УДК 004.6

ЯЦИШИН А.Ю.

ПРОЕКТУВАННЯ МУЛЬТИБАЗОВИХ СХОВИЩ ДАНИХ НА ОСНОВІ ДВОХФАЗНОГО АЛГОРИТМУ

У статті описується підхід до проектування сховищ даних на основі структурованості даних і запитів користувачів. Уводиться поняття мультибазового сховища даних, будується його математична модель та пропонується двохфазовий алгоритм проектування мультибазових сховищ даних із використанням генетичного алгоритму.

The article describes approach to design of data warehouses based on data structuredness and user queries to the data warehouses. The concept of multibase data warehouse is introduced, its mathematical model is built and two-phase building algorithm for multibase data warehouses which uses genetic algorithm is suggested.

Вступ

При проектуванні сховищ даних постає питання розташування даних у сховищі для забезпечення ефективного його функціонування.

У наукових публікаціях були запропоновані різні архітектури сховищ даних і розглянуті питання, пов'язані з джерелами даних. Однак підходу, який би описував проектування сховищ даних з врахуванням інформації про джерела даних і запити, що обробляються цим сховищем, запропоновано не було.

Метою статті є розгляд питання проектування сховищ даних з врахуванням їх структурованості та запитів до сховища.

Огляд існуючих архітектур сховищ даних

«Багатовимірне сховище даних». У книзі «The Data Warehouse Toolkit. Second Edition» Р. Кімболл (R. Kimball) запропонував підхід до побудови сховищ даних, який також відомий як підхід «знизу вгору», оскільки проектування сховища починається з нижнього рівня сховища (вітрин даних) і закінчується його верхнім рівнем (операційними базами даних) [1].

Сховище, побудоване за даним підходом, включає операційні бази даних, перехідну область, вітрини даних, засоби доступу до даних.

У цьому сховищі операційні бази даних зберігають первинну інформацію. Перехідна область слугує для перенесення даних з операційних баз даних до вітрин даних, які містять як детальні, так і агреговані дані.

Запропонована архітектура багатовимірного сховища даних дозволяє ефективно виконувати запити як до агрегованих даних, так і зберігати детальні дані. Використання архітектури шини та понять узгодженості дозволяє отримати незалежні між собою вітрини, які використовують один набір вимірів, що забезпечує можливість паралельної розробки вітрин різними підрозділами, а також цілісність усієї системи в цілому.

«Реляційне сховище даних». Другим основним підходом до побудови сховищ даних є підхід Інмона. Цей підхід відомий як підхід «зверху вниз», оскільки проектування системи починається з верхнього рівня (власне сховища даних) і закінчується нижнім рівнем (вітринами даних) [2, 3]. Сховище, побудоване за таким підходом, було названо «корпоративною інформаційною фабрикою». Крім того, наявність вітрин даних дозволяє отримувати агреговані дані та використовувати переваги систем класу on-line analytic processing (OLAP).

Таке сховище, за Інмоном, містить корпоративні застосування, перехідну область, операційне сховище даних, сховище даних підприємства, вітрини даних відділів, застосування систем прийняття рішень, сховища для дослідження або видобування даних, альтернативні системи зберігання даних.

Незалежні вітрини даних. Певною модифікацією підходу Кімболла є підхід незалежних вітрин даних, описаний у праці [4]. Основною відмінністю цього підходу є незалежність і неузгодженість між собою вітрин даних. Даний підхід не деталізує первинні бази даних, перехідну область і застосування користувачів.

Централізоване сховище даних. Узагальненим варіантом підходу Інмона є підхід до побудови централізованого сховища даних, у якому багатовимірні представлення забезпечуються реляційною базою даних [5]. Відмінністю від підходу Інмона є те, що в сховищі немає

вітрин даних. Цю архітектуру слід вибирати тоді, коли сховище даних є інтегральною частиною стратегічного рішення.

Об'єднане сховище даних. Об'єднане сховище даних [5] передбачає наявність уже розгорнутої системи (бази даних, сховища даних, застосування і т.ін.) та призначене для отримання доступу до даних з існуючих джерел. Дані інтегруються (логічним або фізичним способом), використовуючи спільні ключі, глобальні метадані, запити та інші методи. Використання цієї архітектури рекомендується в існуючій складній інфраструктурі підтримки прийняття рішень, коли фірма не хоче перебудовувати цю інфраструктуру. Основною перевагою цієї архітектури є простота проектування і можливість використання існуючих систем для побудови сховища.

Основним недоліком цієї архітектури є складність аналізу даних у вітринах різної структури та гетерогенність побудованої в результаті системи.

Цей підхід бажано використовувати у випадку, коли ресурси є обмеженими, навики IT-персоналу є низькими, і сховище не є стратегічним рішенням (наприклад, частиною домену)

Об'єднане сховище даних. Своєрідним поєднанням підходів Інмона та Кімболла є так зване об'єднане (federated) сховище даних [6], яке складається з первинних баз даних (реляційні бази даних), вітрин даних (багатовимірні сховища) ,загальної перехідної області, що виконує операції отримання, перетворення та завантаження даних ;власне об'єднаного сховища даних, в яке завантажуються дані з реляційних баз даних і з якого вивантажуються дані для багатовимірних вітрин даних.

