УДК 004.384

І. І. Бородій, Я. С. Парамуд, В. В. Сав'як
Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

ПРИНЦИПИ ПОБУДОВИ ПРОГРАМНОЇ СИСТЕМИ ФОРМУВАННЯ АГРЕГОВАНИХ ДАНИХ

Розглянуто принципи побудови програмної системи формування агрегованих даних, а також основні принципи побудови програмних систем формування агрегованих даних. Проведено їхній порівняльний аналіз, запропоновано альтернативний принцип побудови програмної системи. За цим принципом побудови можна усунути проблеми швидкої та надійної обробки даних, масштабування, автоматизації роботи складових програмної системи, якості та безпеки даних.

Ключові слова: принципи побудови, програмна система, аналіз даних, агрегування даних, бізнес-інтелект, реляційна база даних, сховище даних, кіоск даних, OLAP-куб.

I. Borodii, Y. Paramud, V. Savyak
Lviv Polytechnic National University,
Computer Engineering Department

PRINCIPLES OF CONSTRUCTING A SOFTWARE SYSTEM OF THE AGGREGATED DATA FORMATION

This paper is devoted to the principles of constructing a software system of the aggregated data formation. The main principles of constructing a software system of the aggregated data formation are considered and their comparative analysis are carried out. An alternative principle of constructing a software system is proposed. This principle of constructing allows to eliminate the problems of fast and reliable data processing, scaling, automation of the software system components, improve data quality and security.

Key words: principles of constructing, software system, data analysis, data aggregation, business intelligence, relational database, data warehouse, data mart, OLAP-cube.

Вступ

Сьогодні без ефективних програмних систем збереження даних не обійдеться практично жодне підприємство. Це можуть бути різні підприємства: банки, страхові, транспортні компанії, мережі супермаркетів, телекомунікаційні та маркетингові фірми, організації, що задіяні в сферах послуг тощо [1]. Всі вони збирають, опрацьовують та зберігають дані, які містять десятки терабайт або навіть петабайт даних. На основі цих даних вирішуються найрізноманітніші завдання, пов'язані зі взаємодією із клієнтами або ж для розв'язання аналітичних задач. Проблеми обробки і збереження даних підприємства стають все актуальнішими, і багато спеціалістів у галузі інформаційних технологій намагаються постійно покращувати засоби роботи з цими даними. Спеціалісти, які займаються аналізом даних, повинні мати доступ до всієї інформації, яка їх цікавить, а також використовувати зручні і прості засоби для представлення і роботи з цією інформацією.

Задачі технологій сховищ даних (Data Warehouse) та бізнес-інтелекту (Business Intelligence) скеровані на вирішення цих проблем.

Бізнес-інтелект – це процес прийняття «інтелектуальних» бізнес-рішень аналізом доступних даних. З технологічного погляду, бізнес-інтелект об'єднує області сховищ даних, розроблення OLAP та Data Mining систем, формування звітів, візуалізації та аналізу даних [2].

Сховище даних – це організований масив даних підприємства, що обробляється і зберігається в єдиному апаратно-програмному комплексі, що забезпечує швидкий доступ до даних підприємства, багатовимірний аналіз даних, створення звітів з метою відображення статистичних даних [3].

Оскільки дані, що містяться в сховищі даних, повинні охоплювати все підприємство, їхня реалізація зазвичай займає багато часу, обсяг якого залежить від розміру підприємства. Внаслідок цих недоліків багато компаній починають з відносно простих рішень, які основані на кіосках даних (Data Mart).

Кіоск даних являє собою склад даних, що містить всі дані на рівні окремого підрозділу. Він дозволяє користувачам мати доступ до даних, які стосуються лише окремої частини компанії. Наприклад, відділ збуту зберігає всі свої дані щодо збуту в своєму власному кіоску даних, дослідний відділ – у своєму кіоску даних тощо.

Агрегування даних – це процес збирання вихідних даних з метою представлення цих даних у зведеній формі для статистичного аналізу. Наприклад, вихідні дані можуть бути агреговані протягом певного періоду часу із застосуванням статистичних функцій над даними, такими як визначення середнього арифметичного, максимального або мінімального значення, суми значень, підрахунок кількості значень. Після виконання процедури агрегування даних вони можуть бути представлені засобами формування звітів з метою прийняття бізнес-рішень [4].

