Розроблено програмні засоби опрацювання інформаційних ресурсів систем електронної контент-комерції. З позиції системного підходу застосовано принципи опрацювання інформаційних ресурсів у СЕКК, що дало змогу розробити методи формування, управління та реалізації комерційного контенту. Реалізовано комплексний метод супроводу контенту, що дає можливість розробити модуль реалізації комерційного контенту.

1. Берко А. Системи електронної контент-комерції / А. Берко, В. Висоцька, В. Пасічник. – Л.: Вид-во Львівської політехніки, 2009. – 612 с. 2. Большакова Е. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика / Е. Большакова, Д. Ландэ, А. Носков, Э. Клышинский, О. Пескова, Е. Ягунова. – М: МИЭМ, 2011. –272 с. 3. Брайчевский С. Современные информационные потоки / С. Брайчевский, Д. Ландэ // Научно-техническая информация. – 2005. – № 11. – С. 21–33. 4. Клифтон Б. Google Analytics: профессиональный анализ посещаемости веб-сайтов / Б. Клифтон. – М: Вильямс, 2009. – 400 с. 5. Корнеев В. Базы данных. Интеллектуальная обработка информации / В. Корнеев, А. Гареев, С. Васютин, В. Райх. – М: Нолидж, 2000. – 352 с. 6. Ландэ Д. Основы моделирования и оценки електронных информационных потоков / Д. Ландэ, В. Фурашев, С. Брайчевский, О. Григорьев. – К: Інжиніринг, 2006. – 348 с. 7. Ландэ Д. Основы интеграции информационных потоков: монография / Д. Ландэ. – К: Інжиніринг, 2006. – 240 с. 8. Пасічник В. Математична лінгвістика / В. Висоцька, В. Пасічник, Ю. Щербина, Т. Шестакевич. – Л: "Новий Світ – 2000", 2012. – 359 с. 9. Советов Б. Моделирование систем / Б. Советов, С. Яковлев. – М: ВШ, 1998. 10. Федорчук А. Контент-мониторинг информационных потоков / А. Федорчук. – К., 2005. – № 3.

УДК 004.43

О.В. Годич, Ю.О. Прокопів
Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж

ВІЗУАЛЬНА ПРЕДМЕТНО-ОРІЄНТОВАНА МОВА ЗАПИТІВ

Подано результати створення візуальної предметно-орієнтованої мови запитів, яка є візуальним засобом взаємодії з даними і побудови правил моніторингу стану інформаційних систем. Використання розробленої мови надає експертам предметної області потрібні засоби для розширення системи ad-hoc у процесі її використання з метою вчасного реагування на виробничі потреби.

Ключові слова: предметно-орієнтована мова, інформаційні запити, проекційне редагування, HCI.

This article discusses a visual domain-specific query language that supports data interaction and composition of business rules as part of a software system, which can be used directly by domain experts as part of the information system itself. Such approach provides domain experts with necessary tools to enhance the live system in order to meet dynamically changing real-life requirements without the tedious and often complex development/deployment cycles currently used in the software industry.

Key words: domain-specific language, data querying, projectional editing, HCI.

Вступ. Постановка проблеми.

Мова є невід'ємною характеристикою людини, і можливості мов (природних і формальних) є визначальними щодо здатності мислити й успішно вирішувати складні проблеми. Наприклад,

здатність до обчислень з використанням римської системи числення є незрівнянно нижчою, ніж із використанням індо-арабської позиційної системи [1–3].

Історично так склалося, що мови програмування загального призначення (Тюрінг-повні мови) використовуються для створення складних інформаційних систем виробничого спрямування (enterprise resource planning systems, asset maintenance systems тощо). Так само як з іншими формальними і природними мовами, мови програмування загального призначення характеризуються синтаксисом і семантикою. В основу семантики мов програмування загального призначення покладено обчислювальну модель. Нині домінуючою є імперативна парадигма програмування. Мови, які належать до цієї парадигми, концептуалізують обчислення у вигляді послідовності чітко визначених команд (слово за словом), які за смислом близькі до низькорівневих машинних кодів. Це фактично робить такі мови програмування добрим інструментом для інструктування комп'ютерів того, що слід робити, але невідповідним інструментом для опису проблем реального світу [4, 5]. Нині ще досі відбуваються дебати серед фахівців щодо переваг і недоліків використання предметно-орієнтованих мов (Domain Specific Languages, DSL), які за своєю суттю є декларативними порівняно із мовами загального призначення (General Purpose Languages, GPL) у контексті створення інформаційних систем. DSL і GPL мови володіють унікальними наборами властивостей для подолання різних аспектів складності створення програмного забезпечення [6] і сучасні тренди вказують на поєднання обох підходів [7].

Теорія і практика створення предметно-орієнтованих мов репрезентовано низкою різних підходів. Зокрема, одним із найсучасніших є підхід на основі проекційного редагуванням [8, 9], але й традиційні підходи із використанням текстової репрезентації також є поширеними (Spoofax, http://strategoxt.org/Spoofax, Xtext, http://www.eclipse.org/Xtext). Засоби побудови предметно-орієнтованих мов узагальнено під назвою "мовні воркбенчі" (projectional/textual language workbench) [10].

Усі ці підходи до створення предметно-орієнтованих мов розглядають процес розроблення інформаційних систем як зовнішній до самих систем – інформаційні системи є продуктом застосування DSL. На наше переконання інформаційні системи, у сенсі кінцевого продукту, з яким безпосередньо взаємодіють користувачі, повинні містити засоби, які надавали б можливість підлаштування і розширення систем в режимі ad-hoc без залучення її розробників. Забезпечення таких засобів повинно реалізовуватися у вигляді вбудованих у систему предметно-орієнтованих мов. У цій статті обговорено результати зі створення візуальної предметно-орієнтованої мови запитів (visual domain-specific query language, VDSQL) як інтегрованої частини інформаційної системи. Мову VDSQL створено на принципах проекційного редагування і спрямовано на побудову інформаційної функції систем [11] у вигляді вбудованої у систему мови. Однією із важливих характеристик розробленої мови є застосовність до інформаційних систем, в основу яких покладено об'єктно-орієнтовану модель, а дані зберігаються у реляційних базах даних. Отже, одним з ключових завдань цієї мови було подолання об'єктно-реляційної несумісності (object-relational impedance mismatch).

