УДК 004.9

Н.Б. Шаховська
Національний університет “Львівська політехніка”,
кафедра інформаційних систем та мереж

МЕТОД ТРАНСФОРМАЦІЇ ОПЕРАЦІЙ АЛГЕБРИЧНОЇ СИСТЕМИ «ПРОСТІР ДАНИХ» В ОПЕРАЦІЇ МОДЕЛЕЙ ДАНИХ ДЖЕРЕЛА

Розроблено метод трансформації запиту користувача простору даних у запит до джерела даних.

Ключові слова: простір даних, алгебраїчна система, база даних, сховище даних.

This Paper concerns method of transformation from dataspace query to data source query.

Key words: dataspace, algebraic system, database, datawarehouse.

Вступ

Глобалізаційні аспекти розвитку сучасного суспільства викликають потребу у побудові складних систем функціонування окремих предметних областей та інформаційних технологій для них. Так, на прикладі університету – це формування рейтингів викладачів та кафедр, визначення показників успішності та якості навчання тощо; на прикладі обласної адміністрації – розрахунок критичних показників розвитку регіону на основі даних, отриманих від організацій різних форм власності. Проте це складно зробити у зв’язку з невідповідністю між вимогами, що ставляться до інформаційних систем, та необхідністю організації (пошуку об’єктів, їх систематизації, узгодження, інтеграції даних) різнотипних інформаційних об’єктів у складну інформаційну систему, що проявляється через:

- слабку структуризацію зв’язків між об’єктами,
- потребу долучення нових об’єктів у систему,
- недотримання загальних стандартів організації ведення документообігу,
- неможливість проведення систематизації через велику кількість об’єктів та їх різну природу.

Аналіз останніх досліджень і постановка задачі

Інформаційним об’єктом, який здатен розв’язати задачі опрацювання інформації з різнотипних джерел, є простір даних (множина різнотипних даних, об’єднаних середовищем керування моделями – визначення Гелеві, Франкліна, Меєра, 2005 р.). У [1–6] введено алгебричну систему «простір даних» DSa=<$\boldsymbol{Ip}$, Ω_P, Ω_F>,де $\boldsymbol{Ip}$ – скінченна множина станів інформаційних продуктів, $\Omega_P = \{O_{P0}, \boldsymbol{O_{Pu}}, \boldsymbol{O_{Pb}}\}$– множина операцій над інформаційними ресурсами, де O_{P0}– нульарна операція, результатом якої є стан заданого ІП у просторі даних; $\boldsymbol{O_{Pu}}$ – множина унарних операцій над простором даних, результатом є зміна стану простору даних; $\boldsymbol{O_{Pb}}$ – множина бінарних операцій над просторами даних, результатом є утворення нового простору даних; Ω_F – множина предикатів, заданих на множині інформаційних продуктів простору даних. Унарними операціями над просторами даних є множина $\boldsymbol{O_{Pu}} = \{Ss, Se, Sm, s_{ac}, f_{Ip}, C, Ag, S\}$, де f_{Ip} – операція визначення структур даних; Ss, Se, Sm – операції пошуку; s_{ac} – операція доступу, C – операція консолідації, Ag – операція агрегації, S – операція трансформації тексту в семантичну мережу.

Проте необхідно розробити формалізми трансформації запитів, що формуються за допомогою введених операцій, в операції, що виконуються на рівні конкретного інформаційного продукту.

Метод комутативного *відображення*, який запропонував Калініченко [7], вимагає опрацювання структурованих джерел (баз даних), що для концепції простору даних є частковим випадком.

Тому *метою* статті є розроблення методу отримання відповіді на запит користувача. *Наукова новизна*: розроблення методу трансформації елементів метамови простору даних у операції на рівні джерела даних. *Практична цінність*: розроблення класу доступу до джерела даних.

Аналіз отриманих наукових результатів

Розроблення алгоритму отримання відповіді на запит користувача простору даних

Подамо алгоритм отримання відповіді на запит користувача до простору даних (рис. 1).

Перед початком інтерактивного сеансу роботи більшість операційних систем вимагає введення ім'я і паролю користувача. Введене ім'я є ідентифікатором користувача, а його пароль – аутентифікатором. Операційна система зазвичай зберігає не сам пароль, а його хеш-суму, забезпечуючи, отже, практичну неможливість відновлення пароля.

Рис. 1. Алгоритм отримання відповіді на запит у просторі даних

Алгоритмічно процедура *авторизації* подається як послідовне передавання одного або декількох потоків даних між суб'єктом і інформаційною системою і їхнє проміжне опрацювання обома сторонами. В результаті цих дій обидві сторони обміну повинні упевнитися, що вони є тими, за кого себе видають.

Задача *візуалізації* – виявлення і подання структур та відношень в наборах даних, що належать до певної предметної області та дозволені користувачу за профілем.

У випадку просторів даних, з наявністю в них структурованих, напівструктурованих та неструктурованих джерел, метою візуалізації є визначення способу відображення даних. Для візуалізації запиту користувача з метою його подальшого виконання використовується дерево. Побудова дерева здійснюється відомими методами формату запису виразу (польський запис). Для візуалізації обрано польський інверсний запис (запис спочатку операндів, а потім операцій).

Процедура *семантичного аналізу* запиту полягає у перевірці коректності задання операндів. Під коректністю задання операндів розумітимемо наявність операнда в одній з множин:

- каталог даних,
- словник даних,
- сформована множина ключових слів заданої предметної області.

У разі невходження у жодну з множин запит вважається невиконаним.