Особливою рисою цієї архітектури є те, що в ній загальні дані розподіляються між сховищами даних і вітринами даних,.

Об'єктно-реляційне сховище даних.

У статті "Metadata for Object-Relational Data Warehouse" розглядається об'єктно-реляційне сховище даних, що будується на основі логічної архітектури сховища [7] з використанням об'єктно-орієнтованого підходу та містить шар метаданих [8].

При цьому підході більшість шарів отриманої архітектури, крім носія даних, складається з множини об'єктів різних типів, які виконують основні функції кожного компонента.

У даній архітектурі, потік даних аналогічний потокам у інших архітектурах сховищ даних, де дані збираються з різних систем операційних баз даних, узагальнюються, агрегуються та інтегруються в сховище даних і використовуються тільки для читання даних і підтримки комплексного аналізу [1, 7, 9, 10].

Ця архітектура складається з наступних компонентів.

1. Шар інтерфейсу застосувань. У цьому шарі сховища об'єкти інкапсулюють складні процеси від користувачів сховищ даних. Об'єкти даного компоненту діляться на різні групи, які обслуговують різні служби. Ґрунтуючись на їх функціональних можливостях, кожна служба відповідає на відповідні запити додатків сторонніх розробників, аналізаторів даних та інших користувачів системи сховища даних.

2. Компоненти збору даних, що можна розглядати як інструмент, який будує підсистему даних сховища. Об'єкти збору даних отримують, перетворюють і передають дані з оперативних сховищ даних в об'єктно-реляційну базу даних сховища.

3. Компоненти управління сховищем, які безпосередньо звертаються до даних сховища даних або отримують дані з інших баз даних. Вони надають послуги, які з'єднують шар інтерфейсу застосувань та об'єктно-реляційну базу даних.

4. Метадані, які застосовуються для задавання інформації про дані сховища даних.

5. Носії даних. Представлення даних, в об'єктно-реляційному сховищі даних відрізняються від такого у реляційному середовищі і об'єктно-орієнтованому сховищі даних. Наприклад, в об'єктно-реляційному сховищі даних прості дані можуть бути розміщені в багатовимірних структурах, що схожі на реляційних системи управління базами даних [7, 9, 10]. У випадку складних даних, дані можуть бути змодельовані в ієрархічні структури об'єктів, як це пропонується для об'єктно-орієнтовної СКБД.

Гібридне сховище даних. У роботі [11] було запропоновано концепцію узагальненого гібридного сховища даних.

У гібридному сховищі присутні реляційна та багатовимірна бази даних. Для проектування гібридного сховища поєднуються підходи проектування «зверху вниз» і «знизу вгору», тобто реляційна і багатовимірна бази даних проектуються в комплексі для забезпечення оптимального функціонування сховища даних у цілому.

Крім того, гібридне сховище даних має свою систему керування (СКГСД), за допомогою якої здійснюється робота з сховищем, наприклад, виконання запитів до сховища.

Проектування гібридних сховищ даних здійснюється на основі генетичного алгоритму, який визначає ознаки розміщення областей сховища, матеріалізованості та індексування стовпців таблиць сховища.

Рівні структурованості даних

Однією з істотних характеристик даних є їх структурованість, що може суттєво вплинути на дисковий простір, який займає сховище даних, тому що у випадку неструктурованих і слабко-структурованих даних вони зберігаються розріджено. Це призводить до падіння швидкодії сховища даних у залежності від операцій «введення – виведення».

Структурованість даних традиційно є одним з ключових чинників при ухваленні рішення про відповідний формат представлення даних (наприклад, реляційна бази даних для структурованих і у форматі XML для частково структурованих даних).

Цей вибір, в свою чергу, зазвичай визначає організацію даних (наприклад, з використанням теорії залежностей і нормальних форм для реляційної моделі та ієрархічної структури тегів у випадку XML), що має вирішальне значення при прийнятті рішення про те, як індексувати ці дані (наприклад, індекси на основі збалансованого дерева для реляційної моделі і індекси на основі схеми перелічення у випадку XML).

Структурованість також суттєво впливає на доступ до даних (наприклад, за допомогою SQL запитів для реляційних баз даних і XPath / XQuery для XML).

Таким чином, від структурованості даних залежать усі аспекти керування даними, а продуктивність СКБД вимірюється за даними з очікуваним рівнем структурованості (наприклад, в тесті TPC-H для реляційної моделі і XMark для даних XML).

Дані у джерелах можна класифікувати за рівнем структурованості наступним чином.

Структуровані дані. Це ті дані, які мають чітко задану структуру і точно відповідають цій структурі. Щодо структурованих даних не виникає питань щодо їх розміщення, оскільки вони зберігаються щільно в силу наповненості всіх атрибутів в всіх кортежах даних, а тому їх в розміщують в реляційних базах даних.

Слабко-структуровані дані. Існує декілька визначень слабко-структурованих даних.

За одним із них, *"слабко-структуровані дані є даними, які описують самі себе"* [12]. Відповідно до цього формулювання структури даних зберігаються разом зі своїми даними як метадані за допомогою міток. Ці мітки представляють собою семантику кожного елемента даних. Крім того, дані значення пов'язані один з одним за допомогою вбудованої ієрархії, яка є природнім зв'язком між елементами даних.

