УДК 004.9:004.65:004.71: 654.9

РОЗРОБКА КОНЦЕПТУАЛЬНОЇ СХЕМИ ПОБУДОВИ СХОВИЩА ДАНИХ

Грицук В.Ю., аспірант,
Харківський національний автомобільно-дорожній університет

Анотація. В роботі розглянуто спосіб побудови просторових моделей на базі класичних моделей «сутність-зв'язок», який можна використовувати для проектування сховищ даних на основі даних підприємства. Реалізація здійснюється, щоб отримувати оперативну технічну базу для виконання операцій узагальнення та орієнтації в ієрархіях даних.

Ключові слова: OLAP, Data Warehouse, Онтологія, Вітрини даних, Big Data.

Сховища даних на даний момент є в практичному застосуванні однією з найбільш актуальних сфер використання сучасних технологій баз даних. Одне з найголовніших питань у їх проєктуванні полягає в тому, як розробити відповідну конструкцію бази даних для забезпечення підтримки запитів кінцевого користувача. Метод проектування сховищ даних і вітрин даних на основі моделі «сутність-зв'язок» забезпечує структуризацію сховища даних і дозволяє переконатися, що конструкція сховища відображає глибинну смислову структуру даних, що лежить в основі його побудови. Крім того, це дозволяє створити більш гнучкий дизайн сховища, який з часом може реагувати на змінні вимоги до аналізу.

Запропоновану системну архітектуру зображено на рис. 1. Система тісно поєднує сховище даних, бази даних та інтерфейс доступу до даних через API до ядра та OLAP, яке взаємодіє з клієнтом за допомогою графічного інтерфейсу користувача. Структури даних DW пропонується зробити багатовимірними:

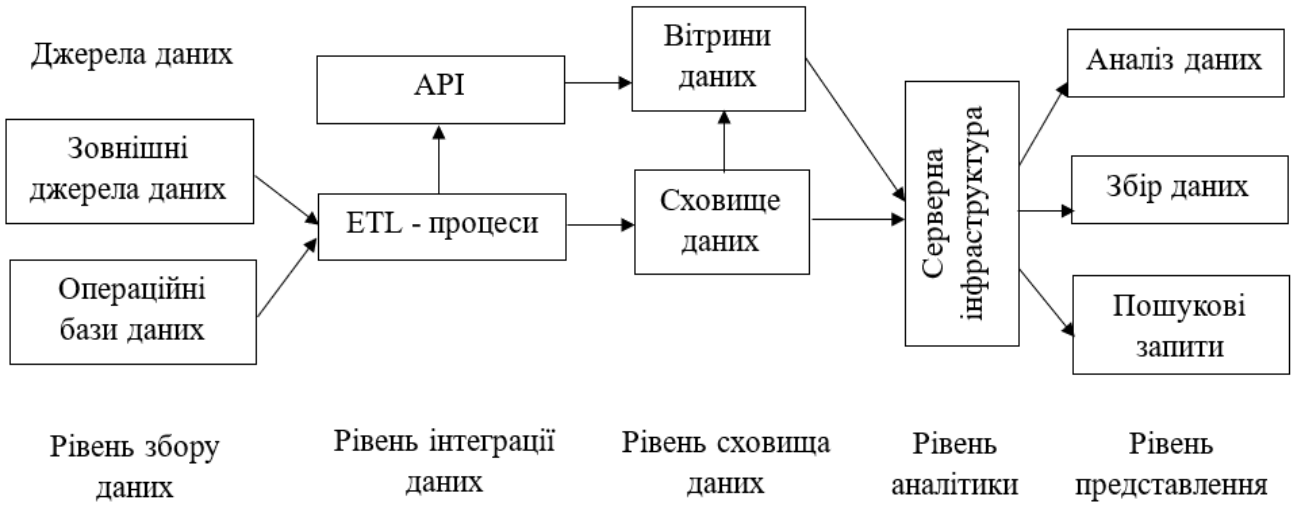

Рис. 1. Пропонована схема сховища даних

Система складається з наступних рівнів:

1. Джерела даних: у запропонованій системі передбачено використання декількох джерел даних, зокрема, реляційних баз даних та сховища даних. Кожне з джерел даних описує інтерфейс підтримуваних ним сервісів і забезпечує відповідну схему їхньої реалізації.