Процедура опрацювання запитів полягає у виконанні:

1) операцій пошуку джерел Ip, що містять відповідь на запит користувача,
2) записування параметрів запиту у тимчасове сховище параметрів запиту A,
3) запуску інтелектуального агента $Agent(Ip)$,
4) якщо джерело даних структуроване, здійснити відбір даних за запитом $Se(s_A(Ip))$,
5) якщо джерело даних напівструктуроване чи неструктуроване, побудувати семантичну мережу $S(Ip)$,
6) консолідації даних $C(Ip)$,

7) запуску модифікованого оператора зрізу для відсікання непотрібних даних $s^{cons}{}_A(cg')$,

8) відображення вмісту отриманого сховища консолідованих даних.

Зазначимо, що користувачі-експерти оминають процедури візуалізації та семантичного аналізу і можуть безпосередньо звертатися до каталогу даних *Cg* чи словника даних *Dic*, а також виконувати етапи процедури опрацювання даних у необхідному для них порядку. Сформовано *кроки виконання операції пошуку* джерел, що містять відповідь на запит користувача:

1. Визначається тип пошуку *TS*: $TS = \begin{cases} Ss,1 \\ Se,2 \\ Sm,3 \end{cases}$
2. Визначаються параметри запиту.
3. За значення *TS* вибираються інформаційні продукти з каталогу *Cg* та визначається тип даних *M*.

$$Ss(Ip \cup S(Ip)) \wedge (M = 'unstruct' \vee M = 'parse' \vee M = 'struct'), Ts = 1$$
$$Se(s(Ip)) \wedge M = 'struct', Ts = 2$$
$$s(Cg) \wedge M = 'struct', Ts = 3$$

4. Отримується доступ до вибраних джерел.
5. Інтерпретація запиту.
6. Виконання запиту.
7. Трансформація результату і повернення його користувачеві.

Розроблення метамови описання джерел даних та встановлення відповідності між їхніми структурами даних

Опишемо елементи метамови простору даних. Вважатимемо, що запит q до простору даних заданий коректно, якщо він складається з операцій імен операцій Ω_P та елементів, описаних у *Cg* та *Dic*.

$$q_{object(c_1 \ldots c_n)} : par = \left\{ \begin{matrix} object \in Cg, P(object) > 0, \\ (c_1, \ldots, c_n) \in Dic \end{matrix} \right\} : par, \quad (1)$$

де *object* – об'єкт, про який йдеться у запиті, $(c_1, \ldots, c_n)$ – назви характеристик об'єкта, *par* – список параметрів запиту.

Залежно від типу джерела параметри можуть відігравати роль: параметрів пошуку – у текстових даних; умови вибору – для структурованих даних.

Абстрактний синтаксис метамови задається як

$$\{R\} \cup \{Rel\} \cup \{K\} \cup \{Hd\}, \quad (2)$$

де описані схеми баз даних, сховищ даних, ключових слів текстових файлів, заголовків веб-документів відповідно (*R*– схеми БД, *Rel* – СД, *K*– ключові слова, *Hd*– заголовки веб-документів).

Семантичний синтаксис метамови складається з елементів, що містяться у словнику синонімів *Dic*. *Схема семантичних функцій* містить операції Ω_P алгебричної системи «простір даних» подана в таблиці.

Схема семантичних функцій простору даних

Операція	Синтаксичний запис	Призначення та опціональні значення аргументів	Приклад запиту
1	2	3	4
$Sm_{object}(par)$	$where(object: [par])$	$object \in Cg$ – назва об'єкта чи характеристики об'єкта, що шукається (*trust* – рівень довіри, *key* – ключове слово пошуку) *par* – характеристика шуканого об'єкта (*place* – місце знаходження, *code* – об'єкт, який семантично зв'язаний з шуканим).	*where* ('замок') – визначення місця знаходження усіх джерел даних, в яких назва таблиці чи атрибута дорівнює «замок»,
	$who(object: [par])$		*who* ('замок') – визначення джерел, в яких назва атрибута дорівнює синонімові значення «замок», отриманого з *Dic*,

Продовження табл.

1	2	3	4
	how(*object*: [*par*])		*how* (0.7: ‘замок’) – пошук джерел, в яких назва характе-ристики або синонімічної назви рівна «замок», а рівень довіри – більший рівний 0.7
Se_c(*object*)	*Se*(*c* : *object*)	*c* – умова пошуку: [*par*]Θ*param*[,[*par*1]Θ*param*1]; ($par \in Cg$ – назва атрибута, зна-чення якого повинно задоволь-няти умову, Θ – символ порів-няння, *param* – значення, з яким необхідно порівняти, $object \in Cg$ – перелік назв об’єк-тів або характеристик об’єктів, що необхідно повернути в запиті.	*Se*(‘замок’,crDate<1400: admunit, name) – визначення назви аміні-стративної одиниці (admunit), та назви об’єкта, для яких: – назва таблиці чи атрибута дорівнює «замок», – назва атрибута дорівнює синонімові значення «замок», отриманого з *Dic*, – дата створення менша 1400.
Ss_c(*object*)	*what*(*object*)	$object \in Cg$ – назва об’єкта чи характеристики об’єкта, що шукається . *c* – значення об’єкта, що шукається (опціонально) .	*what* (‘замок’) – відкриття усіх текстових файлів, в яких знайдено «замок».
	which(*object* [: *c*])		*which* (‘замок’:3) – вибір з текс-тового файлу 3-х слів до слова «замок» та 3-х слів після слова «замок»
Agg(*par:type*)	*Se*(*c* : *object*) [*Agg*(*par: type*)] – опціональна частина операції *Se*	*c* – умова пошуку, $object \in Cg$ – назва об’єкту, для якого агрегуються дані; $par \in Cg$ – назви характеристик об’єкта *object* , за якими виконується групова операція; *type* – вид групової операції (SUM, COUNT, AVG)	*Se* (‘готель’, ‘кількість зірок’>3) *Agg*(’кількість номерів’ : COUNT) – знайти кількість номерів у готелях, кількість зірок у яких більша за 3
C_c(*object*)	*Cons*(*object* [:*c*])	$object \in Cg$ – назва об’єкта, з яких треба завантажити дані *c* – умова пошуку: $par\Theta param$ ($par \in Cg$ – назва атрибута, значення якого повинно задовольняти умову, Θ – символ порівняння $=, \neq, <, >, \leq, \geq$, *param* – значення, з яким необхідно порівняти).	*Cons*(‘готель’: ‘клас’=4) – вибір з каталогу всіх об’єктів, назва таблиці або виміру яких дорівнює «готель» або синонімічній до нього назві, – вибір рядків, значення атри-буту «клас» для яких дорівнює 4, – запуск інтелектуального агента для порівняння структур даних, – автоматичне формування схо-вища консолідованих даних з атрибутами, які визначив інтелектуальний агент, – завантаження даних у сховище.
S(*par*)	*Semant*(*par*)	*par* – значення об’єкта, що шукається	*Semant* (‘замок’) – вибір з текс-тового файлу усіх слів після слова «замок» до стоп-символів, автома-тичне формування сховища консолідованих даних з атрибу-том «замок» та завантаження вибраних слів у сховище
profile(*ex*)	*profile*(*ex*)	*ex* – рівень довіри до об’єкта	*profile*(*0.3*) – вибір усіх об’єктів, рівень довіри до яких більший рівний за 0.3
$DS_3=DS_1 \cup DS_2$	*Union*(DS_1: Cg_1[,DS_2:Cg_2])	DS_1: Cg_1 – шлях до каталога даних для простору даних та назва відношення, в якому міститься перелік джерел даних (елемент каталогу даних) DS_2 – опціональне – поточний простір даних	*Union* (‘d:\my.mdf’: ’metadata_sourse’)

