УДК 004.9

Петрашенко А.В., к.т.н.,
Замятін Д.С., к.т.н.,
Михайлюк А.Ю., к.т.н.

ПРОГРАМНІ ЗАСОБИ ЛОГІЧНОЇ ІНТЕГРАЦІЇ РЕСУРСІВ У НЕСТРУКТУРОВАНИХ ТЕКСТОВИХ СХОВИЩАХ

**Національний технічний університет України
«Київський політехнічний інститут»**

Запропоновано підхід до логічного асоціювання ресурсів в неструктурованих сховищах, який базується на механізмі шаблонів документів. Розглянуто спосіб реалізації подання шаблонів у реляційній формі та проведено оцінку швидкодії алгоритмів обробки текстових документів із використанням запропонованого підходу.

Вступ

Відповідно до відомих моделей інформатизації підприємств та організацій, зокрема моделі *IMM* (*Informatization Maturity Model*) [1], процес комп'ютеризації бізнес-процесів починається з впровадження окремих ізольованих автоматизованих систем для підтримки різних сфер діяльності. Кожна з таких систем пропонує власні засоби зберігання, пошуку та систематизації даних, що призводить до появи декількох окремих інформаційних баз в межах одного підприємства. Зростання об'ємів інформації, яка зберігається у таких базах з часом значно знижує ефективність комплексної обробки даних і, відповідно, негативно впливає на продуктивність праці персоналу. Тому задача поєднання окремих електронних ресурсів, дані в яких мають приховані зв'язки, у сховища та засоби обробки даних у сховищах є надзвичайно актуальною.

На сьогодні відомо ряд підходів до інтеграції окремих електронних ресурсів у сховища. Зокрема J. Darmont та ін. [2] розглядають довільну одиницю збереження інформації (документ) як фізичний файл, що безпосередньо містить інформацію з предметної області, та набір атрибутів, які є специфічними до формату збереження даних, що застосовано у даному файлі. Для ряду розповсюджених форматів у роботі запропоновано узагальнене подання на основі *XML*. Нажаль, такий підхід дозволяє зберігати лише технічні характеристики файлів, тобто нехтує семантичними даними з предметної області. *T. Stöhr, R. Müller* та *E. Rahm* [3] розглянули модель інтеграції технічних і семантичних даних. Будь-які сутності з предметної області подаються як сукупність атрибутів та зв'язків між ними. Процес обробки даних подається як послідовний ряд трансформацій, які можуть бути агрегуючими (за багатьма даними один результат) та неагрегуючими. Трансформації застосовуються до даних, які попередньо відфільтровані за вказаними критеріями. Цей підхід є надто узагальненим, що не дозволяє застосувати його до безпосередньої розробки програмного забезпечення для обробки інформації у сховищах. У роботах [4, 5, 6] сховище даних подається як сукупність документів, які відповідно подаються множиною атрибутів та їх значень. Для роботи із сховищами виділено ряд уніфікованих формально визначених операцій, на основі комбінації яких виникає можливість реалізовувати програмні компоненти для обраної предметної області. Даний підхід при практичній реалізації має важливе обмеження: часто певні набори документів мають спільні алгоритми обробки, тому доцільним є введення поняття типу документу. Інформація про тип документа може бути збережена у

визначеному атрибуті, але, в такому разі, цей атрибут буде вимагати окремих механізмів обробки, що призводитиме до ускладнення програмного коду реалізації. Крім того, тип документа може визначатися значеннями декількох атрибутів, що ще більше ускладнює розробку. У роботі [7] ця проблема показана на прикладі організації підсистеми розмежування доступу у сховищі документів.

В даній роботі пропонується спосіб визначення наборів документів, які обробляються за специфічними алгоритмами, лише на основі значень атрибутів, що дозволяє створювати системи, поведінка яких визначається вхідними даними.

Постановка задачі

Нехай, сховище W складається з множини документів D_i, $i = 1...n$, A — множина всіх можливих атрибутів документів $D_i \in W$, тоді документ D_i можна подати як кортеж, що складається з множини атрибутів $A_j \in A$ документа та їх значень V_j:

$$D_i = <A_j : V_j>.$$

Таким чином, неструктуроване текстове сховище можна подати як множину документів, кожний з яких складається з множини значень певних атрибутів [6].