За іншим формулюванням, *"слабко-структуровані дані є даними, що не мають схеми"* [13], тобто відсутня фіксована, жорстка схема, якій повинні відповідати дані.

Існують також визначення слабко-структурованих даних, заснованих на нерегулярності даних, яку вони можуть представляти. Наприклад, до слабко-структурованих даних відносяться *"дані, які представлені деякими закономірностями* (*це не зображення або текст*)*, але, можливо, не так добре* (*структуровані*)*, як деякі реляційні дані або дані ODMG"* [14].

В роботі «Ozone: Integrating Structured and Semistructured Data» визначено слабко-структуровані дані як "*дані, які є нерегулярними або які відображають неоднорідність структури та типу, оскільки вона не може відповідати жорсткій, заздалегідь визначеній схемі* " [15].

Слабко-структуровані дані з точки зору структури мають такі ключові характеристики:

- структури цих даних є нерегулярними, неявними та носять частковий характер;
- слабко-структуровані дані є більш гнучкими в порівнянні з сильно-структурованими даними.

Основні характеристики слабко-структурованих даних з точки зору схеми даних можна описати наступним чином:

- використовуються апріорні, а не апостеріорні схеми, та індикативні, а не обмежувальні структури;
- схеми можуть не враховуватися та суттєво змінюватися;
- типи даних елементів є еклектичними та не є точними;
- немає чітких відмінностей між елементом схеми та даних.

Частково-структуровані дані. У роботі «A Performance Analysis of a Hybrid Relational-XML

Approach to Store Partially-structured Data» було дане наступне визначення частково-структурованих даних:

«Частково-структуровані дані – це гібридні дані, частково структуровані і частково слабко-структуровані. Вони містять точки входу з структурованих даних до частково структурованих даних і навпаки" [14].

Згідно цієї роботи частково-структуровані дані розміщуються в документах XML наступним чином.

Частково-структурований документ XML містить у своїй ієрархічній структурі хоча б один високо-структурований і, як мінімум, один слабко-структурований елемент, де кожне піддерево з коренем у високо-структурованому елементі може містити комбінацію високо-структурованих елементів та/або слабко- структурованих елементів, які є вузлами цього піддерева. Кожне піддерево з коренем у слабко-структурованому елементі, складається тільки з слабко-структурованих елементів в якості його вузлів.

Неструктуровані дані. Це такі дані, для яких не визначена жодна структура. Для таких даних постають суттєві проблеми їх структурованого зберігання, оскільки вони не мають метаданих, а тому відсутня будь-яка інформація щодо типів даних і рекомендацій щодо їх оптимального зберігання.

Варто зазначити, що така класифікація поширюється не тільки на дані, а й на джерела даних. У дослідженні [15] XML-документи класифікуються у залежності від ступеня структурованості даних, що містяться в документі:

- високо-структуровані документи, що містять лише високо-структуровані дані, тобто дані, які мають чітку структуру або організацію та відповідають реляційній або об'єктно-орієнтованій моделі даних;
- слабко-структуровані документи містять лише слабко-структуровані дані;
- неструктуровані документи містять лише неструктуровані дані.

Аналіз існуючих досліджень з проектування сховищ даних

Аналіз вище наведених підходів до побудови сховищ даних дозволяє зробити наступні висновки.

По-перше, існуючі архітектури сховищ даних не передбачають розподілу даних по їх структурованості.

По-друге, наявних компонент в описаних архітектурах сховищ даних недостатньо для оптимального збереження даних всіх рівнів структурованості.

Зокрема, розміщення в одній базі даних (з використанням її моделі даних) сильно-структурованих, слабко-структурованих та частково-структурованих даних має наступні недоліки.

При представленні частково структурованих даних з використанням моделі високо-структурованих даних (наприклад, реляційної моделі) отримуємо складну структуру даних із надлишком при збереженні слабко-структурованої частини даних. Будь-які незначні зміни в цій структурі даних можуть суттєво вплинути на схему високо- структурованих даних. Крім того, модель високо-структурованих даних не надає навігаційного підходу для пошуку її даних (так званий перегляд даних), яка є корисною особливістю моделей слабко-структурованих даних.

Якщо ж представити сильно-структуровані дані з використанням моделей слабко-структурованих даних, то ці моделі не враховують структуровану частину даних високо-структурованої частини документа. Це означає, що дані розглядаються як слабко-структуровані, при цьому втрачаються їх переваги як сильно-структурованих – наприклад, типізація цих даних для оптимізації їх зберігання та виконання запитів до них.

Мультибазове сховище даних

Для проектування сховища даних з врахуванням як запитів, так і структурованості даних пропонується взяти за основу концепцію гібридних сховищ даних і розширити сховище даних базою даних XML та прямим використанням файлової системи (див. [11]).

Це розширення дасть змогу розміщувати структуровані дані в реляційній базі даних, слабко-структуровані – в багатовимірній базі даних, частково-структуровані поділяти між реляційною базою даних і базою даних XML, а неструктуровані дані зберігати у вигляді файлів файлової системи. За допомогою системи керування сховища даних буде забезпечений доступ до інформації незалежно від того, де вона розміщена.