Закінчення табл.

1	2	3	4
$DS_3=DS_1 \cap DS_2$	*Inters*(DS_1: Cg_1[,DS_2:Cg_2])	DS_1:Cg_1 – шлях до каталога даних для простору даних та назва відношення, в якому міститься перелік джерел даних (елемент каталогу даних) DS_2 – опціональне – поточний простір даних	*Inters* (‘d:\my.mdf’: ’metadata_sourse’
$DS_3=DS_1-DS_2$	*Differ*(DS_1: Cg_1[,DS_2:Cg_2])	DS_1: Cg_1 – шлях до каталога даних для простору даних та назва відношення, в якому міститься перелік джерел даних (елемент каталогу даних) DS_2 – опціональне – поточний простір даних	*Differ* (‘d:\my.mdf’: ’metadata_sourse’

Метод інтерпретації

Послідовність кроків трансформації запиту метамови у запити до каталогу даних та синонімічного словника відбуваються так.

1. Визначається тип операції і здійснюється перевірка, чи позначення такої операції належить до множини операцій Ω_P алгебричної системи «простір даних». Якщо ні – повідомлення про помилку.
2. За типом операції визначається призначення параметра – елемент каталогу, словника чи сховища консолідованих даних.
3. Визначаємо предметну область, до якої належить об’єкт *object*.
4. Якщо об’єкт описаний більше ніж для однієї предметної області – повідомлення про помилку.
5. Якщо в каталозі даних не знайдено значення параметра, пошук у словнику синонімів, повернення на крок 3, інакше – повідомлення про відсутність.
6. Пошук $(c_1,...,c_n)$ у словнику синонімів. Перерахунок кількості звернень.
7. Трансформація параметрів запиту користувача у запит мовою джерела даних.
8. Виконання операції. Повернення результату.

Отже, вирази метамови задовольняють такі властивості:

1. Складаються з алфавіту:

$$Q = \{Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7\}, \quad (3)$$

$Q_1 = \{object_1, object_2, ...object_i, ...\}, object_i \in Cg, P(object) > 0$ – предметні змінні; $Q_2 = \{c_1, c_2, ..., c_m\}$ – вільні константи; $Q_3 = \{par_1, par_2, ..., par_n\}$ – предметні константи; $Q_4 = \{O^n\}$ – n-місні предикатні символи ($0 \le n \le 2$); $Q_5 = \{=, \neq, <, >, \le, \ge\}$ – символи порівняння; $Q_6 = \{\wedge, \vee, \neg, \exists, \forall\}$ – логічні символи; $Q_7 = \{',', '(', ')'\}$ – допоміжні символи.

2. Сигнатура метамови становить

$$\Phi = Q_1 \cup Q_2 \cup Q_3 \cup Q_4. \quad (4)$$

3. Якщо $object_c$ – вільна змінна формули b, par_j – константа, Θ – символ порівняння, то вираз вигляду $O^n(object_c(b))\Theta par_j$ є формулою.

Розроблення формалізму трансформації запиту користувача у мову доступу джерела даних

У загальному випадку у просторі даних існує нескінченна множина запитів користувачів, поданих за допомогою елементів метамови. Механізм відповіді на запит ґрунтується на встановленні відображення елементів множини запитів у елементи множини засобів реалізації (мов доступу, що використовуються в джерелах даних). Як вже зазначалося вище, усі джерела даних простору даних згруповані у три блоки (рис. 2):

- структуровані – реляційні бази даних, сховища даних (MOLAP, ROLAP), об’єктно-реляційні бази даних;
- напівструктуровані – xml, веб-сторінки, електронні таблиці;
- неструктуровані – текстові файли.

Ставиться задача побудови відображення елементів метамови простору даних в елементи мов запитів джерел даних: $\Phi \to \Phi_{source}$. При цьому виконання операцій до структурованих джерел здійснюватиметься безпосередньо джерелами даних за сформованим за допомогою трансформатора запитом. Для напівструктурованих і неструктурованих джерел даних операція виконуватиметься за допомогою засобів виконання запитів у просторі даних.