Аналіз останніх досліджень та публікацій

З огляду на те, що результатом досліджень, який подано у цій статті, є візуальна предметно-орієнтована мова для побудови інформаційних запитів, у цьому розділі розглянуто поточний стан досліджень саме у напряму створення візуальних мов запитів до баз даних (data querying). Головними критеріями оцінювання існуючих мов є зручність їхнього використання і набір функціональних властивостей з погляду побудови запитів до даних (наприклад, можливість застосування агрегаційних функцій, побудова обчислювальних виразів тощо).

Hyperflow [12]. Поєднує в собі характеристики візуальних мов запитів і мов візуальних потоків даних. Структура речень мови базована на використанні орієнтованих графів та функціональних структур потоків даних. Конструкції графів використовуються для вибору тієї частини предметної області, яка цікавить користувача. Структури потоків даних використовуються

для визначення операцій групування і сортування. Використання контекстно-залежного меню допомагає користувачеві у побудові коректних запитів.

GLASS [13]. Graphical Query Language for Semi-Structured Data – мова для візуальних запитів до XML-документів. Ґрунтується на мові схеми ORA-SS (Object-Relationship-Attribute model for Semi-Structured data), речення якої формується у спосіб побудови орієнтованого графу залежностей між сутностями. Логічні вирази будуються в текстовому вигляді (включно із кванторами існування і загальності).

Klaeidoquery [14]. Логічні вирази на цій мові будуються візуально. Для задання кон'юнкції чи диз'юнкції використовується послідовне та паралельне з'єднання умов відповідно. Заперечення логічної умови також задається графічним способом. Квантифікація колекційних властивостей (один до багатьох чи багато до багатьох) здійснюється за допомогою візуальних засобів.

GOQL [15]. Особливість цієї мови полягає у можливості подання логічних частин запитів у вигляді блоків-коробок (boxes), які потім можна використовувати як елементарні частини запитів. Наприклад, щоб побудувати об'єднання використовується коробка *union*, в яку покладено менші коробки запитів. Логічні умови конструюються за допомогою блоків *and* та *or*, з'єднаних між собою зв'язками.

Більшість із розглянутих мов репрезентують структуру речень, а отже, й запитів, за допомогою графів. Графи є найбільш гнучким способом подання об'єктів інформаційної системи і зв'язків між ними. Водночас використання графів як засобу візуалізації ускладнює сприйняття для користувачів. Це зумовлено тим, що користувачі повинні чітко розуміти природу зв'язку, який зображено дугою між двома об'єктами. У цьому випадку для кращого розуміння природи зв'язку біля дуги ставлять різноманітні позначення, наприклад 1..1 або 1..n.

Використання вкладених блоків є альтернативним способом задавання зв'язків між об'єктами. Таким способом можна задавати зв'язки між об'єктами у вигляді вкладених блокових структур, уникаючи великої кількості дуг, що притаманні графовим поданням. Одними із найбільш сучасних візуальних мов, які належать до мов загального призначення, є Blockly (https://code.google.com/p/blockly/) і Scratch (http://scratch.mit.edu/)**.** Ці мови розроблені спеціально для навчання дітей програмуванню і для розвитку їхнього обчислювального мислення (computational thinking). Саме ідеї Scratch було покладено в основу розробленої мови VDSQL щодо способу візуалізації. Scratch базований на метафорі побудови блоків, в якій програми створюють шляхом маніпулювання блоками, стикуючи їх між собою як частинки пазла. Мовні команди і типи подаються блоками різних форм з частинами, які можуть бути складені разом тільки в синтаксично правильному порядку. Це є прикладом проекційного редагування, яке не допускає синтаксичних помилок, що є одним із труднощів, зокрема під час вивчення мов програмування. Такий підхід дає можливість користувачам зосередитися на задачах, які треба розв'язувати, а не на "механічних" аспектах програмування. Користувач перетягує командні блоки з палітри, щоб створити стиковані нагромадження блоків (процедури), які керують поведінкою програми. Різні стеки блоків можуть виконуватися паралельно [16–19]. На рис. 1 зображено приклад короткої програми мовою Scratch.

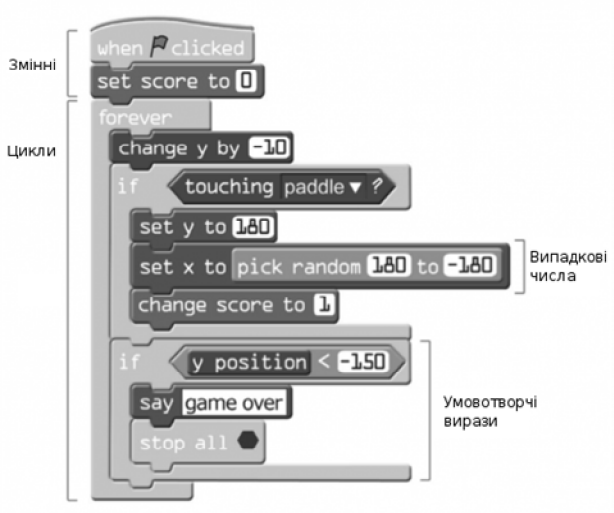

Рис. 1. Приклад програми мовою Scratch

Поширеність і зручність використання Scratch дає підстави говорити, що саме блоковий підхід до візуалізації є кращим для сприйняття, ніж графовий, який притаманний іншим візуальним мовам запитів. Процес формування запитів не лише має бути простим для розуміння, але й приносити задоволення. Це сприяє кращому обміну досвідом між користувачами та глибшому розумінню предметної області, сприяє подальшому уточненню деталей предметної області з покращенням інформаційної системи загалом.