Досить типовою є ситуація, коли певна підмножина документів сховища W вимагає обробки за допомогою певних специфічних алгоритмів. Прикладами таких ситуацій можуть бути наступні задачі:

–формування списку атрибутів для документа, який буде створено;

–визначення порядку атрибутів та їх розташування на екрані при візуалізації документів, які вже існують;

–розрахунок значень агрегатних функцій за набором документів, які мають вказаний набір атрибутів (наявність документів без вказаних атрибутів буде призводити до помилкових значень розрахунків, наприклад, середнього значення);

–визначення набору документів, які може переглядати чи модифікувати певний користувач (задача розмежування доступу);

–тощо.

Зазвичай, групування документів за принципом спільної поведінки виконується на основі поняття типів документів, які безпосередньо проектуються на класи сутностей з предметної області. Віднесення документа до певного типу можливо шляхом введення додаткової складової до його визначення. Наприклад, якщо T_k — ідентифікатор типу k, то документ може подаватися множиною:

$$D_i = \{T_k, <A_j : V_j>\}.$$

Нажаль, таке рішення значно ускладнює практичну реалізацію програмних систем на його основі. Обробка окремої складової визначення вимагатиме додаткової розробки програмного коду для пошуку, каталогізації, впорядковування і т. ін. за її значенням, оскільки наявні засоби, що працюють зі значеннями атрибутів для цього непридатні.

Інший варіант задання типу документу можливий на основі створення окремого спеціального атрибуту $A_n \in A$. Оскільки він містить метаінформацію щодо документа, назвемо його метаатрибутом. Цей метаатрибут не буде відрізнятися за властивостями від інших, тому всі функції роботи з атрибутами будуть з ним працювати. Але, в більшості випадків, для обробки такого метаатрибуту у стандартних алгоритмах необхідно реалізувати специфічну поведінку для атрибуту, номер якого співпадає з n, щоб відрізняти його від суттєвих атрибутів. Якщо в подальшому виникне необхідність групувати документи за підтипами, типи будуть описуватись вже множиною метаатрибутів $A' \supseteq A$, для кожного з яких необхідно передбачити особливу поведінку. Це вимагатиме переробки

програмного коду, що є небажаним, оскільки фактично призводить до реалізації нової системи.

Таким чином, умовою розробки експлуатаційно придатної системи підтримки сховища документів є наявність механізмів групування документів за типами.

Шаблони документів

Вище було показано, що введення метаатрибутів з певними специфічними властивостями, наприклад, атрибутів типу призводить до необґрунтованого ускладнення моделі і, як наслідок, ускладнення архітектури програмних засобів, побудованих на їх основі. При цьому, значення суттєвих атрибутів документу залежать від його класу у предметній області, і, тому, здатні визначити його тип. Якщо набору існуючих атрибутів не достатньо для того, щоб відрізнити один тип документу від іншого, можна ввести додатковий атрибут, який відображатиме те, що в предметній області розглянуті документи відрізняються за типами, і, тому, буде суттєвим, а не метаатрибутом. Будемо вважати, що тип документа визначається значеннями підмножини атрибутів $\{A'_1, A'_2, ..., A'_n\} \in A$. Нехай V^A — це значення атрибуту A, V — певна константа, а характеристична булева функція $f(V^A, V)$ повертає істинне значення, якщо V^A та V знаходяться у відповідному відношенні. Наприклад, якщо значення атрибуту A відносяться до числового типу, функція f може бути задана як

$$f(V^A, V) = V^A \theta V, \qquad (1)$$

де $\theta \in \{=, \neq, >, <, \leq, \geq\}$. Оскільки тип документа визначається значеннями його атрибутів, то приналежність документа до заданого типу визначатиметься булевою функцією $F(f_1, f_2, ..., f_n)$, де $f_1, f_2, ..., f_n$ — відповідні характеристичні функції для атрибутів $A'_1, A'_2, ..., A'_n$, яку ми назвемо *шаблоном*. Відомо [8], що будь-яку булеву функцію можна подати у вигляді диз'юнктивної нормальної форми (ДНФ), тобто у вигляді диз'юнкції кон'юнкцій вхідних параметрів, кожна з яких визначає певну конституенту одиниці. Тобто, шаблон типу документа F можна задати як

$$F(f_1, f_2, ..., f_n) = \vee_{i=1..k}(\wedge_{j=1..n} f_j), \qquad (2)$$

де k — кількість конституент одиниці. Для оптимізації розрахунку приналежності документу шаблону функція F може бути мінімізована на основі відомих методів [9].