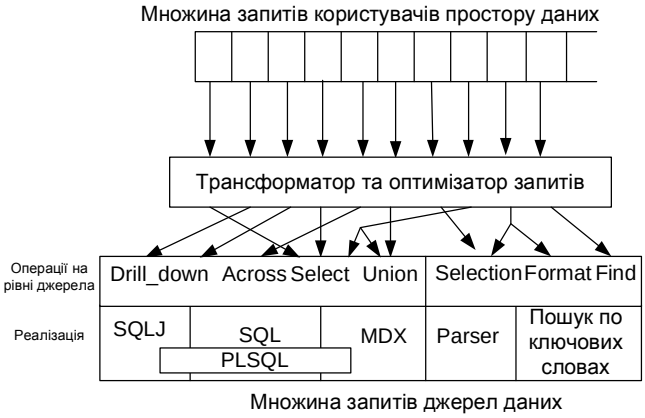


Рис. 2. Механізм трансформації запиту користувача у запит до джерела даних

Предметні змінні Q_1 відображаються:

- для баз даних – у назви відношень або атрибутів: $object \to db_i(R), object \to db_i(R(A))$,
- для сховищ даних – у назви вимірів: $object \to dw_i(Rel)$,
- для напівструктурованих даних – в заголовки стовпців, теги заголовків різних рівнів: $object \to table_i(A), object \to < A >$, де <…> – тип тегу,
- для неструктурованих даних – відображення не існує.

Вільні константи Q_2 відображаються:

- для баз даних – у значення атрибута: $c \to s_{key}(r)$,
- для сховищ даних – у значення виміру: $c \to slice_{key}(Rel)$,
- для напівструктурованих даних – у ключове слово, за яким здійснюється пошук: $c \to Find(key)$, де $Find$ – оператор пошуку,
- для неструктурованих даних – у ключове слово, за яким здійснюється пошук $c \to Find(key)$.

Предметні константи Q_3 відображаються:

- для баз даних – у синонімічні назви відношень або атрибутів: $par \to db_i(TOP\ \ 1\ \ s_{table='par'}(Dic)), object \to db_i(R(TOP\ \ 1\ \ s_{attr='par'}(Dic)))$, де $TOP\ n$ – вибір з результату запиту перших n кортежів;
- для сховищ даних – у синонімічні назви вимірів:
$$par \to dw_i(TOP\ \ 1\ \ s_{rel='par'}(Dic)),$$
- для напівструктурованих даних – в синонімічні заголовки стовпців, теги заголовків різних рівнів:
$$par \to table_i(TOP\ \ 1\ \ s_{rel='par'}(Dic)), par \to < TOP\ \ 1\ \ s_{rel='par'}(Dic) >;$$
- для неструктурованих даних – полягає у пошуку в тексті елементів словника синонімів:
$$par \to Find(s_{par}(Dic)).$$

Для прикладу – пошук всіх файлів, у яких є відомості про заробітну плату працівників. Елемент словника даних – «заробітка плата».

Опишемо перетворення елементів **операцій простору даних** Q_4 у елементи мови джерел даних за типами.

Передусім трансформовано унарні операції.

Запит до метаданих передусім визначає, де розташоване джерело.

Для структурованих даних цей запит виконується безпосередньо до каталогу та словника даних. Трансформація в SQL:

- отримання даних про джерело відповіді та місцерозташування джерела:

$$Sm_{,,object'}(place) \to p_{place}\begin{pmatrix} s_{Attr='object'}(Cg) \cup s_{table='object'}(Cg) \cup \\ \cup s_{Attr=s_{Attr='object'}(Dic)}(Cg) \end{pmatrix};$$

- визначення елементів даних в просторі даних, що можуть залежати від заданого елемента даних, і підтримання гіпотетичних запитів:

$$Sm_{,,object'}(code) \to p_{code}\begin{pmatrix} s_{code='object'}(Cg) \cup s_{table='object'}(Cg) \cup \\ \cup s_{Attr=s_{Attr='object'}(Dic)}(Cg) \end{pmatrix};$$

- визначення рівня вірогідності неточної чи неправильної відповіді:

$$Sm_{Trust}(place) \to p_{place}\left(s_{trusr>=Trust}(Cg)\right).$$

Для неструктурованих та напівструктурованих даних необхідно знайти джерело, що містить вказаний користувачем набір ключових слів. Для цього з каталогу даних вибираються усі неструктуровані або напівструктуровані джерела, відбувається почергове відкриття джерел даних та пошук по ключовому слову:

$$Sm_{object'}(place) \to p_{place}\left(s_{(M=2)\vee(M=3)}(Cg)\right) >< Find('object').$$

Структуровані запити $Se_c(object)$ вимагає трансформації у мову джерела даних

- для баз даних – вимагається виконання таких кроків:
  - вибір структурованих джерел даних: $Se_1 = p_{code}\left(s_{M=1}(Cg)\right)$;
  - визначення структур даних з каталогу для вибраних джерел: $Se_2 = p_{object}\left(s_{\text{code in } Se_1}(Cg)\right)$;
  - foreach $obj_i \in Se_2 \quad s_c(obj_i)$;
- для сховищ даних – вимагається виконання таких кроків:
  - вибір структурованих джерел даних: $Se_1 = p_{code}\left(s_{M=1}(Cg)\right)$;
  - визначення структур даних з каталогу для вибраних джерел: $Se_2 = p_{object}\left(s_{\text{code in } Se_1}(Cg)\right)$;
  - foreach $obj_i \in Se_2$

    foreach $col_k \in obj_i$

    foreach $row_j \in obj_i$

    $across(slice_c(col_k), slice_c(row_j))$;
- для напівструктурованих даних типу веб-сторінка – за словником даних визначаються необхідні теги і в них здійснюється пошук:

$$Se_c(object) \to p_{obect}(Dic >< s_{M=3}(Cg)) >< Find(c);$$

- для напівструктурованих даних типу текстовий файл з відомим форматування – за словником даних визначаються види форматування, виділяється фрагмент тексту цього форматування, копіюється його вміст:

$$object \rightarrow Find(p_{firmattype}(s_{object}(Dic)))$$

```
foreach object
Selection
  .ParagraphFormat.Alignment=Left(1,formattype)
  .Font.type = Mid(formattype, 3, 1)
   .Font.Caps = Right(formattype, 1)
   InStr(1, .Text, Right(formattype, 2);
Copy.Selection
```

- для неструктурованих даних – відображення не існує.