Візуальне подання мови VDSQL

Однією з мотивацій використання візуального подання мов програмування є те, що такий спосіб забезпечує опис бажаних дій на вищому рівні абстракції, який дає можливість уникнути потреб вивчення складного синтаксису. У своїх дослідженнях B. Myers зазначає, що візуальний стиль програмування є простішим для розуміння особливо для *не* програмістів чи програмістів-новачків [20]. З огляду на те, що VDSQL напрямлена насамперед на експертів предметної області і досвідчених користувачів системою, тобто осіб, не дотичних до програмування, то актуальність графічного подання зростає. У результатах досліджень O. Clarisse і S. Chang з використання візуальних мов стверджується, що формат візуального програмування є ближчим, ніж текстовий, до ментальних репрезентацій вирішуваних користувачами завдань [21].

Альтернативою до традиційного способу подання мов програмування, де мова подається послідовністю символів і де семантичному аналізу передує синтаксичний аналіз, є подання мов у вигляді певної абстрактної моделі, яка редагується у спосіб проектування [22]. На рис. 2 схематично зображено головну ідею проекційного редагування. Візуальна репрезентація, із якою користувачі взаємодіють безпосередньо, проектується у деяку абстрактну репрезентацію. У такий спосіб, маніпулюючи графічними примітивами мови редагується абстрактна модель (наприклад, абстрактного синтаксичного дерева), яку можна перетворити у виконувану репрезентацію, а також збережувану для пізнішого повторного використання.

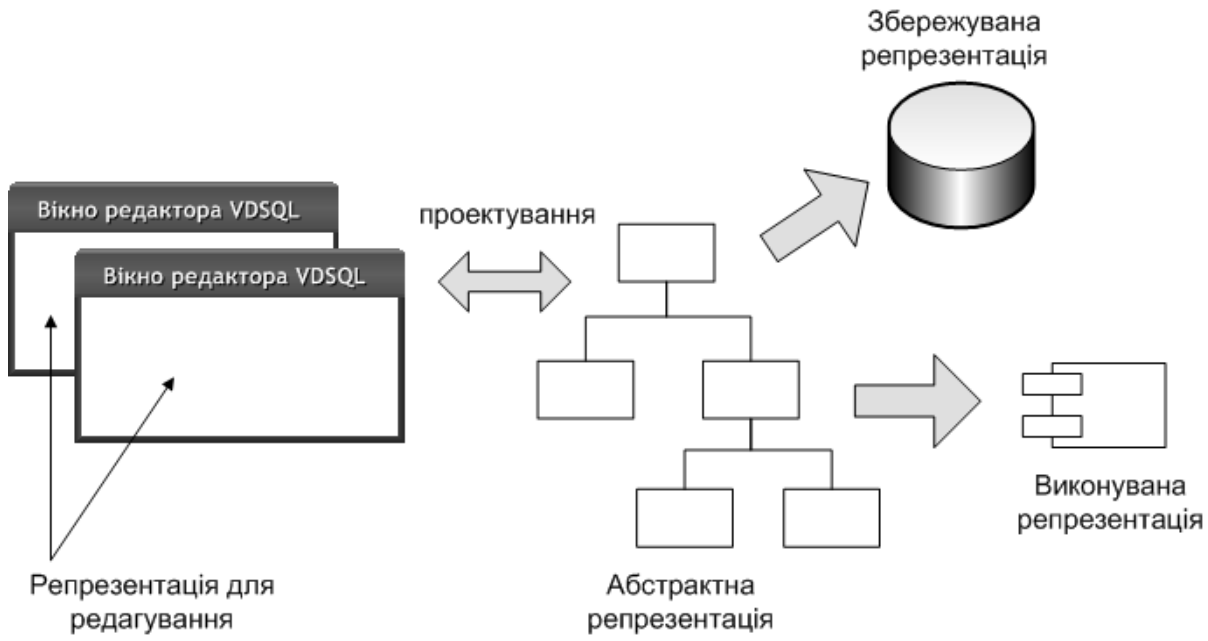

Рис. 2. Схема проекційного редагування

Мова VDSQL ґрунтується на *проекційному редагуванні*, яке ще називають *структурне редагування*. Головною метою використання проекційного способу редагування є гарантування синтаксичної коректності утворених мовних конструкцій, чого вдається досягти без потреби синтаксичного аналізу [23]. Візуальне подання мови дає можливість забезпечити таку підтримку побудови речень, що унеможливлює створення синтаксично не коректних речень. При цьому

середовище візуального редагування повинно містити контекстну допомогу, аби підказувати користувачам можливі способи побудови речень. Реалізацію такої систему контекстної допомоги для VDSQL обговорено в наступному підрозділі цієї статті.

Головною метою будь-якої предметно-орієнтованої мови є надання зручного способу взаємодії користувачів зі семантичною моделлю, яку покладено в основу інформаційної системи. На поточному етапі дослідження мова VDSQL надає інтерактивні засоби для формування запитів до семантичної моделі, яку подано у вигляді об'єктно-орієнтованої моделі. Використовуючи VDSQL, можна вибирати дані у спосіб взаємодії із об'єктно-орієнтованою моделлю системи, а також специфікувати правила моніторингу стану інформаційної системи, порушення яких автоматично повідомляється користувачам системи за допомогою обраного способу (ел. поштою, SMS тощо).