Крім визначення приналежності документа заданому типу, практичну цінність має задача знаходження набору документів, що належить даному типу. Якщо за визначення операції фільтрації з [4] переписати у вигляді:

$$W.where(f) = \{D \mid D \in W \wedge \exists A \in D \wedge f(V^A, V) = True\}, \qquad (3)$$

де W — сховище документів;
D — документ зі сховища;
f — характеристична функція, то вона буде повертати нове сховище, що містить документи, які відповідають цій функції. Також для сховищ в [6] визначені теоретико-множинні операції об'єднання та перетину:

$$W_1 \cup W_2 = \{D \mid D \in W_1 \vee D \in W_2\}$$

та

$$W_1 \cap W_2 = \{D \mid D \in W_1 \wedge D \in W_2\}$$

відповідно. З (2) та (3) очевидно, що

$$\{D \mid D \in W \wedge (\exists A_1 \in D \wedge f_1(V^A_1, V_1) = True) \vee (\exists A_2 \in D \wedge f_2(V^A_2, V_2) = True)\} = \{D \mid D \in W.where(f_1) \vee D \in W.where(f_2)\} = W.where(f_1) \cup W.where(f_2)$$

Тобто диз'юнкція характеристичних функцій відповідає об'єднанню сховищ, аналогічно теж саме можна довести і для кон'юнкції та перстину. Звідси можна запропонувати правило знаходження нового сховища документів, що відповідає заданому типу T. Для заданого шаблону необхідно виконати наступні дії:

1. Для всіх характеристичних функцій f_n отримати відфільтровані сховища $W_n = W.where(f_n)$.

2. Для всіх диз'юнкцій з шаблону отримати об'єднання відповідних відфільтрованих сховищ.

3. Обчислити перетин всіх об'єднань, що отримані на кроці 2.

Сховище, яке буде отримано на кроці 3 буде набором документів, що відповідає заданому типу. На рис. 1 зображено схему алгоритму для отримання набору документів за певним шаблоном.