Запити про довільні дані – виконуються до напівструктурованих і неструктурованих джерел даних:

$$Ss_c(place) \rightarrow s_{(M=2)\vee(M=3)}(Cg) >< Find(c).$$

Операція трансформації тексту в семантичну мережу виконується для неструктурованих даних. Полягає у пошуку значень певних предметних констант. Для цього:

- шукаються предметні константи зі словника даних $par \rightarrow Find(s_{par}(Dic))$
- виділяється текст: *par*.Selection
- виділення слів, що знаходяться за знайденими предметними константами: *S*=Right (.Text, Find(‘ ‘ ∨ ’,’ ∨ ’.’))
- $Create(par, Cg')$
- $Insert(Cg', S)$

Консолідація даних – передбачає або фізичне, або логічне інтегрування даних. У разі фізичної консолідації отримані дані з джерел записуються у сховище консолідованих даних.

На етапі розроблення простору даних заповнюється словник синонімів *Dic* з наповнення стандартів, що відповідають предметній області, для якої і розробляється ПД. Потім необхідно виконати операцію пошуку даних. У разі фізичної консолідації будується сховище консолідованих даних та здійснюється завантаження.

Для різних типів джерела даних операція консолідації виконується так:

- для баз даних – вимагається виконання таких кроків:
  - побудова схеми сховища консолідованих даних за допомогою інтелектуального агента $f_{Se_c(object)}(Cg') \rightarrow Del(Diff, Dic), Create(H, Cg')$
  - $Insert(Cg', Se_c(object))$
- для сховищ даних – вимагається виконання таких кроків:
  - $Cg' = push(across(Se_c(object)))$
  - $Insert(Cg', Se_c(object))$
- для напівструктурованих даних:
  - з словника даних вибираються теги, по яких здійснювати пошук $Ss_c(object)$
  - за назвами тегів будуємо сховище даних $Create(A, Cg')$
  - $Insert(Cg', Ss_c(object))$
- для неструктурованих даних – аналогічна операції трансформації тексту в семантичну мережу.

Агрегація даних – виконується винятково в структурованих джерелах або у сховищі консолідованих даних.

- для баз даних – трансформація виконується в команду GROUP BY. Оскільки для коректного виконання команди необхідною умовою є долучення у групування параметрів, що передують параметру операції Ag, то використання вказаної операції дозволено лише в межах операції Ss. Також за наявності підтримки джерелом даних стандарту SQL/3 і вище можна використовувати GROUP BY GROUPING SETS;

- для сховищ даних – трансформація виконується в команду GROUP BY CUBE, і їй також повинен передувати запит до структурованих даних.

Профіль користувача – вибірка з каталогу даних джерел, до яких користувач має право доступу:

$$\text{profile} \rightarrow s_{ac=\text{Yes}}(Cg) >< s_{P<=ex}(Dic).$$

Далі здійснимо трансформацію бінарних операцій. Виконуються незалежно від типів джерел, що наявні у просторах даних на рівні каталогу та словника даних.

Операція *об'єднання* просторів даних:

- шукається з'єднання каталогів за місцем розміщення джерел $Cg_{temp1} = Cg_1 >< _{place} Cg_2$;
- для знайденого з'єднання перевизначаються структури даних: $Cg_{temp2} = \varnothing$;

foreach *object*

$$Cg_{temp2} = Insert(Cg_{temp2}, f_{object}(Cg_1));$$

- $Cg_1 = Cg_1 - Cg_{temp1}$;
- $Cg_2 = Cg_2 - Cg_{temp1}$;
- $Cg_3 = Cg_1 \cup Cg_{temp2} \cup Cg_2$.

Операція *перетину* просторів даних виконується аналогічно до об'єднання із заміною в останньому блоці операції на протилежну.

Операція *різниці* трансформується в традиційну без змін.

Символи порівняння Q_5 використовуються стандартно.

Логічні символи Q_6 трансформуються стандартно:

$\wedge \rightarrow AND$

$\vee \rightarrow OR$

$\neg \rightarrow NOT$

Символи $\exists, \forall$ трансформуються тільки для структурованих джерел з операцією запиту до структурованих даних:

$\exists \rightarrow$ EXISTS

$\forall$ використовується для порівняння кількості отриманих даних і кількості даних в джерелі: $\forall \rightarrow COUNT(Se_c(object)) = COUNT(s(object))$.

Допоміжні символи Q_7 використовуються стандартно.

Реляційна модель – підсистема алгебричної системи «простір даних»

Твердження: Реляційна модель – підсистема алгебричної системи сигнатури простір даних.

Доведення

Насамперед використаємо модель реляційної БД за Б.І. Плоткіним [8], де вона розглядається як певний автомат

$$B=(F, Q, R),$$

де F – множина відношень БД у зафіксованих станах; Q – реляційна алгебра (алгебра запитів), R – результат запитів (стани відношень або нові відношення у результаті застосування Q).

$$F = \{z_1, z_2, ..., z_n\},$$

$$Q = \{p, s, >< \cup, \cap, -, \bar{}\},$$

$z_i \in F$ – відношення у певному стані.