Об'єктно-орієнтована модель й елементи мови VDSQL

З погляду об'єктно-орієнтованої моделі інформаційної системи, концептуальними засадами мови VDSQL є принцип *прямої маніпуляції* (direct manipulation principle) [24]. Відповідно до цього принципу самі об'єкти, які моделюють сутності предметної області, є центральними елементами мови VDSQL, щодо яких формуються запити і правила на цій мові. Об'єкти з їхніми властивостями зображаються графічно і користувач може маніпулювати ними безпосередньо за допомогою маніпулятора миші на екрані монітора або використовуючи сенсорний механізм вводу планшетів і смартфонів. На рис. 3 подано приклад блокового подання об'єкта, що моделює сутність "замовлення" (direct charge purchase order) із деякої предметної області. Блок-об'єкт може містити довільну кількість блоків, які зображають властивості цього об'єкта – ті властивості, які не беруть участі у формуванні запиту чи правила можуть не бути доданими з метою спрощення сприйняття інформації. Кожен блок, який репрезентує властивість об'єкта, можна використати для формування запиту у спосіб накладання на властивість обмежень, або ж її використання як частина іншого обмеження. Як можна бачити з рис. 3, додатково до властивостей, які визначені на рівні семантичної моделі, об'єкти можуть бути динамічно розширені обчислювальними властивостями, формулу обчислення яких визначає користувач.

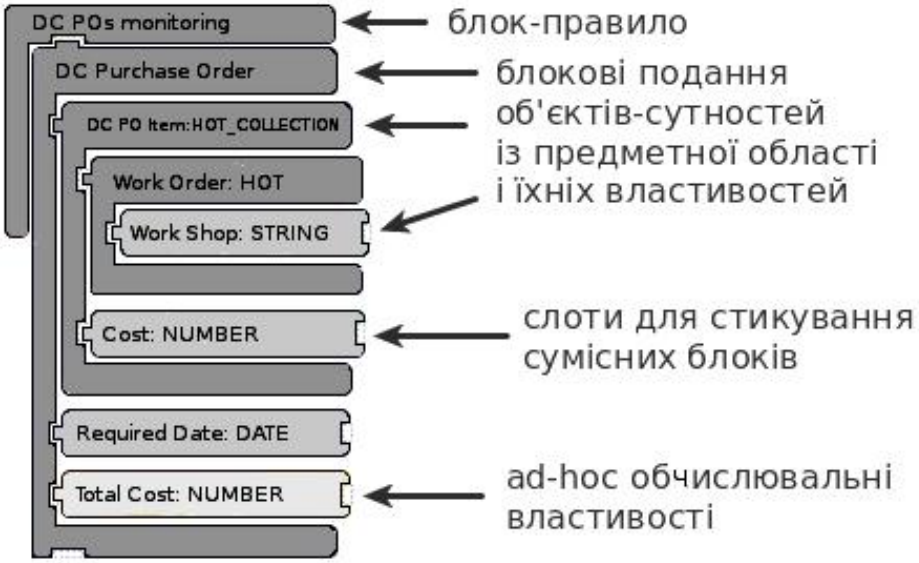

Рис. 3. Блокова структура для подання об'єктів-сутностей з їхніми властивостями

Елементарними складовими мови VDSQL є чітко визначена множина блоків, які можна комбінувати у спосіб стикування один до одного з метою формування складених виразів. Блоки поділяються на дві групи. Одна група репрезентує об'єктно-орієнтовану модель предметної області (сутності та їхні властивості), інша – елементарні складові мови, які репрезентують операції (логічні операції порівняння, операції агрегування тощо) і літерали. Кожен блок має один чи більше слотів, до яких можуть долучатися інші сумісні блоки. Слоти є тією абстракцією, що дає

можливість асоціювати елементи мови, подані блоками, між собою для побудови коректних виразів. Залежно від природи блоку і місця його розташування слоти набувають конкретного семантичного навантаження, яке визначає їхню сумісність із рештою блоків. Саме механізм динамічного визначення семантичної сумісності блоків гарантує побудову коректних виразів.

Редактор VDSQL надає два способи стикування блоків між собою, які доповнюють один одного. Перший спосіб реалізується за допомогою операції D'n'D (drag and drop), де один або група стикованих блоків переносяться і стикується до блоку зі сумісним слотом. Інший спосіб – це механізм вибору сумісних блоків із контекстно залежної палітри, яка візуалізує множину допустимих блоків для продовження побудови виразу на запит користувача. Надання палітри сумісних блоків на вибір відбувається у момент активування слоту (наприклад, вказавши на слот курсором миші), з якого користувач хоче розпочати чи продовжити побудову виразу.

На рис. 4 зображено D'n'D дію перенесення блоку-властивості "Price/Ltr" (ціна пального за літр) для його стикування у слот блоку, який репрезентує операцію множення, з метою побудови виразу для обмеження властивості "Purchase Price" (ціна закупівлі).

На рис. 5 зображено другий спосіб стикування блоків – контекстно залежна палітра блоків, яка випадає у момент активування слоту. У цьому випадку перелік можливих операцій з метою подальшої розбудови виразу для обмеження властивості "Purchase Price" (ціна закупівлі) складається з арифметичних, логічних і агрегаційних операцій. Лише сумісні у цьому контексті операції пропонуються на вибір користувача. Детальний опис механізму визначення контекстно залежної сумісності блоків є темою майбутніх публікацій. Водночас зауважимо, що в основу цього механізму покладено теорію типів, яка формалізує процес визначення типу виразу та його складових (type inference) на етапі редагування – це й визначає сумісність блоків та їхніх комбінацій.