2. Модуль обробки ETL здійснює перетворення даних, що містяться в метаданих, тобто витягує дані і перетворює їх відповідно до правил очищення, конвертації та завантаження.

3. API доступу до даних: для всіх доступів до вхідних даних передбачено використання стандартного програмного інтерфейсу, який відокремлює програмний код алгоритмів аналізу та OLAP від інформації, що міститься у вихідних даних [1].

4. Вітрини даних (ВД) - це невелике сховище даних, яке зберігає лише певну підмножину даних, одержаних з головного сховища даних. Вміст сховищ даних являє собою дані з певної конкретної області, що представляє для нас найбільший інформаційний інтерес. Для керування даними використовуються багато серверів сховищ даних. Ці серверні засоби надають багатовимірні представлення даних для різноманітних інтерфейсних інструментів [2].

5. Інтелектуальне ядро для аналізу даних та OLAP в серверній частині: Ядро аналізу надає типові аналітичні програми для узагальнення даних, класифікації, виявлення правил об’єднання та послідовності, виявлення відхилень та прогностичного моделювання.

Операції OLAP включають згортання/ розгортання (збільшення/зменшення рівня узагальнення) вздовж однієї або декількох багатовимірних ієрархій, нарізку (виділення і проекціювання) і розгортання (переорієнтацію багатовимірного представлення даних) [3, 4, 5].

6. Система управління сховищем даних (DWMS - Data Warehouse Management System): модуль DWMS відповідає за обробку, очищення, перетворення та інтеграцію необроблених даних у DW, а також за їх перетворення та інтеграцію в DW результатів видобутку інформації. Модуль також відповідає за підтримку взаємозалежності та послідовності даних у DW.

Загалом, дані - це набір певних кількісних або якісних значень; Big Data можна розділити на три типи [5, 6]:

- Структуровані дані. Дані мають певну структуру або схему, яка організована або у вигляді реляційної бази даних, або в будь-який інший спосіб, який є зручним для обробки: наприклад, формують рядки і стовпці, в електронних таблицях (наприклад, у файлах CSV), у формі таблиць і таблиць даних, а також оброблені дані (які пройшли багато процедур з очищенням і подальшою фільтрацією).

- Напівструктуровані дані. Ці дані важко отримувати, обробляти та зберігати у вигляді структурованих даних. Для виконання цих завдань потрібне програмне середовище для роботи з Big Data (наприклад, Apache Hadoop). Наприклад, XML-файли, JSON-файли.

- Неструктуровані дані. Повністю неорганізованими даними важко оперувати, і для доступу до них необхідне сучасне програмне забезпечення та інструментальні засоби. Приклади включають відео, аудіо, зображення, електронні листи, файли Word, PowerPoint, pdf, веб-сторінки, координати місцезнаходження та потокові дані.

Для управління сховищем даних можна виділити три процеси: процес ETL та інтеграції, який забезпечує сховище початковими даними з операційних баз даних за допомогою специфічних для кожного типу джерела (ST) процесів; процес адміністрування та контролю (MD&KR), який забезпечує роботу з метаданими та базою знань (за допомогою цього процесу відбувається взаємодія між адміністратором та сховищем даних); та процес аналізу та експлуатації, який забезпечує роботу з користувацькими запитами, створення звітів, побудову кубів даних, підтримку OLAP, тощо (рис. 2). Кожен з цих типів процесів підтримує та оновлює метадані та базу знань [7]. Кожні результати аналізу у формі кубів, списків, аналітичних запитів чи будь-яких інших проміжних результатів можуть утворювати нові джерела даних, які можуть бути повторно інтегровані до сховища.