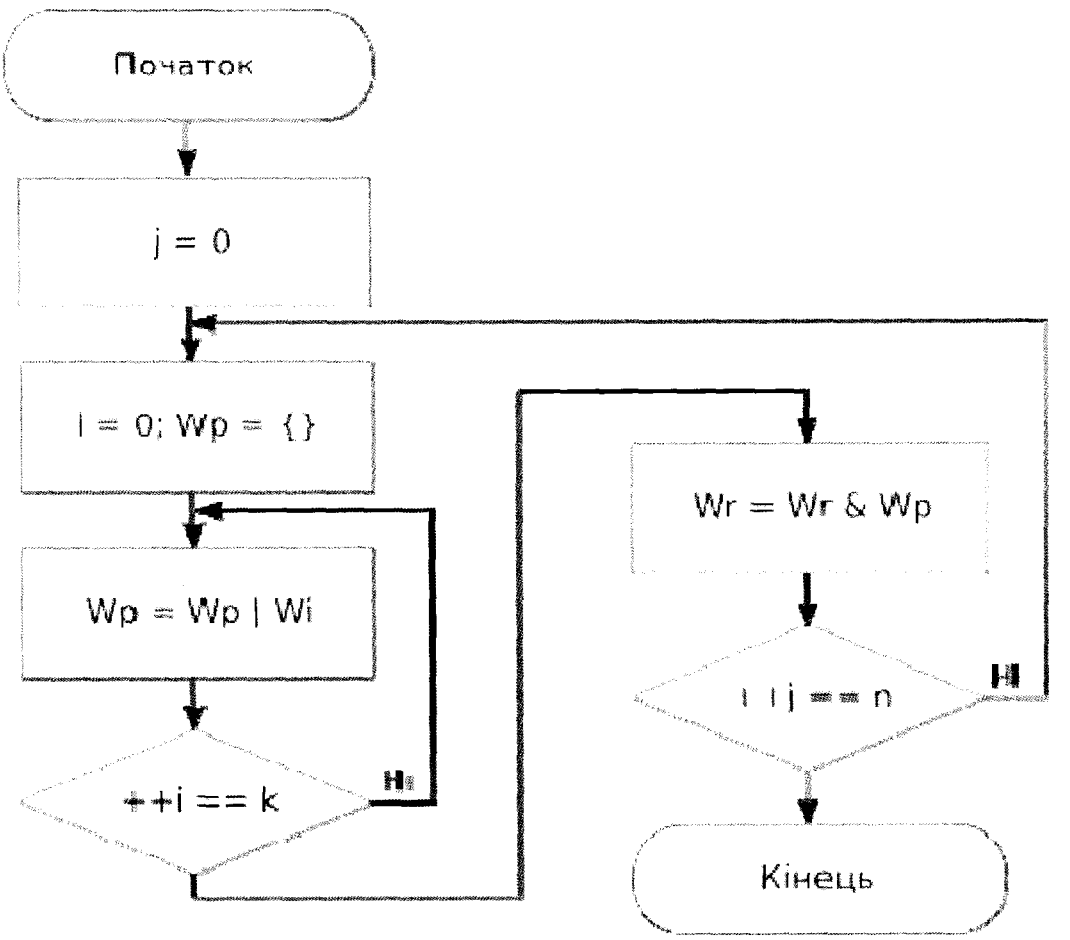

Рис. 1. Схема алгоритму отримання набору документів, що відповідають заданому шаблону (N — кількість документів у сховищі)

Шаблони у реляційному поданні

На сьогодні фактичним стандартом розробки програмних засобів обробки документів у сховищах стало застосування трирівневої архітектури, в якій в основі шару збереження інформації використовується реляційна система управління базами даних [10]. Тому для уніфікації інтерфейсів доступу до структур даних, які використовуються програмними засобами, доцільно розробити ефективне подання шаблонів документів у вигляді таблиці реляційної бази даних (БД). Відповідно до (2), шаблон документів подається включає в себе декілька груп кон'юнкцій значень характеристичних функцій для даного документа, які об'єднуються за допомогою диз'юнкції. Оскільки останній крок є однаковим для будь-якого шаблону, збереження інформації про нього у БД є надлишковою. Тому таблиця шаблонів повинна містити опис характеристичних функцій та надавати можливість групувати їх у групи кон'юнкцій.

Визначення характеристичної функції, надане у (1), можна також поширити на рядкові дані та дані типу дата шляхом введення для них функцій порівняння на основі впорядкованості у лексографічному або у хронологічному порядку відповідно. Крім розглянутих типів даних, а саме число, рядок та дата, значенням атрибуту документу може бути також бінарний об'єкт, наприклад зображення, звукозапис тощо, але використання таких атрибутів для визначення типу документу є недоцільним через можливу неоднозначність при інтерпретації бінарних об'єктів. Таким чином, можна вважати набір операторів $=, \neq, >, <, \leq, \geq$ достатнім для порівняння значень атрибутів всіх типів даних, які можуть бути використані у характеристичних функціях. Характеристична функція надає можливість порівняти значення певного атрибуту з заданою константою, тому для її визначення, крім оператора, також потрібно вказати ідентифікатор атрибута та константу.

Отже, характеристична функція може бути подана як кортеж (f, A, θ, V), де f – ідентифікатор характеристичної функцій, A – ідентифікатор атрибуту, θ – оператор з наведеної вище множини, V – константа, з якою відбувається порівняння. Кожний елемент кортежу може бути безпосередньо поданий як відповідне поле у таблиці реляційної БД. Треба зауважити, що подання булевої функції у ДНФ передбачає її опис у базисі (І, АБО, НЕ), тобто для забезпечення функціональної повноти, необхідно забезпечити можливість отримання інвертованого значення характеристичної функції $f(V^A, V)$. Але оскільки множина $\{=, \neq, >, <, \leq, \geq\}$ складається з трьох пар протилежних операторів, інвертування функції можна реалізувати шляхом вибору необхідного оператору.

Наступним питанням опису шаблону у реляційній формі є поєднання характеристичних функцій у групи кон'юнкцій. Для того, щоб ідентифікувати певну групу можна перенумерувати всі групи в межах одного шаблону. Тоді приналежність функції групі кон'юнкцій буде визначатися номером номером цієї групи. Таким чином, шаблон документів можна описати як кортеж $<S, G, f>$, де S — ідентифікатор шаблону, G - ідентифікатор групи у шаблоні, f - ідентифікатор характеристичної функції. На основі запропонованого подання шаблону можна побудувати структуру сегменту БД, який відповідає за збереження документів та їх шаблонів. На рис. 2 наведено таку структуру у вигляді діаграми класів *UML*. Фрагмент БД, який призначено для збереження атрибутів документів та їх значень побудовано на основі моделі *Entity-Attribute-Value* [11], що надає можливість зберігати документи довільної структури.