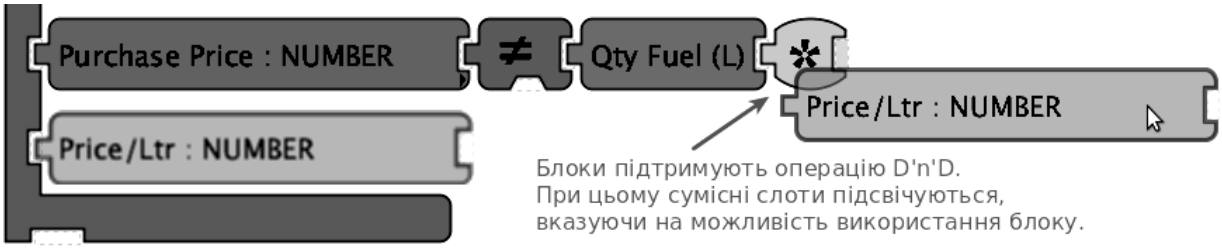

Рис. 4: Ілюстрація D'n'D способу стикування блоків

Рис. 5. Ілюстрація використання контекстно залежної палітри для стикування блоків

Група блоків, які репрезентують операції і літерали, є доволі численною, а правила стикування блоків відповідно до механізму семантичної сумісності є достатньо складними для вичерпного переліку й опису у статті. Тому з метою висвітлення деяких можливості мови VDSQL розглянемо низку ілюстративних прикладів.

На рис. 6 зображено вираз для властивості “property1” типу String деякого об’єкта предметної області. Цей вираз є частиною запиту до семантичної моделі інформаційної системи, який накладає обмеження на цю властивість. Поданий вираз містить два обмеження, що поєднані логічною операцію кон’юнкції (блок AND). Перше обмеження (ліворуч від блоку AND) вимагає, аби значення властивості “property1” було меншим або дорівнювало (у сенсі порівняння величин типу String) значенню “H”, поданому блоком-літералом. Друге обмеження (праворуч від блоку AND) вимагає, аби значення тієї ж властивості закінчувалося значенням “ello,world”. Це обмеження використовує операцію конкатенації двох літералів лише для ілюстрації такої можливості.

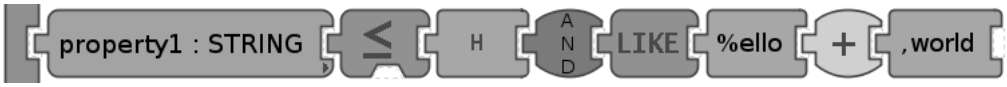

Рис. 6. Вираз-умова на властивість типу String

Рис. 7 ілюструє вираз-обмеження на властивість типу Date. Які в попередньому прикладі, у цьому обмеженні присутні дві умови, поєднані логічною операцією диз’юнкції. Перше обмеження використовує операцію порівняння дат в сенсі їхньої різниці у 15 днів. Тобто вимагається, щоб значення властивості “property3” дорівнювало вказаному літералу типу Date (23 листопада 2003 року) або було меншим принаймні на 15 днів. Таке обмеження є прикладом компактності подання доволі складної умови порівняно із текстовою альтернативою. Зауважте, що можливість використання такого обмеження є контекстно залежне і застосовне лише у випадку порівняння величин типу Date. Інше обмеження (праворуч від блоку OR) вимагає, щоб значення властивості було в межах попереднього місяця (IN PREV M, де місяць визначається в момент обчислення виразу) із допустимим відставанням у 19 днів.

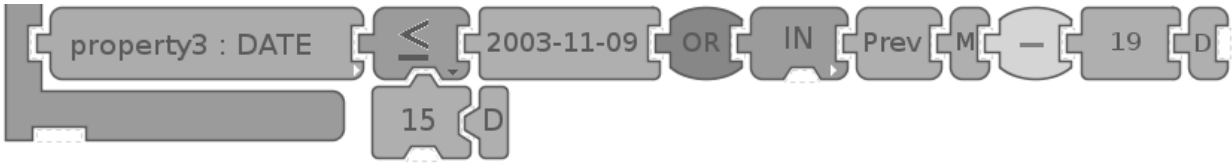

Рис. 7. Вираз-умова на властивість типу Date

На рис. 8 подано обмеження на властивість числового типу (NUMBER). Попри компактність подання цього виразу, він репрезентує доволі складне обчислення. Цей вираз обмежує значення властивості так, щоб воно було меншим або дорівнювало значенню $28.3+(-3.8)$ у межах абсолютного значення 17.6 або 5.5%.

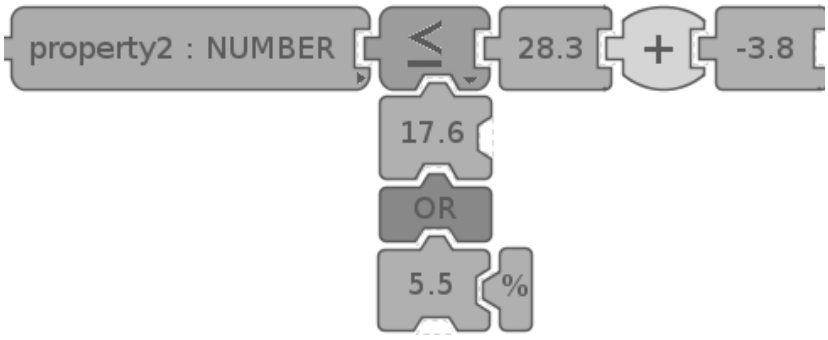

Рис. 8. Вираз-умова на властивість типу Number

Вирази на рис. 7 і 8 ілюструють приклади складних умов для побудови запитів, використовуючи компактну та інтуїтивно зрозумілу візуальну форму. Можливості редактора для мов на основі проекційного редагування є більш визначальними для зручності використання мови, ніж це є для текстових мов. Завдяки механізму семантичної сумісності блоків і контекстно залежної палітри блоків, а також з огляду на впровадження принципу безпосередньої маніпуляції з об'єктно-орієнтованою моделлю, використання мови VDSQL забезпечує інтуїтивний і високорівневий спосіб взаємодії з інформаційною системою.