У зв'язку з вищевикладеним визначимо мультибазове сховище даних як розширення гібридних сховищ даних наступним чином.

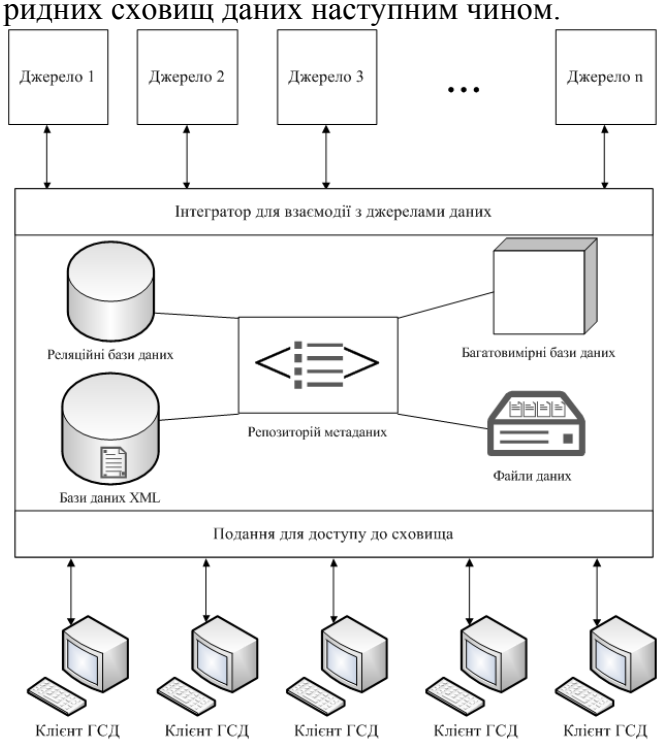

Рисунок 1. Архітектура сховища даних згідно концепції мультибазових сховищ даних

Визначення 1. Мультибазове сховище даних (рис.1) – це сховище даних, яке містить реляційні бази даних, багатовимірні бази даних, бази даних XML, файли файлової системи, репозиторій метаданих, інтегратор джерел даних, подання для доступу до сховища.

Основними характерними властивостями, які відрізняють мультибазові сховища даних від інших сховищ даних, є наступні.

1. Наявність своєї системи керування мультибазовим сховищем даних (СКМСД), за допомогою якої здійснюється робота з сховищем (виконання запитів до сховища).
2. Наявність у сховищі реляційної БД, основними призначенням якої є зберігання структурованих даних і даних, до яких здійснюється частий доступ.
3. Наявність у сховищі багатовимірної БД, яка може містити як атомарні, так і узагальнені дані. Основним призначенням багатовимірної бази даних є зберігання даних, до яких виконуються складні запити.
4. Наявність у сховищі бази даних XML, основним призначенням якої є зберігання слабкоструктурованих даних і слабкоструктурованої частини частково-структурованих даних;
5. Збереження неструктурованих даних у вигляді файлів, що зберігаються безпосередньо у файловій системі.
6. Взаємодія з джерелами даних, що здійснюється за допомогою інтегратора та полягає у відслідковуванні змін даних і метаданих, які відбуваються у джерелах, і застосуванні цих змін відповідно до налаштувань сховища даних.
7. Уніфікований доступ користувачів до сховища даних через подання сховища даних, який дає змогу користувачам звертатися до даних за допомогою єдиного інтерфейсу, незалежно від фізичного та логічного розташування цих даних у сховищі.

При проектуванні мультибазового сховища даних для забезпечення його оптимального функціонування у цілому пропонується поєднати підходи проектування на базі структурованості джерел даних і запитів до сховища даних.

Математична модель мультибазового сховища даних

Спочатку визначимо джерела даних, для яких проектується сховище даних.

У випадку розділених структурованих файлів метадані задаються частково. У таких файлах можуть бути задані лише назви полів даних, що входять у файл, а типи даних не задаються. Розділені структуровані файли визначаються через множину атрибутів даних $SF = \{sf_i\}$.

У випадку структурованих файлів XML метадані задаються за допомогою схеми, яка містить назву тегу, тип його значення та інші метадані. Файли XML визначаються через множину атрибутів тегів $X = \{x_i\}$.

У випадку реляційних баз даних структура даних задається за допомогою відношень. Відношення визначаються як впорядкований набір атрибутів відношення $R = (r_1, r_2, ..., r_1)$, де R – відношення, а r_i – атрибути відношення. Якщо два відношення $R_1 = (r_1, r_2, ..., r_1, r_{l+1}, ..., r_{l+p})$ та $R_2 = (r_l, r_{l+1}, ..., r_{l+p}, ..., r_{l+k})$ мають спільні атрибути $r_1, r_{l+1}, ..., r_{l+p}$, то в одному з відношень цей набір атрибутів є первинним ключем і приймає

унікальні значення, а в іншому він є зовнішнім ключем.

У випадку багатовимірних баз даних структура даних задається набором вимірів $D=\{d_1,d_2,...,d_q\}$ та мір $M=\{m_1,m_2,...,m_u\}$.