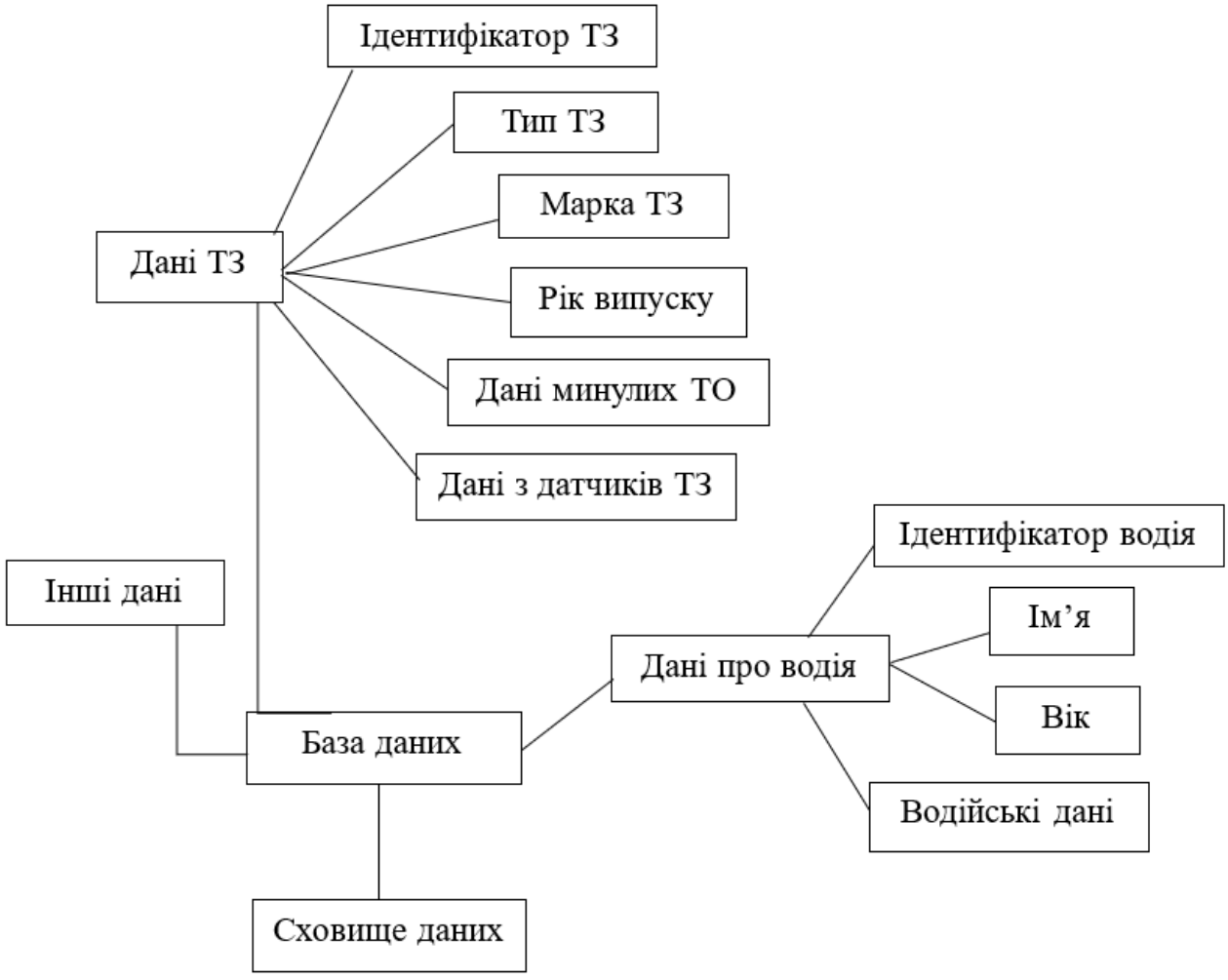

Рис. 2. Онтологічний вигляд отримання даних

Однак під час інтеграції даних семантична неоднорідність інформаційних джерел стає все більш помітною. Відповідно можна застосовувати онтологію (рис. 2) у зрозумілій для розуміння системі спосіб для представлення даних з різних типів джерел і створення глобальної моделі даних, а також за допомогою цієї структури створювати загальний словник [8].

Всі розподілені джерела даних користуються спільним словником і загальними знаннями в глобальній онтології, щоб максимально знизити рівень семантичної неоднорідності в даних з кожного ресурсу. Оскільки сховище даних призначене для підтримки прийняття рішень, важливо, щоб дані у сховищі були точними. Однак, оскільки використовуються великі обсяги даних з різних джерел, існує значна вірогідність помилок та аномалій у даних. Отже, засоби, які допоможуть виявити аномалії в даних і виправити їх, можуть бути дуже корисними. Засоби очищення даних застосовують специфічні для домену знання для очищення даних. Вони часто включають методи синтаксичного аналізу та нечітких співставлень для очищення даних з різних джерел [9].