Робоча область редактора VDSQL є необмеженою. Тому з метою навігації він реалізує концепцію Zooming User Interface (ZUI) [25]. Це забезпечує можливість змінювати масштаб зображених на робочому просторі виразів і переміщувати цілий робочий простір, використовуючи панорамування. За допомогою такого механізму користувачі отримують можливість зосереджуватися на різних деталях побудованих виразів. Зокрема це дає змогу зосередитися на деталях окремого запиту для його перегляду чи редагування або оглянути усі запити в сукупності з метою швидкого переміщення між ними чи здійснення загального оцінювання складності побудованих правил.

Рис. 9 ілюструє використання ZUI. Рис. 9, *а* містить п'ять об'єктів-сутностей предметної області, зображених усіх одночасно. Рис. 9, *б* і 9, *в* ілюструють побудову виразу у масштабі 150 %. На рис. 9, *б* зображено момент перетягування блоку, а на 9, *в* – результат стикування цього блоку.

Репрезентації мови VDSQL

Усім мовам, які реалізують принцип проекційного редагування, притаманні три репрезентації: абстрактна, збережувана і виконувана. Абстрактна репрезентація є проміжною ланкою між тим поданням мови, яке спостерігає користувач, і двома іншими репрезентаціями.

Абстрактною репрезентацією VDSQL є абстрактне синтаксичне дерево (abstract syntax tree, AST), яке формується у результаті безпосереднього створення і редагування виразів у візуальний спосіб. Абстрактне синтаксичне дерево є зручною моделлю, оскільки надається до простого способу трансформування в інші репрезентації шляхом здійснення обходу дерева. Саме у такий спосіб будуються репрезентація для збереження з метою пізнішого відтворення запитів, а також репрезентація для виконання виразів.

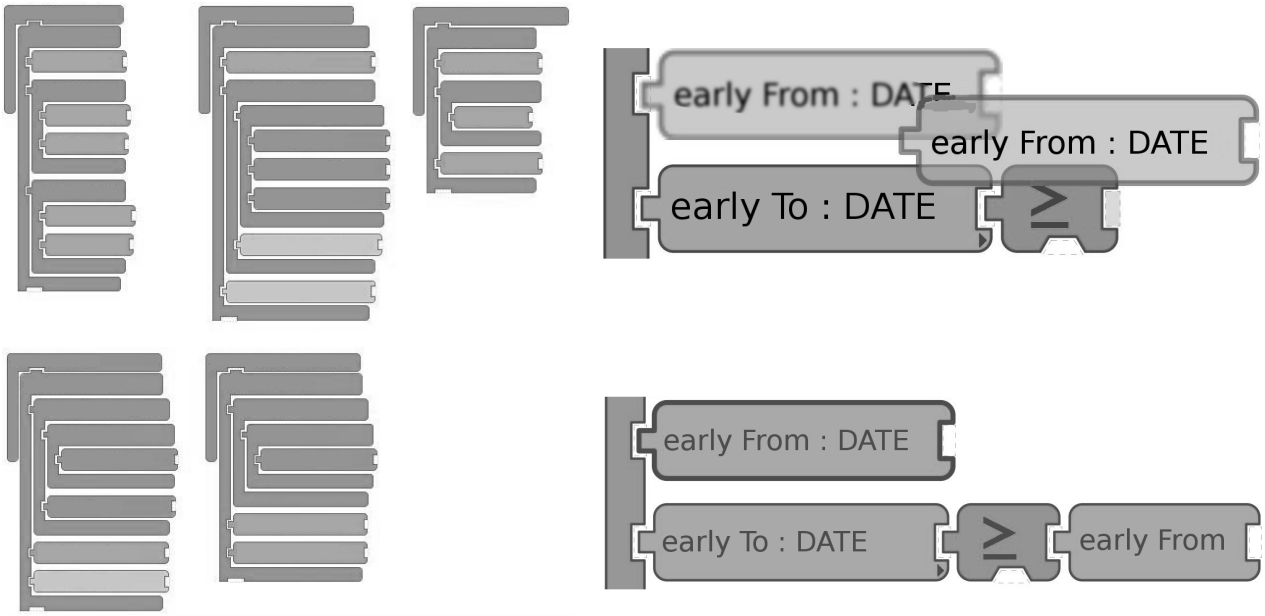

Рис. 9: Редактор VDSQL реалізує концепцію ZUI: а – перегляд об'єктів (масштаб 33 %); б – перетягування блоку для стикування (масштаб 150 %); в – результати стикування (масштаб 150 %)

Головною вимогою до збережуваної репрезентації є взаємно однозначна відповідність із абстрактною репрезентацією. Цього можна досягти доволі просто, зокрема, реалізувавши

серіалізацію AST у вигляді довільного формату, який підтримує вкладені структури. Серед поширених форматів такими є XML і JSON. У нашому досліджені для серіалізації AST було використано формат XML.

