УДК 004.4

Михайлюк А.Ю.
кандидат технічних наук
Замятін Д.С.
кандидат технічних наук
Петрашенко А.В.
кандидат технічних наук

# ПРОБЛЕМИ СИСТЕМАТИЗАЦІЇ ДАНИХ У НЕСТРУКТУРОВАНИХ ТЕКСТОВИХ СХОВИЩАХ

Активне використання глобальних інформаційних мереж призводить до неконтрольованого зростання обсягів неструктурованих ресурсів, представлених, зокрема, в текстовому вигляді. Через відсутність формальних засобів визначення атрибутів значно ускладнюється процедура систематизації інформації (каталогізації, пошуку тощо).

Сучасні методи аналізу та каталогізації текстових ресурсів, що базуються на теоретичному апараті неструктурованих сховищ даних, розроблених, зокрема, W. Inmon та іншими [1, с.83-90; 2, с.58-62], використовують семантикоорієнтовані методи, які мають значну обчислювальну вартість та відносно високу похибку. Альтернативою складним методам можна вважати запропонований в роботі [3, с.117-119] підхід до обробки неструктурованих текстомістких сховищ на основі уніфікованого набору операцій.

## Модель неструктурованого текстового сховища

Для конкретизації задачі, будемо вважати, що в текстовому сховищі, яке ми розглядаємо, можна виділити окремі текстові фрагменти, які в подальшому будемо називати документами. Тобто сховище $W$ складається з множини документів $D_i$, $i = 1...n$. Властивість неструктурованості можна інтерпретувати як відсутність у документів спільних характеристик, таких, як, наприклад, тип документу, за якими їх можна було б формально поділити

на певні групи. Тому, в якості джерела інформації для структурування можуть виступати лише значення атрибутів документа, включаючи його безпосередній текст. Нехай $A$ — множина всіх можливих атрибутів документів $D_i \in W$, тоді документ $D_i$ можна подати як множину кортежів, кожний з яких складається з атрибуту $A_j \in A$ документа та його значення $V_j$:

$$D_i = \{<A_j : V_j>\}$$

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

**Операції над сховищами**

Розробка систем підтримки сховищ неструктурованих даних тісно пов'язана з предметною галуззю, який присвячені документи, що містяться у сховищі. Це означає, що кожна предметна галузь вимагає розробки спеціалізованих засобів, які будуть реалізовувати відповідні специфічні алгоритми обробки. В наведеній роботі запропоновано ввести набір уніфікованих операцій, результатом яких повинні бути нові сховища, що має спростити формальний опис відповідних алгоритмів. Реалізація подібних операцій у вигляді попередньо відтестованих бібліотечних модулів дозволить спростити та прискорити розробку систем обробки неструктурованих даних.

На даному етапі в набір таких операцій пропонується включити базові теоретико-множинні дії та додаткову операцію вибірки.

Серед теоретико-множинних операцій доцільно використати:

а) об'єднання

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

б) перетин

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

в) різницю

$$W_1 - W_2 = \{D \mid D \in W_1 \wedge D \notin W_2\}.$$

Операція вибірки в якості аргументу отримує значення певної характеристичної булевої функції *f*, яка послідовно застосовується до всіх документів сховища, та повертає нове сховище, що містить лише ті документи, для яких значення функції істинне:

$$W.where(f) = \{D \mid D \in W \wedge f(D) = True\}.$$

Крім операцій, які повертають сховища в якості результату, для отримання інформації про значення атрибутів доцільно включити функцію, яка визначає множину наявних значень атрибутів:

$$W.values(A_j) = \{V_j \mid \exists A_j{:}V_j \wedge D \in W \wedge A_j \in D\}.$$

**Операції над значеннями атрибутів**

В якості характеристичної функції *f* повинна виступати булева функція, яка певним чином аналізує значення атрибутів документа. Для багатьох предметних областей доцільно буде використати наступні функції:

а) функція фільтрації за значенням

$$A_j{:}V_j\ \theta\ const = True, \text{ якщо } A_j \in D_i \wedge V_j\ \theta\ const,$$

де $\theta \in \{=, \neq, >, <, \geq, \leq\}$;

б) функція фільтрації за вмістом

$$const \text{ in } A_j{:}V_j = True, \text{ якщо } A_j \in D_i \wedge V_j \text{ <містить> } const,$$

де під <містить> розуміється входження *const* в якості підрядка.

В залежності від конкретної предметної області можуть бути введені додаткові операції, які враховують відповідні алгоритмічні особливості.

Запропоноване подання текстового сховища разом із введеними операціями дозволяє формально описувати дії, які потрібно виконати над ресурсом, щоб отримати певний результат.