Використовуючи ці позначення, представимо мультибазове сховище даних (МСД) наступним чином:

$$MDW = <B,T,A,F,V> ,$$

де B – множина атрибутів даних сховища;

T – множина таблиць даних сховища;

A – множина областей сховища;

F – множина файлів сховища;

V – множина подань для доступу до сховища.

Визначення 2. *Атрибутом даних b сховища* називається іменована сутність, яка задається його назвою, типом значення та доменом (можливим діапазоном значень). Для множини атрибутів даних B сховища виконується:

$$B = \{b_i\} = SF \cup X \cup R \cup D \cup M .$$

Атрибут сховища даних може бути представлений атрибутом відношення реляційної БД, виміром або мірою багатовимірної БД, тегом XML.

Визначення 3. Таблицею t даних сховища називається впорядкована множина наборів значень атрибутів. Ці набори атрибутів називаються кортежами таблиці. Для множини таблиць T даних сховища виконується:

$$T = \{t_i \mid t_i = \{b_j\}\} .$$

Таблиця сховища даних може бути представлена таблицею реляційної БД, набором значень міри для всіх можливих значень вимірів, набором файлів XML, які мають спільну множину тегів.

Визначення 4. *Областю сховища даних a* називається множина зв'язаних таблиць, які мають хоча б один спільний атрибут:

$$A = \{a_i\} \mid a_i = \{t_j\}, j \in \overline{1,w}, \exists b_k \in t_{j_1}, b_k \in t_{j_2})\} .$$

Визначення 5. *Файлом сховища даних f* називається іменована послідовність бітів, яка зберігається у вигляді файлу файлової системи.

Визначення 6. *Поданням сховища даних v* є впорядкована трійка

$$v = <B_v,T_v,S_v>,$$

де $B = \{b_v\}$ – множина атрибутів даних, на яких визначене це подання; $T = \{t_v\}$ – множина таблиць даних, на яких визначене це подання; $S = \{s_v\}$ – множина умов вибору даних.

Задача проектування мультибазового сховища даних

Визначимо задачу проектування мультибазових сховищ даних наступним чином.

Нехай, задані множини атрибутів розділених файлів SF, файлів XML X, відношення у реляційній базі даних R, виміри D і міри M багатовимірної бази даних. Крім того, задане значення граничної частоти доступу до даних F_c.

Необхідно спроектувати МСД, а саме:

- визначити області сховища даних A, таблиці T, атрибути B, файли F;
- знайти значення структурованості джерел даних ST_s та ознак розміщення даних джерел по структурованості SL_s;
- знайти значення ознак розміщення областей по запитах L_a, індексування полів таблиць M_c та матеріалізації I_c, на яких значення цільової функції часу виконання запиту буде мінімальним серед всіх можливих наборів значень цих змінних.

Оскільки МСД є розширенням гібридного, то можна здійснити його проектування на основі запитів згідно підходу, описаного у [11].

Для врахування структурованості даних при проектуванні МСД необхідно визначити процедури розрахунку структурованості даних різних джерел і розподілу даних різного рівня структурованості у сховищі даних.

Розрахунок структурованості ST_s різних джерел даних здійснимо із використанням наступних співвідношень.

Для реляційних баз даних ST_s=1. Це випливає з того, що дані в реляційних базах є структурованими.

Для багатовимірних баз даних у якості структурованості можна взяти зважену суму наповненостей по кожній мірі

$$ST_S = \sum_{\forall M \in \{M\}} CV(\{D_M\}, M) \times WT(M), \qquad (1),$$

де M – міри джерела S, а D_M – всі виміри, на яких визначені елементи для мірі M, $CV(\{D_M\}, M)$ – структурованість для міри M; $WT(M)$ – коефіцієнт зваженості для міри M .

Структурованість CV даних міри M знаходять за формулою:

$$CV(\{D_M\}, M) = \frac{\sum_{\forall D \in \{D_M\}} E_D^M}{\prod_{\forall D \in \{D_M\}} |D|}, \qquad (2),$$

де E_D^M – кількість клітинок, які визначені по виміру D міри M.

Коефіцієнт зваженості $WT(M)$ для міри M знаходять за формулою:

$$WT(M)=\frac{\sum_{\forall D\in\{D_M\}}|D|}{\sum_{\forall M'\in S}\sum_{\forall D\in\{D_{M'}\}}|D|}. \quad (3)$$

Для повністю заповненої бази маємо

$$\sum_{\forall D\in\{D_M\}}E_D^M=\prod_{\forall D\in\{D_M\}}|D|. \quad (4),$$

При умові (4) з формул (1-3) випливає, що ST_S=1, а для пустої бази даних ST_S=0.

Для файлів даних XML можемо скористатися співвідношеннями, визначеними в [16].