Побудова виконуваної репрезентація є значно складнішим завданням і вимагає детальнішого обговорення. Як вже зазначалося у попередніх розділах, мова VDSQL застосовна для побудови запитів і правил моніторингу інформаційних систем, які у своїй основі мають об'єктно-орієнтовану модель зі збереженням стану в реляційних базах даних. Одним із найважливіших критеріїв до побудови виконуваної моделі було поставлено її семантичну прозорість із тим механізмом побудови запитів, який використовується у побудові і модифікації інформаційної системи на рівні коду. Саме тому було розроблено API взаємодії між об'єктно-орієнтованою і реляційною моделями, який має форму вбудованої у Java спеціалізованої мови із назвою Entity Query Language (EQL). Ця мова дає можливість формулювання запитів до реляційних баз даних як частини коду інформаційної системи мовою Java, де рівнем абстракції є саме об'єктно-орієнтована модель системи, що й вирішує питання об'єктно-реляційної несумісності [27]. Мова EQL є виконуваною репрезентацією мови VDSQL, яка бере на себе завдання транслювання об'єктно-орієнтованих запитів у SQL. Завдяки такому підходу зберігається відповідність між запитами, які формують користувачі мовою VDSQL і розробники інформаційної системи мовою EQL. Приклад доволі складного запиту мовою EQL, який використовує розвинений агрегаційний апарат, подано на рис. 10. В основу цього запиту покладено об'єкт-сутність "наряд на роботу" (work order) із предметної області автомобільного парку. Зображений на рисунку код є стандартним Java кодом, але завдяки способу побудови відповідного API має характеристики вбудованої спеціалізованої мови (embedded DSL) [10].

```
1   select(WorkOrder.class).
2   where().
3   prop("vehicle.model.make.key").eq().val("MERCEDES").and().
4   begin().
5     prop("vehicle.model.key").starts_with().any_of_values("315", "316").or().
6     prop("vehicle.model.key").eq().val("VITO").
7   end().and().
8   year_of().prop("actualStart").in().values(2009, 2010, 2011).and().
9   prop("vehicle.station.zone.sector.division.key").eq().val("NORTH").
10  yield_and_group().prop("vehicle.station.zone.sector").as("sector").
11  yield().
12    begin_expr().
13      sum_of().prop("actualCost").div().val(3).
14    end_expr().as("averageYearlyMaintenanceCostPerSector").
15  model_as_aggregate();
```

Рис. 10. Редактор VDSQL реалізує концепцію ZUI

Висновки

Подані у статті результати є завершенням першого етапу досліджень зі створення мовно-орієнтованих засобів побудови і підтримки життєвого циклу інформаційних систем, семантична модель яких є об'єктно-орієнтована. Розроблену візуальну мову VDSQL можна використовувати як інтегровану частину інформаційних систем з метою модифікації існуючої чи створення нової функціональності щодо їхньої інформаційної функції.

Концепція проекційного редагування забезпечує ефективний спосіб побудови коректних виразів мовою VDSQL, що разом із можливостями редактора забезпечує її інтуїтивне використання. Значну увагу в дослідженні звернено на спосіб подання мови VDSQL. Із відомих нам результатів VDSQL є першою спробою використання підходу із блоками, які можна стикувати, з метою подання мовних елементів і реалізації структурного способу формування виразів для побудови інформаційної функції систем. З огляду на результати багаторічних досліджень проекту Scratch при Массачусетському технологічному університеті використаний підхід дає надію на швидке опанування мовою недосвідчених користувачів системи [26]. Водночас цей підхід не привносить суттєвих обмежень щодо функціональних можливостей мови VDSQL.

Важливим аспектом дослідження є розроблення ефективної виконуваної моделі, до якої трансформується абстрактна репрезентація мови VDSQL. Розроблена мовою Java модель дає змогу виконувати об'єктно-орієнтовані запити, які транслюються у відповідні запити мовою SQL. Саме завдяки виконуваній моделі у вигляді EQL API виконання запитів мовою VDSQL досягається обходом відповідного абстрактного синтаксичного дерева, у результаті чого формується запит на EQL API [27].

Подальші етапи дослідження з розвитку VDSQL стосуватимуться розширенню функціональних можливостей мови у напрямку не лише формування запитів до існуючих об'єктів-сутностей системи, але й модифікації та створення нових сутностей. Така функціональність забезпечить можливість побудови моделі предметної області засобами, які є вбудовані у саму інформаційну систему. Також це дає надію на можливість автоматизації низки важливих аспектів життєвого циклу інформаційних систем, таких як еволюції бізнес-правил, які досі не є складовою самих систем.