З наведеної схеми сегмента БД видно, що механізм шаблонів не вимагає внесення змін у сегмент БД, в якому зберігаються документи, тобто є надбудовою над даними і, тому, не впливає на реалізацію функцій бізнес-логіки, які не пов'язані з типами документів.

Параметри реалізації механізму шаблонів

Інформація про тип документа, яка надається механізмом шаблонів, може бути використана у випадках, коли необхідно задати специфічну поведінку для стандартних функцій. Значну частину таких функцій складають засоби, які повертають інформацію користувачу у реальному часі. Для таких випадків, особливо при значних об'ємах сховища, швидкодія алгоритмів, які реалізують механізм шаблонів стає критичною. Розглянемо це питання детальніше.

Основу механізму шаблонів складають характеристичні функції, реалізація яких полягає у застосуванні функції фільтрації до всього масиву документів у сховищі. Фільтрація документів за значенням певного атрибуту вимагає проходу по всім документам, тобто, при кількості

документів у сховищі N, складність одноразового застосування функції складатиме $\Theta(N)$. Якщо у визначенні шаблону використано M характеристичних функцій, складність зростає до $\Theta(M \times N)$. При великих значеннях N обчислення всіх потрібних функцій може тривати занадто довгий час, щоби забезпечити достатній час відклику користувацького інтерфейсу.

Збільшити швидкодію операції визначення значень характеристичних функцій можна за рахунок їх попереднього обчислення. Оскільки кожна функція повертає значення з множини $\{True, False\}$, результат її застосування до всього масиву документів можна подати як бітовий рядок, де кожний біт відповідає певному документу, тобто використати так звані бітові індекси [12].