Системою типів T є наступна множина

$$\tau=\{\mathrm{T}\mid\exists(s,w3type,T)\in D\} \quad (5)$$

Набором екземплярів I типу T в системі типів D є множина

$$I(T,D)=\{s\mid\exists(s,w3type,T)\in D\}. \quad (6)$$

Набором властивостей P типу T в системі типів D є множина:

$$P(T)=\{p\mid s\in I(T,D),\exists(s,p,o)\in D\}. \quad (7)$$

Кількість входжень властивості p в набір екземплярів $I(T, D)$ (позначимо через $OC(p, I(T, D))$) визначається наступним чином:

$$OC(p,I(T,D))=\\ =\left|\{s\mid s\in I(T,D),\exists(s,p,o)\in D\}\right|. \quad (8)$$

Структурованість для одного типу даних визначається за формулою:

$$CV(T,D)=\frac{\sum_{\forall p\in P(T)}OC(p,I(T,D))}{|P(T)|\times|I(T,D)|}. \quad (9)$$

Структурованість набору даних $CH(\tau,D)$ визначається за формулою:

$$CH(\tau,D)=\sum_{\forall T\in\tau}WT(CV(T,D))\times CV(T,D), \quad (10)$$

де $\mathrm{WT(CV(T,D))}$ – коефіцієнт зваженості для типу даних T у наборі D, який визначається як

$$WT(CV(T,D))=\frac{|P(T)|+|I(T,D)|}{\sum_{\forall T'\in\tau}|P(T')|+|I(T',D)|}. \quad (11)$$

Крім того, якщо у двох наборів даних (які містяться або в двох тегах одного файлу, або в двох різних файлах) є спільні теги (атрибути), то їх розраховуються як один з'єднаний набір, а загальна структурованість всього джерела розраховується наступним чином:

$$ST_S=CH(D,S)=\sum_{\forall D\in S}CH(\tau_D,D)\times\\ \times WT(CH(\tau_D,D)). \quad (12)$$

При цьому коефіцієнт зваженості визначається наступним чином:

$$WT=\frac{\sum_{\forall T\in\tau_D}(|P(T)|+|I(T,D)|)}{\sum_{\forall D'\in S}\sum_{\forall T\in\tau_D}(|P(T)|+|I(T,D)|)}. \quad (13)$$

Для нерозподілених файлів ST_S=0.

Для розподілу даних у сховищі по їх структурованості, виділимо такі класи структурованості даних.

1. Ідеально-структуровані дані. Для таких даних ST = 1, оскільки множина атрибутів кожного елемента набору даних співпадає з множиною всіх атрибутів набору

$$\forall I\in D: A(I)=A(D). \quad (14)$$

Для джерел з ідеально-структурованими даними SC_s.= 0.

2. Сильно-структуровані дані. Для таких даних:

$$\frac{|A(D)|\times|I|-|I|}{|A(D)|\times|I|}<ST<1. \quad (15)$$

При цьому існує хоча б один елемент набору даних, множина атрибутів якого дорівнює множині атрибутів набору:

$$\exists I\in D: A(I)=A(D). \quad (16)$$

Для джерел з ідеально структурованими даними SC_s.= 1.

3. Слабко-структуровані дані. Для таких даних об'єднання множин атрибутів усіх елементів набору даних дорівнює множині всіх атрибутів набору, але при цьому всі множини атрибутів елементів набору включені строго в множину всіх атрибутів набору:

$$\forall I\in D:\bigcup_{I\in D}A(I)=A(D),\\ \forall I\in D: A(I)\subset A(D). \quad (17)$$

Для джерел з слабко-структурованими даними SC_s.= 2.

4. Частково-структуровані дані. Такі дані є по суті змішаним типом між структурованими та слабко-структурованими. Для таких даних, як і для слабко-структурованих, усі множини атрибутів елементів набору строго включені в множину всіх атрибутів набору даних, але існує така підмножина атрибутів, на яких ці дані є ідеально або сильно структурованими:

$$\forall I\in D:\bigcup_{I\in D}A(I)=A(D),\\ \forall I\in D: A(I)\neq A(D),\\ \forall I\in D:\exists\{A\}\subset A(I). \quad (18)$$

Структурованість таких даних обмежена знизу наступним значенням

$$ST > 1/|A(D)|. \quad (19)$$

Для джерел з слабко-структурованими даними SC_s.= 3.

5. Неструктуровані дані. Такі дані представляються одним або декількома елементами без атрибутів:

$$|I| > 0, |A(D)| = 0. \quad (20).$$

Для джерел з слабко-структурованими даними SC_s.= 4.

Двохфазний алгоритм проектування мультибазових сховищ даних

Проектування МСД на базі структурованості даних виконаємо за допомогою двохфазного алгоритму, першою фазою якого є розміщення даних джерел на основі їх структурованості (проектування сховища даних), а другою – перерозподіл даних на основі запитів користувачів (оптимізація сховища даних).

Перша фаза алгоритму.

Як вхідні дані використовуються задані користувачем метадані джерел даних.

Вихідними даними є сховище даних з розміщеними у ньому даними.

1. Користувач визначає джерела даних $S = \{s_i\}, i = \overline{1,n}$.

2. Покласти $i = 1$.

3. Визначити структурованість ST_s джерела s_i, користуючись формулами (1, … ,13).

4. Визначити клас структурованості SC_s джерела s_i згідно формул (14, …, 20).

5. Якщо $SC_s = 0$ або $SC_s = 1$, то дані джерела s_i розмістити в реляційній базі даних (SL_s=0) і перейти до п. 9, інакше перейти до п. 6.