**Реалізації базових режимів обробки документів**

*Атрибутивний пошук*

Задача атрибутивного пошуку зводиться до використання операції вибірки, тобто якщо потрібно знайти всі документи, у яких атрибут “Місто” дорівнює значенню “Київ”, результатом буде:

$$W.where(A(\text{“Місто”}) = \text{“Київ”}).$$

Відповідно пошук за декількома умовами буде визначатися перетином вибірок за кожним з критерієм:

$$W.where(A(\text{“Місто”}) = \text{“Київ”}) \cap W.where(A(\text{“Рік”}) = \text{“2009”})$$

Так само, режим пошуку за приналежністю до діапазону можна виразити за допомогою операції перетину:

$$W.where(A(\text{“Рік”}) > \text{“2005”}) \cap W.where(A(\text{“Рік”}) < \text{“2009”})$$

*Повнотекстовий пошук*

Режим повнотекстового пошуку повертає множину документів, в тексті яких міститься задане слово. Даний режим можна реалізувати з використанням функції фільтрації за вмістом:

$$W.where(\text{“слово” in } A(\text{“Текст”}))$$

Широко використовуються варіації пошуку документів, в які входять усі слова із заданих, будь-яке з них або без заданих слів. Такі комбінації можна виразити відповідними формулами:

$$W.where(\text{“слово1” in } A(\text{“Текст”})) \cap W.where(\text{“слово2” in } A(\text{“Текст”}))$$

$$W.where(\text{“слово1” in } A(\text{“Текст”})) \cup W.where(\text{“слово2” in } A(\text{“Текст”}))$$

$$W - W.where(\text{“слово” in } A(\text{“Текст”})).$$

*Каталогізація*

Побудова каталогу документів у загальному випадку включає вирішення двох задач: побудову дерева каталогу та визначення набору документів, які відносяться до вибраної гілки. Побудова дерева зводиться до визначення всіх наявних значень атрибуту на певному рівні ієрархії каталогу. Наприклад, для документів з атрибутом $A(\text{“Рік”}) = \text{“2009”}$ та $A(\text{“Місто”}) = \text{“Київ”}$ обчислення списку гілок за атрибутом $A(\text{“Автор”})$ виглядатиме так:

$$(W.where(A(\text{“Місто”}) = \text{“Київ”}) \cap W.where(A(\text{“Рік”}) = \text{“2009”})).values(A(\text{“Автор”}))$$

Інша задача — знаходження набору документів, які відносяться до обраної гілки, зводиться до фільтрації:

$$W.where(A(\text{“Місто”}) = \text{“Київ”}) \cap W.where(A(\text{“Рік”}) = \text{“2009”}$$
$$\cap\ W.where(A(\text{“Автор”}) = \text{“Шевченко”})$$

*Пошук дублікатів*

Для реалізації пошуку документів подібних до заданого часто застосовуються методи із застосуванням хеш-функцій. Розглянемо сховище

$$W_1 = W.where(A(\text{“Hash”}) = V_1),$$

де $V_1$ — перший елемент множини $V = W.values(A(\text{“Hash”}))$, *Hash* — атрибут, значення якого відповідає певній хеш-функції, розрахованої за текстом документа $D_1$.

Тоді якщо $W_1 \neq \varnothing$ — знайдено частковий збіг $D_1$ з якимось документом із $W$. В залежності від розміру $D_1$ та налаштувань алгоритму (зокрема, кількості слів в ланцюжку, на основі якого будується контрольна сума), на цій підставі можна стверджувати про копіювання частини тексту.

Проте, якщо характер $D_1$ передбачає широке використання цитат і механізми їхнього видалення не враховані в процедурі нормалізації тексту, то такий підхід може виявитися неефективним. В цьому разі прийнятним критерієм подібності виступатиме розмір сховища

$$W' = W_1 \cap W_2 \cap ... \cap W_n \mid \forall W \in \{W_1, W_2, ..., W_n\}, W \neq \varnothing,$$

де $W_2, ..., W_n$ будуються аналогічно до $W_1$.

Таким чином, показано можливість реалізації традиційних методів обробки текстових ресурсів за допомогою уніфікованого набору операцій.

**Систематизація неструктурованих даних**

Систематизація неструктурованих текстових ресурсів ускладнюється відсутністю єдиної форми подання загальновживаних атрибутів, зокрема персональних даних, дат та їх діапазонів, назв установ, топонімів і т.ін. Наприклад, декілька статей, написаних одним автором можуть бути підписані у різний спосіб: “Шевченко Тарас Григорович, “Т. Шевченко”,

“Шевченко Т.Г.” Існуючі інформаційно-пошукові засоби не здатні зв'язати наведені варіанти запису з одним автором.