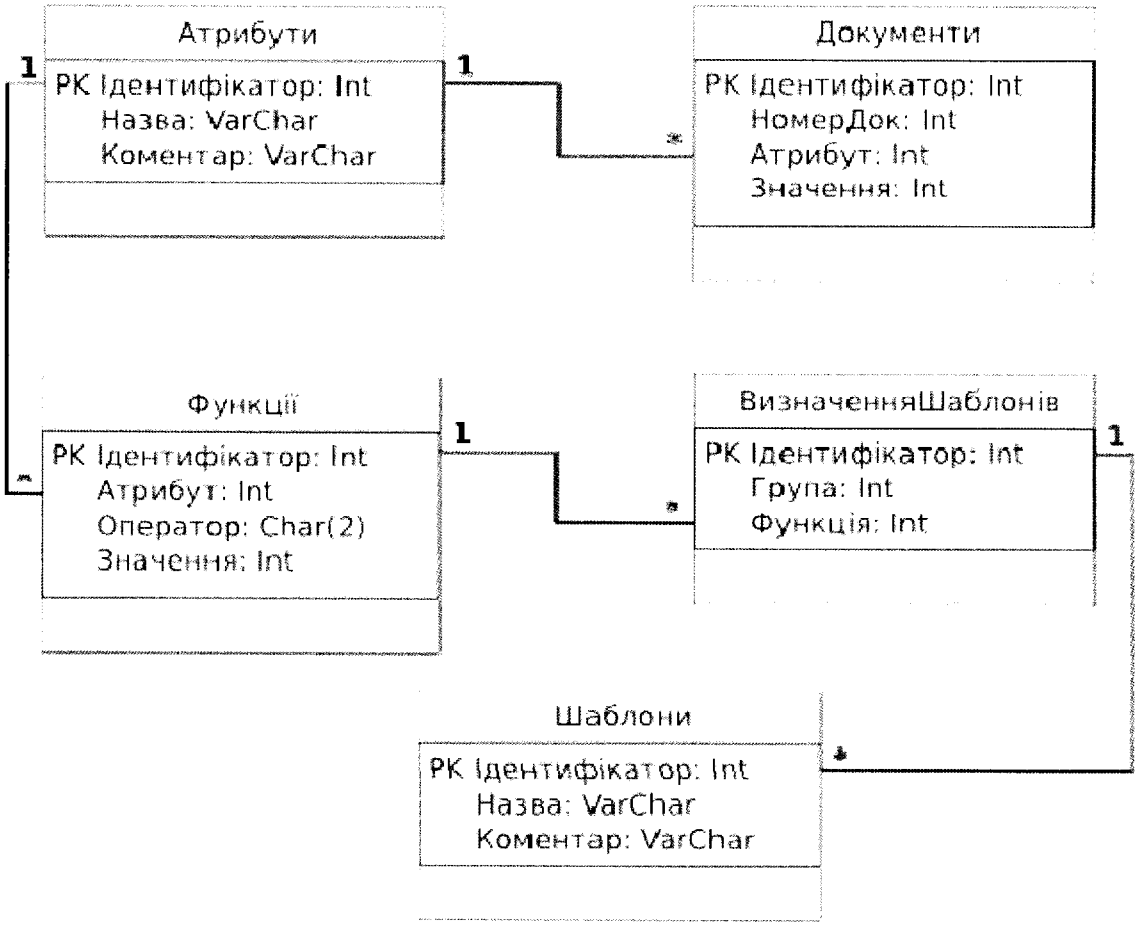

Рис. 2. UML-діаграма структури сегмента БД, який відповідає за зберсження шаблонів

Для оцінки об'єму пам'яті, яка потрібна для збереження набору бітових індексів характеристичних функцій будемо вважати, що кількість документів у сховищі складає 1000000 (на сьогодні кількість книжок у бібліотеці Конгресу США, яка вважається найбільшою у світі, близько 140 млн. [13], тобто запропонована оцінка є більш ніж оптимістичною). Тоді, на збереження індексу однієї функції витрачатиметься 125000 байтів. Враховуючи те, що один індекс може бути використаний як для самої функції, так і для її інвертованого значення, можна вважати, що на одну функцію потрібно вдвічі менше місця, тобто близько 60 кБ. Оскільки об'єм оперативної пам'яті сучасних обчислювальних систем вираховується гігабайтами, використання бітових

індексів суттєво не вплине на вимоги до апаратної платформи, для якої створюється реалізація відповідних програмних засобів.

Таким чином, використавши бітові індекси, час визначення значень характеристичних функцій можна звести до константного, а визначення масиву документів заданого типу буде виконуватись як диз'юнкція набору кон'юнкцій бітових масивів.

Додаткове збільшення швидкодії розрахунку характеристичних функцій можна отримати за допомогою розпаралелення виконання обчислення. Відповідно до алгоритму, який зображено на рис. 1, визначення приналежності кожного окремого документу заданому шаблону не залежить від приналежності інших документів. Це означає, що розрахунок приналежності для заданого документу можна проводити паралельно з іншими.

Оскільки, при застосуванні бітових індексів, за значення кожної окремої характеристичної функції відповідає 1 біт, швидкодія розрахунку буде залежати від розрядності машинного слова. При кількості документів у сховищі N, якщо кількість диз'юнкцій у шаблоні становить K_D, а кон'юнкцій K_K, кількість кроків, необхідних для розрахунку бітового подання одного масиву складатиме:

$$\sum_{i=1}^{K_D+K_K} \frac{N}{R} = \frac{1}{R} \sum_{i=1}^{K_D+K_K} N,$$

де R — кількість розрядів машинного слова. Зокрема, при застосуванні процесорів з 64-бітною архітектурою, кількість операцій, потрібних для розрахунку буде $N/64$, а при використанні векторних розширень SSE2, які дозволяють виконувати бітові операції над 128-бітними числами — $N/128$.

Останнім часом, значне поширення отримали процесори з багатоядерною архітектурою, використання яких дозволяє зменшити кількість потрібних для розрахунку кроків ще у P разів, де P — кількість ядер у процесорі:

$$\frac{1}{RP} \sum_{i=1}^{K_D+K_K} N.$$

При значних обсягах обчислень за шаблонами, для зменшення навантаження на центральний процесор, можуть також бути застосовані гетерогенні системи, наприклад з використанням технологій *GPGPU* на основі *NVidia CUDA* або *OpenCL*. В такому випадку, крім часу, який витрачатиметься безпосередньо на розрахунок, потрібно також врахувати час на перенесення результату з пам'яті графічного пристрою, який в загальному випадку, є пропорційним N.