1. Whoft B. L. Language, Thought and Reality (2nd edition) / B. L. Whoft – The MIT Press, 2013. – 448 p. 2. McConnel S. Code Complete (2nd edition) / S. McConnel – Microsoft Press, 2004. – 960 p. 3. Scott M. L. Programming Languages Pragmatics (3rd Edition) / M. L. Scot – Morgan Kaufman, 2008 – 944 p. 4. Knuth D. E. Literate programming / D. E. Knuth // Computer Journal – 1984 – Vol. 27, No. 2 – P. 97-111. 5. Ward M. P. Language-Oriented Programming [Електронний ресурс]/ De Montfort University, Computer Science Department Science Labs – 1994, Режим доступу: http://www.cse.dmu.ac.uk/~mward/martin/papers/middle-out-t.pdf. 6. Brooks F. P. The Mythical Man-Month: Essays on Software Engineering (2nd edition) / F. P. Brooks – Addison-Wesley Professional, 1995 – 336 p. 7. Voelter M., Merkle B. Domain Specific -- a Binary Decision? [Електронний ресурс] / The 10th Workshop on Domain-Specific Modeling – 2010, Режим доступу: http://www.dsmforum.org/events/DSM10/Papers/Voelter.pdf. 8. Dmitriev S. Language Oriented Programming: The Next Programming Paradigm [Електронний ресурс] / JetBrains s.r.o. – 2004, Режим доступу: http://www.onboard.jetbrains.com/articles/04/10/lop/mps.pdf . 9. Simonyi C. et al. {Intentional Software [Електронний ресурс] / OOPSLA'06 – 2006, Режим доступу: http://intentsoft.wpengine.com/wp-content/uploads/2012/03/IS_OOPSLA_2006_paper.pdf. 10. Fowler M. Domain Specific Languages / M. Fowler – Addison-Wesley Professional, 2010. – 640 p. 11. Olivé A. Conceptual Modeling of Information Systems / A. Olivé– Springer, 2007. – 455 p. 12. Dotan D., Pinter R. Hyperflow: an integrated visual query and dataflow language for end-user information analysis / D. Dotan, R. Pinter // Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing – 2005 – P. 27-34. 13. Ni W., Ling T. GLASS: A graphical query language for semi-structured data / W. Ni, T. Ling 2003 // Proceedings of Eighth International Conference on Database Systems for Advanced Applications (DASFAA '03) – 2003 – P. 363-370. 14. Murray N. et al. Kaleidoquery: a flow-based visual language and its evaluation / N. Murray, N. Paton, C. Goble, J. Bryce // Journal of Visual Languages and Computing – 2000 – Vol. 11 – P. 151–189. 15. Keramopoulos E. et al. GOQL, a graphical query language for object-oriented database systems / E. Keramopoulos, P. Pouyioutas, C. Sadler // Proceedings of Basque International Workshop on Information Technology – 1997 – P. 35–45. 16. Peppler K. Creative coding: Programming for Personal Expression / K. Peppler, Y. Kafai // CSCL'09 Proceedings of the 9th International conference on Computer supported collaborative learning. –2005. – 76 p. 17. Resnick M. Some Reflections on Designing ConstructionKits for Kids / M. Resnick, B. Silverman // IDC'05 Proceedings of the 2005 conference on Interaction Design and Children. – 2005. – 118 p. 18. Resnick M. A Networked, Media-Rich Programming Environment to Enhance Technological Fluency at After-School Centers in Economically-Disadvantaged Communities [Electronic Resource] // Режим доступу: web.media.mit.edu/~mres/papers/scratch-proposal.pdf. – Last access: 2003. – Title from the screen. 19. Maloney J. Scratch: A Sneak Preview / J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, M. Resnick // C5 '04 Proceedings of the Second International Conference on Creating, Connecting, and Collaborating through Computing. – 2004. – 107 p. 20. Myers B. Taxonomies of visual programming and program visualization / B. Myers // Visual Languages and Computing. – 1990. – № 1. – 102 p. 21. Clarisse O. VICON: A Visual Icon Manager /

O. Clarisse, S. Chang // Visual Languages. – 1986. – 152 p. 22. Fowler M. ProjectionalEditing [Electronic Resource] // Режим доступу: http://martinfowler.com/bliki/ProjectionalEditing.html. – Last access: 2008. – Title from the screen. 23. Donzeau-Gouge V. Programming environments based on structured editors: The Mentor experience / V. Donzeau-Gouge, G. Huet, G. Kahn, B. Lang // INRIA Research report. – 1980. – № 26. – 13 p. 24. Shneiderman, B. Direct Manipulation. A Step Beyond Programming Languages / B. Shneiderman // IEEE Computer 16. – 1982. – № 8. – 57 p. 25. Bederson, B. B., Grosjean, J., & Meyer, J. (2004). Toolkit Design for Interactive Structured Graphics, IEEE Transactions on Software Engineering, 30 (8), pp. 535-546. 26. Maloney J. Scratch: A Sneak Preview / J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, M. Resnick // C5'04 Proceedings of the Second International Conference on Creating, Connecting, and Collaborating through Computing. – 2004. – 106 p. 27. Hodych O.V. Object-relational mapping: Limitations of data querying / O.V. Hodych, N.B. Chaykivskyy, Y.O. Prokopiv, O.L. Maikovych // SAIT-2013 Conference proceedings. – 2013. – 376 p.

УДК 004.652.43+004.652.44

А.Г. Григорович[1], В.Г. Григорович[2]
[1] Дрогобицький державний педагогічний університет ім. Івана Франка,
[2] Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж

ПОДАННЯ РЕЛЯЦІЙНИХ ОПЕРАЦІЙ ЗАСОБАМИ РЕЛЯЦІЙНОГО ЧИСЛЕННЯ ДОМЕНІВ ДЛЯ НЕНОРМАЛІЗОВАНИХ ВІДНОШЕНЬ

Запропоновано вирази реляційного числення доменів для ненормалізованих відношень, еквівалентні операціям розширеної реляційної алгебри.

Ключові слова: ненормалізовані відношення, реляційне числення доменів, реляційні операції, реляційна алгебра.

This article presents expressions of relational calculus of domains for nested relations equivalent to extended relational algebra operations.

Key words: nested relations, relational calculus of domains, relational operations, relational algebra.

Вступ

Дослідження, проведені в межах науково-дослідних робіт "Моделювання складних інформаційно-комп'ютерних систем. Розробка та впровадження сучасних інформаційних технологій" (№ державної реєстрації 0109U002838), "Розробка систем підтримки прийняття рішень в складних інформаційно-управляючих комплексах" (№ державної реєстрації 0109U002424) і "Розробка систем підтримки прийняття рішень з управління розвитком складних розподілених техніко-економічних та соціально-економічних систем" (№ державної реєстрації 0111U002287), показали, що існує актуальна наукова задача обробки великих масивів інформації, яка зберігається у сховищах даних. Така інформація є композитною, тобто складною, отримана з різних джерел та у різних форматах. Спроби застосувати до обробки композитних інформаційних об'єктів нормалізовані підходи виявили ряд недоліків: нормалізація призводить до втрати важливих маркетингових даних та зв'язків між сутностями; не дає суттєвих переваг щодо прискорення процесу обробки; в деяких випадках нормалізація заважає використовувати стандартні методи видобування даних (Data Mining), наприклад, пошук асоціативних правил, тому що деякі зв'язки між сутностями втрачено. Одним із засобів вирішення цих проблем є використання ненормалізованих відношень (nested relations) для подання та опрацювання даних композитних об'єктів.