Будемо вважати атрибут *композитним* $A_j \in A$ , якщо його значення $V_{ij}$ для документу $D_i \in W$ можна подати як певну функцію від інших значень цього атрибута даного документу:

$$V_{ij} = f(V_{in}, V_{in+1},...,V_{in+k})$$

Розрахунок значення $V_{ij}$ за функцією $f$, яка попередньо визначена на основі властивостей предметної області, дозволить зв'язати документ $D_i$ з іншими документами, для яких значення атрибуту $A_j$ співпадає з $V_{ij}$, навіть якщо у документа $D_i$ атрибут $A_j$ відсутній. Для кожного композитного атрибуту можуть існувати декілька функцій. Подібне співставлення надає можливість знайти приховані зв'язки між документами, отриманими з різних джерел та з різним поданням значень атрибутів.

В залежності від типів даних атрибутів, функція $f$ може бути формально записана як послідовність арифметичних операцій, операцій з рядками та перетворень типів (табл. 1).

Таблиця 1

Операції над значеннями атрибутів в залежності від типу даних

| | Рядок | Число | Дата |
|---|---|---|---|
| Рядок | Конкатенація, виділення підрядка, транслітерація та детранслітерація | Перетворення типу | Перетворення типу за шаблоном |
| Число | Перетворення типу | +, -, *, /, інші операції та функції | Перетворення типу за шаблоном |
| Дата | Перетворення типу за шаблоном та виділення елементів дати | Виділення елементів дати | Знаходження різниці між датами |

Співставлення документів з використанням композитних атрибутів, наприклад, у режимі каталогізації, може відбуватись за наступним алгоритмом. Першим кроком виділяються документи для яких заданий атрибут не є композитним. В термінах уніфікованих операцій множина гілок каталогу визначатиметься рівністю:

$$\{W.where(A' = V) \mid \forall\ V \in W.values(A)\},$$

де $W$ — сховище документів, $A'$ — значення певного атрибуту $A$, *where* — операція вибірки документів за значенням атрибутів, *values* — операція знаходження множини всіх наявних значень атрибутів. Після групування таких документів у окремі гілки, шукаються документи, у яких наявні атрибути, що використовуються у кожній функції $f_i$ визначення композитних атрибутів:

$$W' = \{\cap W.where(exists(A)) \mid \forall\ A \in Arg(f_i)\},$$

де $Arg(f_i)$ — множина аргументів функції $f_i$, $exists(A)$ — функція, яка визначає наявність атрибута $A$ в певному документі. Для знайдених документів розраховуються значення атрибутів, на основі яких відбувається подальше формування гілок каталогу. Тобто множина документів, які відповідають гілці каталогу із значенням $V$ складатиметься з:

$$W.where(A' = V) \cup \{\cup W'.where(f_i = V) \mid \forall\ f_i\}$$

**Застосування в сучасних реляційних системах управління базами даних**

Запропонована модель подання сховища документів у вигляді множини кортежів атрибутів передбачає ефективну реалізацію на основі використання підходу “Сутність-Атрибут-Значення” [4, с.34-47], яка дозволяє адаптувати сховище даних до довільної структури наявних документів без модифікації існуючої схеми бази даних, що зменшує вартість підтримки цілісності. В такому випадку сховище подається у вигляді двох пов'язаних між собою таблиць, в яких містяться атрибути та їх значення відповідно.

Розроблений набір уніфікованих операцій має прямі аналогії з операціями реляційної алгебри [5, с.192-205], що дозволяє створити ефективну реалізацію програмних засобів обробки неструктурованих текстових ресурсів на основі сучасних систем управління базами даних.

**Висновки**

В роботі розглянуто проблеми сучасних текстових неструктурованих сховищ даних, запропоновано підходи щодо формального опису бізнес-процесів обробки неструктурованих сховищ даних у вигляді набору уніфікованих операцій та функцій, а також створити ефективну їх реалізацію на основі сучасних систем управління базами даних. Запропонований апарат композитних атрибутів дозволяє проводити додаткову систематизацію ресурсів шляхом пошуку прихованих зв’язків між документами.

Список літератури

1. W. H. Inmon, A. Nesavich Tapping into Unstructured Data: Integrating Unstructured Data and Textual Analytics into Business Intelligence. — Prentice Hall. — 2007. — 264 p.

2. R. Feldman, J. Sanger The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. — Cambridge University Press. — 2007. — 410 p.

3. 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

4. McDonald, C.J.; Blevins, L.; Tierney, W.M.; Martin, D.K. (1988), "The Regenstrief Medical Records", MD Computing (5(5)): 34-47

5. К.Дж.Дейт Введение в системы баз данных, 8-е издание. – К.; М.; СПб.: Издательский дом «Вильямс», 1999. – 1328 с.