Своєю чергою, **Ip** = **{DB**, **DW**, **Wb**, **Nd**, **Gr}**, $DB_i \in$ **DB**, $DB_i = \{r_1, r_2, ..., r_n\}$, де r_i – стани відношень бази даних у певний момент часу.

Тоді можна встановити гомоморфне відображення j

$$\left(O(r_1, r_2, ..., r_n)\right)^j = f^j\left(z_1^{\ j}, z_2^{\ j}, ..., z_n^{\ j}\right), \qquad (4)$$

для якого виконуються такі властивості.

1. Арність наступних операцій збігається і вони є гомоморфними:

А) $(Se)^j = p^j : Se(z) = p(z)$,

Доведення: Коли Інформаційний продукт вироджується у реляційне відношення ($Ip_i.Ir \Rightarrow z$), то $Se: s_{Cg.x='struct'}(p_x(s_K(z)) \cup ... \cup p_x(s_K(z))) \Rightarrow Se: s_{Cg.x='struct'}(p_x(z) \cup ... \cup p_x(z)) \Rightarrow Se: s_{Cg.x='struct'}(p_x(z))$.

Б) $(Ss)^j = s^j : Ss(z) = s(z)$,

Доведення: Коли Інформаційний продукт вироджується у реляційне відношення ($Ip_i.Ir \Rightarrow z$), то $Ss: s_K(z) \cup s_K(z) \cup ... \cup s_K(z) \Rightarrow s_K(z)$.

В) $(Sm)^j = \bar{}^j : Sm(z) = \bar{z}$,

Доведення: Коли Інформаційний продукт вироджується у реляційне відношення ($Ip_i.Ir \Rightarrow z$), $Sm: s_{up}(Cg) \cup s_{up}(P)$, то результатом буде каталог даних у розрізі прав користувача (доповнення).

Г) $(s_{ac})^j = s^j : s_{ac}(z) = s(z)$,

Доведення: Коли каталог даних вироджується у реляційне відношення ($Cg \Rightarrow z$), $\text{profile} : s_{ac=\text{Yes}}(z) >< s_{P<=ex}(Dic) \Rightarrow s(z)$.

Д) $(\cup)^j = \cup^j$,

Доведення: Коли каталог даних вироджується у реляційне відношення ($Cg \Rightarrow z$), profile($f_{Ip}(Cg_1) \cup f_{Ip}(Cg_2)$) $\Rightarrow Cg_1 \cup Cg_2$, оскільки структури даних відомі та збігаються, $Cg_3 = Cg_1 \cup Cg_2$.

Е) $(\cap)^j = \cap^j$,

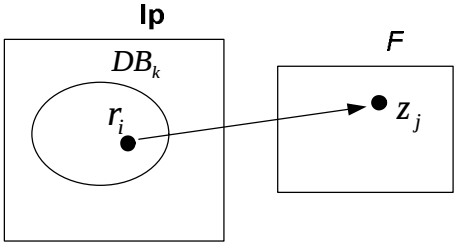
Доведення: Коли каталог даних вироджується у реляційне відношення ($Cg \Rightarrow z$), profile($f_{Ip}(Cg_1) \cap f_{Ip}(Cg_2)$) $\Rightarrow Cg_1 \cap Cg_2$, оскільки структури даних відомі та збігаються, $Cg_3 = Cg_1 \cap Cg_2$.

Є) $(-)^j = -^j$,

Доведення: Коли каталог даних вироджується у реляційне відношення ($Cg \Rightarrow z$), profile($f_{Ip}(Cg_1) - f_{Ip}(Cg_2)$) $\Rightarrow Cg_1 - Cg_2$, оскільки структури даних відомі та збігаються, $Cg_3 = Cg_1 - Cg_2$.

2. Можна побудувати індуковане унарне відношення *P*, де

$$(z_1, z_2, ..., z_n) \in F \Leftrightarrow \exists r_1, r_2, ..., r_n \begin{cases} r_1^{\ j} = z_1 \\ r_2^{\ j} = z_2 \\ \quad ... \\ r_n^{\ j} = z_n \\ (r_1, r_2, ..., r_n) \in F \end{cases}.$$



3. Нульмісний предикат Ω_{F0} для $\forall DB_i \in \mathbf{DB}$ істинний тоді, коли $R \neq \varnothing$.

Оптимізація запиту метамовою користувача простору даних

Запит користувача може бути сформований декількома різними способами. Продуктивність обчислення запиту не повинна залежати від форми запису запиту, що вибрав користувач. Тому необхідно перетворити запит у деяку еквівалентну канонічну форму. Призначенням цього перетворення є пошук подання запиту, у деякому змісті ефективнішого порівняно з вихідним поданням.

Мірою оптимізації обрано:

1) кількість записів, до який здійснюється доступ (для структурованих джерел);

2) вартість доступу до локального джерела.

1. Оптимізація кількості записів

Для першого елемента міри оптимізації здійснимо оптимізацію запитів метамовою до простору даних за аналогією до оптимізації запитів SQL, оскільки було доведено, що реляційна модель є підмоделлю алгебричної системи «простір даних» (див. 2.4).

1. Послідовність операцій Se до одного і того ж джерела може бути замінена на єдину операцію Se до цього джерела, причому умовні вирази всіх вхідних операцій за допомогою AND об'єднуються в одну умову.

$$Se_{p1}(Se_{p2}(object)) \equiv Se_{p1\text{AND}p2}(object)$$

Доведення:

$$t \in Se_{p1}(Se_{p2}(object)) \equiv (t \in object \text{ AND } p1(t))Se_{p2}(object) \equiv$$

$$(t \in object \text{ AND } p1(t)) \text{ AND } p2(t) \equiv t \in object\ Se_{p1ANDp2}(\text{object})$$

Первинний варіант потребує двох проходів під час опрацювання змінної *object*, тоді як перетворений варіант потребує тільки одного проходу.