Час на завантаження бітових індексів в пам'ять пристрою можна не враховувати, оскільки ця операція виконується одноразово перед початком обчислень. У випадку застосування технологій *GPGPU*, кількість операцій для розрахунку складатиме:

$$\frac{1}{RP} \sum_{i=1}^{K_D+K_K} N + \Theta(N).$$

В цьому випадку слід враховувати, що значення P значно більше, ніж при застосуванні багатоядерних процесорів.

Застосування механізму шаблонів у інформаційно-аналітичних системах

Механізм шаблонів, розглянутий у даній роботі, було впроваджено при розробці інформаційно-аналітичної системи, призначеної для аналізу та систематизації сховища текстовміщуючих ресурсів Київського університету ім. Б. Грінченка. Система розроблялася як *web*-орієнтована з використання технології *AJAX* на основі бібліотеки *Jquery*. Для серверної підсистеми було використано мову програмування *Python* та фреймворк *Django*. Для створення та редагування шаблонів було розроблено спеціалізовані візуальні засоби, які за допомогою контекстних підказок дозволяють зменшити ймовірність технічних помилок при введені значень атрибутів та гарантують цілісність даних

у шаблонах. Приклад створення шаблону за допомогою розглянутих засобів наведено на рис. 3.

На основі механізму шаблонів в системі реалізовано наступні функції:

–визначення дозволених режимів доступу (перегляд, додавання, редагування, вилучення, модерування) до документів для обраного користувача;

–визначення списку та порядку атрибутів при додаванні нового документа;

–створення візуального подання для документів, що належать обраному шаблону.

Впровадження механізму шаблонів показало його доцільність, перш за все за рахунок можливості ефективної реалізації у вигляді надбудови над ядром системи, тому в подальшому планується інтегрувати цей механізм з іншими функціональними режимами системи.

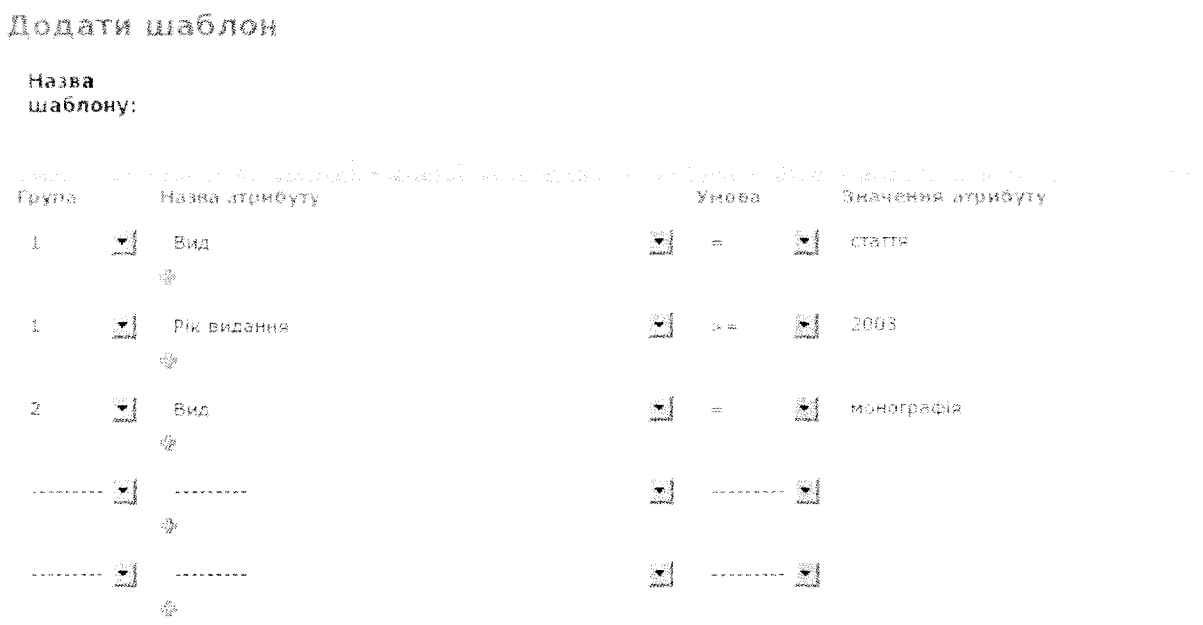

Рис. 3. Інтерфейс користувача режиму створення шаблонів

Висновки

Розглянуто задачу групування документів у сховищах для подальшого застосування спільних алгоритмів обробки та запропоновано спосіб її вирішення, оснований на механізмі шаблонів. Механізм шаблонів документів базується на використанні значень атрибутів, що дозволяє при його реалізації не вводити додаткових спеціалізованих атрибутів, тобто не змінювати модель подання документів в системі. Шаблон визначається як булева функція, яка в якості аргументів отримує значення характеристичних функцій для окремих документів та подається у вигляді диз'юнктивної нормальної форми. Також розроблено алгоритм для обчислення сукупності документів, які належать заданому шаблону.