6. Якщо $SC_s = 2$, то дані джерела s_i розмістити в багатовимірній базі даних (SL_s=1) і перейти до п. 9, інакше перейти до п. 7.

7. Якщо $SC_s = 3$, то поділити джерело даних s_i на два джерела $s_{i'}$ та $s_{i''}$, для яких $SC_{s_{i'}} < 1$ (ідеально-структуровані та сильно структуровані дані), $SC_{s_{i'}} = 2$ (слабко-структуровані дані) та перейти до п. 5 для джерела $s_{i'}$ та розмістити дані джерела $s_{i''}$ у базі даних XML п. 6. для джерела $s_{i''}$, інакше перейти до п. 8.

8. Якщо $SC_s = 4$, то дані джерела s_i розмістити у файлах безпосередньо у файловій системі.

9. Покласти $i = i + 1$.

10. Якщо $i \le n$, перейти до п. 3, інакше перейти до п. 11.

11.Кінець першої фази.

Після виконання першої фази алгоритму сховище даних починає свою роботу.

Друга фаза алгоритму.

Як вхідні дані використовуються поточна структура сховища даних та запит до сховища даних.

Вихідними даними є сховище даних, яке оптимізоване відносно популяції запитів, до якої належить запит, що виконується.

1. Перевірити належність останнього виконаного запиту до поточної популяції запитів. Популяцією запитів вважається послідовність запитів однакового типу (точкового або інтервального) до одної і тої самої області сховища даних.

2. Якщо запит належить до нової популяції ,виконати генетичний алгоритм, інакше перейти до п. 3

3. Кінець другої фази.

Генетичний алгоритм мінімізує цільову функцією часу виконання запитів:

$$z = \sum_{t=1}^{n} t_i(L_a, \{I_c\}, \{M_c\} + \sum_{j=1}^{n-1} T_j + (T_a^{'} - \hat{T}_a)L_a), \quad (21)$$

де n – кількість таблиць запиту, а – область, що містить таблицю i, $\{I_c\}$ та $\{M_c\}$ – ознаки індексування та матеріалізації стовпчиків c таблиці i, $F_a = \sum_{i \in a} f_i$ – частота доступу до області a, $F_a^{'} = F_z$ – встановлене граничне значення частоти доступу до даних області а, $\hat{T}_a = 1/F_a$ – середній час виконання запитів, $T_a^{'} = 1/F'_a$ – граничне значення часу виконання запиту до даних області a.

Генетичний алгоритм є стандартним [17] з імовірністю мутації 0,05, імовірністю схрещування 0,8 , селекцією за методом колеса рулетки та одноточковим схрещуванням. Як початкова популяція береться особина, утворена з поточного стану сховища даних, і всі особини, у яких $\forall c\, I_c = 0$ та $\forall c\, M_c = 0$. Умовою завершення для популяції k є $f_k^* - f_{k+1}^* < \varepsilon$, де f_k^* – найменше значення цільової функції (21 серед усіх особин популяції k, ε – задане мале число.

Перша та друга фази поєднуються у загальному алгоритмі роботи МСД, що описується наступним чином.

1. Виконати першу фазу алгоритму проектування МСД для множини джерел S.

2. Поки сховище даних не отримало повідомлення про завершення роботи, виконувати наступне:

2.1. Якщо сховище даних знаходиться у стані «готовність», то:

2.1.1. Приймати запити від користувачів МСД та інтегратора.

2.1.2. Якщо отриманий запит від користувачів МСД, то:

2.1.2.1. Обробити запит, повернувши результат користувачеві.

2.1.2.2. Виконати другу фазу алгоритму проектування МСД. На час перепроектування встановити стан «обслуговування» сховища.

2.1.2.3. Встановити стан сховища «готовність» і повернутися до кроку 2.1.1.

2.1.3. Якщо отриманий запит від інтегратора про додавання або оновлення джерела даних *s*, то:

2.1.3. Виконати першу фазу алгоритму проектування МСД для джерела даних *s*. На час зміни встановити стан «обслуговування» сховища.

2.1.3.2. Встановити стан сховища «готовність» та повернутися до п. 2.1.1.

2.1.4. Якщо отримане повідомлення є запитом завершення роботи, перейти до п. 3.

3. Завершити роботу сховища даних.

Варто підкреслити, що друга фаза алгоритму проектування МСД не стосується баз даних XML і файлів, розміщених на файловій системі, а призначена лише для оптимізації виконання запитів певної популяції. Ця фаза повторюється для різних популяцій запитів,забезпечуючи оптимізацію розташування даних тих областей, до таблиць яких виконувалися запити.

Двохфазний алгоритм проектування МСД є скінченним, тому що у першій фазі алгоритму маємо скінченну кількість джерел даних, а друга фаза, яка є виконанням генетичного алгоритму для певного запиту, є скінченною в силу скінченної кількості змінних та області застосування даного алгоритму.