Також визначено такі еквівалентності:

$$Se_{p1}(object) \cup Se_{p2}(object) \equiv Se_{p1ORp2}(object)$$

$$Se_{p1}(object) \cap Se_{p2}(object) \equiv Se_{p1ANDp2}(object)$$

$$object - Se_{p}(object) \equiv Se_{NOTp}(object)$$

2. У послідовності операцій Sm для одного і того ж джерела можна ігнорувати всі, окрім останньої.

$$Sm_{object}(\{Sm_{object}(X,Y,Z),Z\}) \equiv Sm_{object}(Z)$$

Доведення:

$$t\{Z\} \in Sm_{object}(\{Sm_{object}(X,Y,Z),Z\}) \equiv (t\{X,Y,Z\} \in Sm_{object}(Z)) \equiv t(Z) \in Sm_{object}(Z)$$

3. Операцію Se для результату операції Sm можна перетворити в операцію Sm для результату операції Se.

$$Sm_{object}(Se_{p}(X)) \equiv Se_{p}(Sm_{object}(X))$$

Доведення:

$$t\{X\} \in Sm_{object}(Se_{p}(X)) \equiv t(X) \in (Sm_{object}(X) \text{AND } Se_{p}(X)) \equiv$$

$$t(X) \in p(X) \text{ AND } Se_{p}(object) \equiv t(X) \in Se_{p}(Sm_{object}(X))$$

Операції Sm доцільно виконувати перед операціями Se, оскільки операції Sm зазвичай зменшують обсяг тих джерел даних, які будуть вхідними для операцій Se. Отже, у цьому випадку зменшується також і кількість даних, які необхідно сортувати для виключення ймовірних записів-дублікатів, які утворяться у процесі виконання операцій Sm.

4. Бінарні операції $\cup,\cap$ ідемпотентні:

$$DS_1 \cup DS_1 \equiv DS_1;\ \ DS_1 \cap DS_1 \equiv DS_1$$

5. Бінарні операції $\cup,\cap$ поглинають одна одну:

$$DS_1 \cup (DS_1 \cap DS_2) \equiv DS_1;\ DS_1 \cap (DS_1 \cup DS_2) \equiv DS_1$$

6. Послідовність операцій Ss до одного і того ж джерела може бути замінена на єдину операцію Ss до цього джерела, причому умовні вирази всіх вхідних операцій за допомогою AND об'єднуються в одну умову.

$$Ss_{p1}(Ss_{p2}(object)) \equiv Ss_{p1ANDp2}(object)$$

Доведення:

$$t \in Ss_{p1}(Ss_{p2}(object)) \equiv (t \in object \text{ AND } p1(t))Ss_{p2}(object) \equiv$$

$$(t \in object \text{ AND } p1(t)) \text{ AND } p2(t) \equiv t \in object\ Ss_{p1ANDp2}(\text{object})$$

Окрім того, накладені обмеження на формування запиту, не притаманні процесору оптимізації операцій реляційної алгебри:

- використання операції агрегації в межах операції запиту до структурованих даних;
- використання символів $\exists, \forall$ в межах операції запиту до структурованих даних;
- заміна операції консолідації для текстових даних на операцію трансформації тексту у семантичну мережу;
- заміна операції профілю користувача на запит до метаданих (здійснюється з метою уникнення операції з'єднання, що виконується для операції профілю).

2. Вартість доступу до локального джерела

Оскільки простір даних є розподіленою системою, то під час виконання запиту необхідно враховувати вартість доступу до джерела та вартість завантаження даних у сховище консолідованих даних. Якщо вартість доступу джерела, що присутній в запиті більше одного разу, є більшою, ніж сумарна вартість завантаження даних, то операцію вибірки доцільно замінити на операції консолідації або трансформації.

Для розв'язання цієї задачі її зведено до задачі булевого програмування:

$$\min \sum_{j=1}^{n} c_j x_j ,$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_j , \quad i = 1,..., m$$

$$x_j \in \{0,1\}, \ j = 1,..., n$$

де значення x_j визначає, яку операцію для джерела j виконувати (джерело j зустрічається в запиті більше одного разу):

$$op = \begin{cases} C, x_j = 0, M = 1 \\ S, x_j = 0, M = 2 \vee M = 3 \\ Se, x_j = 1 \end{cases};$$

c_j – вартість доступу до джерела j; a_{ij} – вартість завантаження об'єкта i з джерела j; b_j – допустима межа затрат вартості (часу) для джерела j.

Для розв'язання задачі вибору джерел для консолідації доцільно обрати алгоритм Балаша.

Проектування засобів для роботи з джерелами даних

Основним класом для роботи з джерелами даних є клас Model. Його специфікація наведена на рис. 3.

Цей клас має такі властивості й методи:

- Connection – послання на з'єднання з джерелом;
- Description – опис джерела;
- FileName – назва файла, в якому знаходиться джерело;
- MetaModelName – назва метамоделі, на основі якої розроблено джерело (тип джерела);
- Models – список пов'язаних джерел;
- Name – назва джерела;
- Entities – колекція сутностей;

- Relations – колекція відношень;
- Load – метод, який відповідає за завантаження СДІР в каталог та словник даних;
- Modify – метод, що відповідає за зміну в каталозі та джерелі даних;
- RemoveModel – метод, що відповідає за знищення інформації про джерело.

