структур БД MongoDB: синтез структур БД, оцінювання цих структур та багатокритеріальний вибір кращого варіанта структури. У даній роботі реалізовано перший етап.

Аналіз різних моделей даних і об'єктів на етапах проектування системи показав, що можливість автоматизації процесу проектування залежить від того, наскільки добре структуровані вхідні дані. Тому спочатку був виконаний пошук такої структури вхідних даних, яка б одночасно надавала вичерпну та строго структуровану, але не надмірну інформацію про предметну область і була б проста, знайома і зручна користувачам. В якості такої структури обрана діаграма класів UML. Для передачі діаграм між програмним продуктом користувача та створеним програмним комплексом використовується специфікація XMI.

З метою вирішення поставленої задачі запропонована технологія автоматизованого синтезу, для якої програмно реалізовані основні модулі – розбору та інтерпретації контейнера об'єктів XMI-документу,

Також створено механізм правил з можливістю розширення, що дозволяє користувачеві задавати або розширювати способи перетворення вхідних даних у вихідні.

Подальші дослідження пов'язані з пошуком оптимальних способів проведення оцінювання згенерованих структур БД.

Список літератури

1. Стрельченко В.В. Автоматизація процедур проектування та тестування структур бази даних MongoDB / В.В. Стрельченко // Збірник матеріалів 18-го Міжнар. молодіжного форуму «Радіоелектроніка і молодь у XXI столітті». – Х.: ХНУРЕ, 2014. – Т. 6. – С. 329-330.

2. JSON [Електронний ресурс]. – Режим доступу: http://json.org. – 18.02.2014 – Introducing JSON.

3. BSON – Binary JSON [Електронний ресурс]. – Режим доступу: http://bsonspec.org. – 18.02.2014. – BSON.

4. UML [Електронний ресурс]. – Режим доступу: http://www.omg.org/spec/UML. – 28.02.2014 – Unified Modeling Language™ (UML®).

5. ISO/IEC 19505-1:2012(E). Information technology – Object Management Group. Unified Modeling Language (OMG UML), Infrastructure. – 2012. – 234 p.

6. ISO/IEC 19505-1:2012(E). Information technology – Object Management Group. Unified Modeling Language (OMG UML), Superstructure. – 2012. – 758 p.

7. formal/2013-06-03. OMG MOF 2 XMI Mapping Specification. Version 2.4.1 – 2013. – 108 p.

8. MetaObject Facility [Електронний ресурс]. – Режим доступу: http://www.omg.org/mof. – 28.11.2014. – OMG's MetaObject Facility.

9. Блинов И.Н. Java: промышленное программирование / И.Н. Блинов, В.С. Романчик. – Минск : Универсал Пресс, 2007. – 704 с.

Надійшла до редколегії 27.05.2015

Рецензент: д-р техн. наук, проф. В.О. Філатов, Харківський національний університет радіоелектроніки, Харків.

АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ СТРУКТУР БАЗЫ ДАННЫХ MONGODB

Н.И. Калита, В.В. Стрельченко

Предложена технология, которая позволяет на основе XMI-документа или другого способа представления информации о предметной области в структурированном виде синтезировать структуры базы данных MongoDB. Программная реализация технологии предназначена для использования в составе системы автоматизированного проектирования MongoDB системным архитектором на этапе проектирования информационного обеспечения компьютеризованных систем управления. Разработанная технология позволяет сократить временные затраты на проектирование путем формализации подхода к нему.

Ключевые слова: *документно-ориентированная база данных, проектирование БД, автоматизированный синтез структур БД, САПР, MongoDB, XML, XMI, UML.*

AUTOMATION OF MONGODB DATABASE STRUCTURE DESIGN

N.I. Kalita, V.V. Strelchenko

The work proposes the technology that allows for a synthesis of MongoDB data structures based on an XMI-document or other means to provide information about the domain in a structured way. A software implementation of this technology is to be used as a part of MongoDB CAD on the stage of information module design for computerized management systems by a system architect; this reduces the time spent on design by formalizing the approach.

Keywords: *Document-oriented databases, DB design automated DB structure synthesis, CAD, MongoDB, XML, XMI, UML.*