Збіжність алгоритму випливає зі збіжності стандартного генетичного алгоритму. Згідно теореми Голланда кількість індивідів, що зберігають, мають певний шаблон (сукупність генів, які потрібно зберегти в наступній популяції), зростає для кожної наступної популяції [17], що забезпечує отримання нових особин зі збереженням цього шаблону та кращим значенням ЦФ. У підсумку це дозволяє знайти оптимальне значення ЦФ.

Висновки

Аналіз існуючих робіт, що стосуються архітектур сховищ даних і структурованості даних показав, що при застосуванні існуючих підходів до проектування сховищ даних не можливо забезпечити ефективне розміщення даних різних класів структурованості. Запропоновано поняття мультибазових сховищ даних, для яких побудовано математичну модель сховища даних і сформульовано задачу їх проектування. Описано розрахунок структурованості джерел даних і методику визначення класів структурованості джерел даних.

Для проектування МСД запропоновано двохфазовий алгоритм, першою фазою якого є розміщення даних джерел на основі їх структурованості, другою – перерозподіл даних на основі запитів користувачів.

Використання такого підходу передбачає проектування на базі двох незалежних критеріїв розподілу даних у сховищі, при чому ці критерії застосовуються послідовно – при ініціалізації сховища та у процесі роботи сховища.

У подальших дослідженнях варто здійснити експериментальну оцінку отриманих теоретичних результатів для різних джерел даних, носіїв даних сховища даних і популяцій запитів до сховища даних, а також розглянути питання класифікації запитів до сховища даних і розподілу даних на основі запитів різних класів.

Список літератури

1. Ralph Kimball. The data warehouse toolkit: the complete guide to dimensional modeling [Текст] / Ralph Kimball – Wiley, 2002 – 436 p.
2. Claudia Imhoff.Corporate information factory [Текст] / Claudia Imhoff, Ryan Sousa – John Wiley & Sons, 2001 – 382 p.
3. W. H. Inmon. Corporate Information Factory Components [Електронний ресурс] / W. H. Inmon – Inmon Data Systems – Режим доступу : http://www.inmoncif.com/view/26.

4. Hugh J. Watson. Data Warehouse Architectures: Factors in the Selection Decision and the Success of the Architectures [Електронний ресурс] / Hugh J. Watson, Thilini Ariyachandra 2003 – Режим доступу : http://www.terry.uga.edu/~hwatson/DW_Architecture_Report.pdf.
5. Wayne Eckerson. Four Ways to Build a Data Warehouse [Електронний ресурс] // Wayne Eckerson – TDWI - 2003 – Режим доступу : http://www.tdwi.org/research/display.aspx?ID=6699
6. Douglas Hackney. Architectures and Approaches for Successful Data Warehouses. [Текст] / Douglas Hackney – www.egltd.com/presents/ArchitecturesApproaches.pdf.
7. M. Wu. Research Issues in Data Warehousing. [Текст]. Datenbanksysteme in Büro, Technik und Wissenschaft/ M. Wu, A. P. Buchmann – BTW 1997: p. 61-82.
8. Thanh N. Huynh. Metadata for Object-Relational Data Warehouse. [Текст] Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000) / Thanh N. Huynh, Oscar Mangisengi, A Min Tjoa – 2000.
9. J. M. Firestone. Object-Oriented Data Warehousing [Текст].Executive Information Systems, Inc., white paper No. 5 / J. M. Firestone – 1997.
10. Ken Orr. Data Warehousing Technology. A white paper [Текст] / Ken Orr. The Ken Orr Institute , 1996.
11. Томашевський В.М. Математична модель задачі проектування гібридних сховищ даних з врахуванням структур джерел даних [Текст]. Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. /Томашевський В.М., Яцишин А.Ю. – К.: Век+, – 2011. – № 53. – 211 с.
12. Serge Abiteboul – Querying Semi-Structured Data [Текст]. ICDT '97 Proceedings of the 6th International Conference on Database Theory / Serge Abiteboul – 1997.
13. S. Bergamaschi. Semantic Integration of Semistructured and Structured Data Sources. [Текст]. ACM SIGMOD Record – Volume 28 Issue 1, March 1999 /. S. Bergamaschi, S. Castano, M. Vincini – 1999.
14. Yasser Abdel Kader. A Performance Analysis of a Hybrid Relational-XML Approach to Store Partially-structured Data [Текст]/. Yasser Abdel Kader – University of Sheffield, Department of Computer Science – 2007.
15. Tirthankar Lahiri.– Ozone: Integrating Structured and Semistructured Data [Текст] DBPL '99 Revised Papers from the 7th International Workshop on Database Programming Languages: Research Issues in Structured and Semistructured Database Programming/ Tirthankar Lahiri, Serge Abiteboul, Jennifer Widom – London – 2000.
16. Songyun Duan. Apples and oranges: a comparison of RDF benchmarks and real RDF datasets [Текст]. SIGMOD '11 Proceedings of the 2011 international conference on Management of data / Songyun Duan, Anastasios Kementsietsidis, Kavitha Srinivas, Octavian Udrea – New York – 2011.
17. Пирогов В.В. Исследование применимости генетических алгоритмов в автоматизированном проектировании вычислительных сетей и в задачах размещения : дис. канд. техн. наук: 05.13.12 / Пирогов Владимир Витальевич – Ульяновск, 2001 – 199 с.