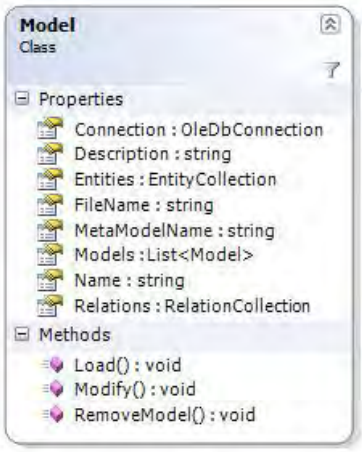


Рис. 3. Специфікація класу **Model**

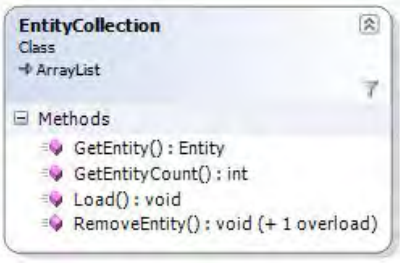


Рис. 4. Специфікація класу ***EntityCollection***

Колекція сутностей джерела описується класом EntityCollection, специфікація якого наведена на рис. 4.

Клас EntityCollection має такі методи:

- GetEntity – метод, який повертає елемент колекції сутності за іменем;
- GetEntityCount – метод, що повертає кількість екземплярів сутності, створених у поточній моделі;
- Load – метод, відповідальний за завантаження колекції сутностей поточної моделі із БД;
- RemoveEntity – метод, відповідальний за видалення сутності з колекції й БД.

На рис. 5 наведений опис класу Entity, що реалізує поняття «Сутність».

Клас Entity містить такі властивості й методи:

- Attributes – список атрибутів сутності;
- Constraints – список обмежень, що накладаються на сутність;
- Count – кількість екземплярів сутності, що може бути створене в моделях;
- Description – опис сутності;
- Name – назва сутності;
- EntityDrawType – піктограма для відображення сутності;
- Entities – колекція сутностей моделі, який належить поточна сутність;
- EntityType – тип сутності; для метамоделей це «Сутність», а для моделей тип визначається сутностями метамодели;
- Operations – список припустимих над сутністю операцій;
- Values–- список значень атрибутів сутності;
- GetAllRelations – метод, що повертає список всіх відносин сутності;
- GetInRelations – метод, що повертає список відносин, для якої ця сутність є приймачем;

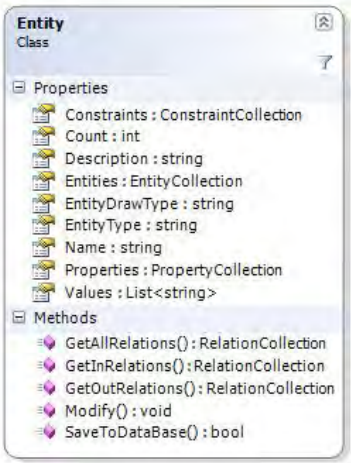


Рис. 5. Специфікація класу **Entity**

- GetOutRelations– метод, що повертає список відносин, для якої ця сутність є джерелом;
- Modify – метод, відповідальний за зміну сутності в БД;
- SaveToDataBase – метод, відповідальний за збереження сутності в БД.

Висновки

Розроблено алгоритми пошуку даних за запитом користувача та валідації отриманих результатів з метою уніфікації роботи з різнотипними джерелами даних, що дало змогу покращити якість консолідованих даних. Розроблено метамову запитів до простору даних, що дозволило уніфікувати типи запитів до усіх інформаційних продуктів, що входять в ПД. Розроблено формалізм трансформації метамови у запит до структурованих, напівструктурованих та неструктурованих джерел даних.

1. Шаховська Н.Б. Формальне подання простору даних у вигляді алгебраїчної системи / Шаховська Н.Б. // Системні дослідження та інформаційні технології = System research & information technologies: міжнародний науково-технічний журнал / Національна академія наук України, Інститут прикладного системного аналізу. – К., 2011. – № 2. – С.128 – 140. 2. Шаховська Н.Б. Формалізація простору даних за допомогою алгебраїчної системи / Н.Б. Шаховська // Радіоелектроніка. Інформатика. Управління / Запорізький національний технічний університет. – Запоріжжя: Видавництво ЗНТУ, 2010. – № 1. – С. 102–109. 3. Шаховська Н.Б. Організація просторів даних для забезпечення якості консолідованих даних / Шаховська Н.Б. // Реєстрація, зберігання і обробка даних. – К., 2011. – № 3. – С. 86–97. 4. Шаховська Н.Б. Методи опрацювання консолідованих даних за допомогою просторів даних / Н.Б. Шаховська // Проблеми програмування / Національна академія наук України, Інститут програмних систем НАН України. – 2011. – № 4. – С. 72–84. 5. Шаховська Н.Б. Особливості інтеграції даних інформаційних систем Національного університету «Львівська політехніка/ Н.Б. Шаховська, Д.О.Тарасов // Складні системи і процеси / Запорізький інститут державного та муніципального управління. – 2009. – № 2. – С. 98–109. 6. Шаховська Н. Б. Алгебраїчна система класу „простір даних» / Н.Б. Шаховська // Науковий вісник Чернівецького університету. Серія Фізика. Електроніка: Збірник наукових праць. – Чернівці, 2009. – № 479: Тематичний випуск «Комп'ютерні системи та компоненти». – С. 48–51. 7. Калиниченко Л.А. Методы синтеза канонических моделей, предназначенных для достижения семантической интероперабельности неоднородных источников информации / Калиниченко Л.А., Ступников С.А., Земцов Н.А. – М.: ИПИ РАН, 2005. – 84 с. 8. Плоткин Б.И. Универсальная алгебра, алгебраическая логика и базы данных / Плоткин Б.И. – М.: Наука, 1991. – С. 292.