Дані у сховищі даних в основному являють собою постійну, узагальнену або історичну інформацію, отриману в результаті аналізу даних в минулому. Сховище даних має бути розроблене таким чином, щоб воно надавало оперативну та оптимальну технічну базу для виконання операцій узагальнення та орієнтації в ієрархіях даних, а також забезпечувало легкий доступ до часових спостережень і дозволяло формувати звіти в будь-якій зручній структурі. Розглянута концепція дозволяє вирішувати наведені задачі.

Література

1. R. Agrawal, A. Arning, T. Bollinger, M. Mehta, J. Shafer, and R. Srikant. The Quest Data Mining System. In Proceedings of the 2nd Int'l Conference on Know ledge Discovery and Data Mining, 1996.
2. Hussein, Emad Saddad & El-Bastawissy, Ali & M., Hoda & Hazman, Maryam. Lake Data Warehouse Architecture for Big Data Solutions. International Journal of Advanced Computer Science and Applications, 2020. 11. 10.14569/ IJACSA.2020.0110854.
3. W. Chen, H. Wang, X. Zhang, and Q. Lin, An optimized distributed OLAP system for big data,‖ in 2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA), 2017, pp. 36– 40
4. J. Song, C. Guo, Z. Wang, Y. Zhang, G. Yu, and J.-M. Pierson, – HaoLap: A Hadoop based OLAP system for big data,‖ J. Syst. Softw., vol. 102, pp. 167–181, 2015.
5. T. John, Data Lake for Enterprises. Packt Publishing Ltd, 2017
6. W. H. Inmon and D. Linstedt, Data Architecture: A Primer for the Data Scientist: Big Data, Data Warehouse and Data Vault. 2014
7. Darmont, Jérôme & Boussaid, Omar & Ralaivao, Jean-Christian & Aouiche, Kamel. (2007). An Architecture Framework for Complex Data Warehouses.

URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type= pdf&doi=0acc942471 ad2623f993be2e39374d675cfe9a8a

8. Jing N., Fan H., Zhai Y., Liu T. (2012). Data Warehouse Design and Optimization for Drilling Engineering. The Open Petroleum Engineering Journal. 5. 124-129. 10.2174/1874834101205010124.

9. Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. *SIGMOD Rec.*, *26*, 65-74. https://dl.acm.org/ doi/pdf/ 10.1145/248603.248616

УДК 004.77

РЕАЛІЗАЦІЯ ВИДАЧІ ДОДАТКОВИХ МЕРЕЖЕВИХ МАРШРУТІВ КЛІЄНТУ VPN НА ОСНОВІ ROUTEROS

Кудінов Є.О., аспірант
Харківський національний автомобільно-дорожній університет

Анотація. Проведено аналіз можливостей операційної системи RouterOS для реалізації видачі додаткових мережевих маршрутів для VPN-клієнтів типу точка-точка в умовах використання вбудованого серверу VPN та за допомогою зовнішнього серверу DHCP.

Ключові слова: RouterOS, Mkrotik, vpn, dhcp, point-to-point

На сьогоднішній день одним із важливіших завдань для багатьох організацій та підприємств є забезпечення своїх робітників надійним та простим доступом до своєї внутрішньої мережи з різних куточків світу. Вартість проекту також є дуже важливим параметром. Таким чином, більш перспективними виглядають недорогі та надійні пристрої з багатим списком підтримуваних протоколів маршрутизації та віртуальних приватних мереж (VPN). Кількість підтримуваних протоколів дозволяє забезпечити надійний да безпечний доступ користувачів у разі різних умов підключення до провайдерів інтернет-мереж, а також забезпечувати стабільне функціонування тунелів між підрозділами.

Одним із таких рішень є пристрої фірми MikroTik, яка виробляє власні маршрутизатори та обладнання з RouterOS за порівняно низькими цінами. Це робить їх рішення привабливими для бізнесу, який потребує надійного мережного обладнання за доступною вартістю. Можливість розгортання на архітектурі ARM (популярній у сучасних одноплатних комп’ютерах) і на стандартних x86-серверах робить RouterOS універсальним рішенням для багатьох сценаріїв використання, від домашніх користувачів до великих дата-центрів.