Для реалізації механізму шаблонів у вигляді програмної компоненти, запропоновано спосіб подання шаблонів у реляційних базах даних та способи підвищення швидкодії алгоритму розрахунку сукупності документів на основі бітових індексів та розпаралелення з використанням багатоядерних процесорів та технологій *GPGPU*. На прикладі впровадження механізму шаблонів у систему обробки корпоративного сховища документів показано його ефективність.

Список літератури

1. M. Zuo, H. Fu. Enterprise – Information Maturity Model Based on Delphi Method // Research and Practical Issues of Enterprise Information Systems II. Vol. 2. – Springer. – 2007. – P. 1117-1126.

2. S. Miniaoui, J. Darmont, O. Boussaid, Web data modeling for integration in data warehouses / First International Workshop on Multimedia Data and Document Engineering (MDDE 01). – Lyon, France. – 2001. – P. 88-97.

3. T. Stöhr, R. Müller, E. Rahm An Integrative and Uniform Model for Metadata Management in Data Warehousing Environments / International Workshop on Design and Management of Data Warehouses (DMDW'99 at CAiSE99). – Heidelberg, Germany. – 1999. – 16 p.

4. Mykhailyuk A., Zamiatin D., Petrashenko A. Unstructured Data Warehouse Processing System Based on an Uniform Set of Functions // Proceedings of the 4-th International Conference ACSN-2009 "Advanced Computer Systems and Networks: Design and Application". – Lviv. – 2009. – P. 117-119.

5. A. Mykhaylyuk, A. Petrashenko, D. Zamiatin Method of Unstructured Text-Based Warehouses Systematization Based on Composite Attributes / Збірник праць Ювілейної міжнародної науково-практичної конференції, що присвячена 50-річчю створення першої на Україні кафедри обчислювальної техніки РКС-2010 "Розподіленні комп'ютерні системи". Т. 1. – Київ. – 2010. – С. 86-88.

6. Михайлюк А.Ю., Замятін Д.С., Петрашенко А.В. Проблеми систематизації даних у неструктурованих текстових сховищах / Вісник університету "Україна". Серія "Інформатика, обчислювальна техніка та кібернетика". – №8. – Київ. – 2010. – С. 88-91.

7. Петрашенко А.В., Замятин Д.С. Методы разграничения доступа в неструктурированных текстовых хранилищах / Тези доповідей Міжнародної науково-практичної конференції "Інформаційні технології та комп'ютерна інженерія". – Вінниця. – 2010. – С. 217-218.

8. Капітонова Ю.В. Основи дискретної математики / Капітонова Ю.В., Кривий С.Л., Летичевський О.А. – К.: Наукова думка. – 2002. – С. 96-107.

9. Самофалов К. Г. Прикладная теория цифровых автоматов / Самофалов К. Г., Романкевич А. М., Валуйский В. Н., Каневский Ю. С., Пиневич М. М. – К.: Вища школа. – 1987. – 375 с.

10. Kadav A., Kawale J., Mitra P. Data Mining Standards, 2008. – 33 p. [Електронний ресурс]. – Режим доступу: http://www.datamininggrid.org/wdat/works/att/standard01.content.08439.pdf

11. McDonald, C.J., Blevins, L., Tierney, W.M., Martin, D.K. The Regenstrief Medical Records / MD Computing (5(5)). – 1988. – P. 34-47.

12. O'Neil, P., Quass, D. Improved Query Performance with Variant Indexes // ACM International Conference on Management of Data (SIGMOD 1997). – Tucson, Arizona. – 1997. – 12 p.

13. Year 2009 at a Glance, 2010, 1 p. [Електронний ресурс]. – Режим доступу: http://www.loc.gov/about/generalinfo.html