Окреслення проблеми

Переважно виділяють такі основні фактори, що стосуються проблеми програмних систем формування агрегованих даних [3]:

– фактор якості даних. Внаслідок таких причин, як помилка при введенні даних, використання інших форматів представлення або одиниць виміру, відсутність своєчасного оновлення програмної системи тощо виникають некоректні дані, які є неточними і безкорисними. Такі дані називають «брудними», і саме ці дані визначають фактор якості даних у сховищах даних;

– фактор ефективності структури сховища даних. Під час проектування структури сховища даних конкретного застосування в багатьох випадках використовують засоби моделювання баз даних. Переважно сховище даних складається з таблиць, полів і записів таблиць, типів даних, обмежень стовпців, зв'язків між таблицями тощо. Проте проектувальники сховищ даних не завжди можуть переконатись в тому, що воно містить всі дані, які необхідні додаткам, які оброблятимуть ці дані, і не містить жодних даних, які не потрібні додаткам. На початкових етапах створення сховища даних часто трапляється, що в ньому відсутні дані, що необхідні для отримання звітів та отримання відповіді на запити. Одночасно присутні дані, які ніколи не буде використано в програмній системі. Наявність непотрібних даних сповільнює швидкодію системи та призводить до неефективного використання середовища зберігання;

– фактори продуктивності та масштабованості. В системах реляційних баз даних для вибірки невеликої кількості потрібних записів без повного сканування таблиці або бази даних використовують різноманітні методи доступу, такі як індекси на основі хеш-функцій або збалансованого дерева. Проте, в загальному випадку, методи доступу не допомагають при отриманні відповіді на запити, результуючий набір яких є частиною таблиці з великим обсягом даних. Крім того, серйозні проблеми спричинюють запити з використанням агрегатних функцій і переміщення файлів до бази даних. Відповідно фактори продуктивності та масштабованості є важливими при вирішенні загальної проблеми створення програмної системи;

– фактор безпеки даних [5]. Розроблення надійних засобів захисту даних було завжди актуальним протягом всієї історії існування обчислювальної техніки. Якщо ж говорити про безпеку даних, що містяться в сховищі даних, то сьогодні можна виділити такі причини виникнення ситуації, що впливає на рівень безпеки:

- різні формати і види збережених даних потребують різних підходів до безпеки;
- різні системи управління базами даних (СУБД) використовують різні інтерфейси для доступу даних, що організовані на основі однієї і тієї самої моделі;
- з'являються нові види і моделі збереження даних.

Аналіз останніх досліджень та публікацій

Базові два підходи щодо побудови програмних систем для аналізу даних та формування звітів, підхід Інмона та підхід Кімбала, розроблено в 90-ті рр. минулого століття.

Підхід Інмона – сховище даних перебуває в третій нормальній формі. Зі сховища даних формуються кіоски даних для аналізу, на основі чого формуються звіти [6].

Особливості підходу Інмона [7]:
- підхід «згори-донизу»;
- орієнтування на керування даних;
- складний у реалізації;
- низька кінцева доступність користувачів до даних;
- представлення моделі сховища даних у вигляді «сутність-зв'язок».

Підхід Кімбала – сховище даних перебуває в просторовій моделі і отримує дані з набору кіосків даних, що перебувають у третій нормальній формі [6].

Особливості підходу Кімбала [7]:
- підхід «знизу-догори»;
- орієнтування на бізнес-процеси;
- простий у реалізації;
- висока кінцева доступність користувачів до даних;
- представлення моделі сховища даних у просторовій моделі.

Порівнюючи моделі Кімбала та Інмона, можна сформувати їхні переваги та недоліки. Модель Кімбала більш масштабована через підхід " знизу–догори", отже, є можливість почати з малого та з кожним кроком все більше масштабувати дані. Крім того, вона є швидшою за модель Інмона. Через цей підхід важко створити повторно використані ETL (extract, transformation, load) структури для різних кіосків даних. З іншого боку, модель Інмона є більш структурованою та простішою для підтримки, оскільки вона є жорсткою і займає більше часу для побудови. Значною перевагою моделі Інмона є те, що сховище даних перебуває в третій нормальній формі: це легше для побудови моделі опрацювання даних. Недоліком обох моделей є складність зміни даних у сховищі даних, що ускладнює роботу з ними.

Концепції підходів Інмона і Кімбала сформовано на початку 1990-х років для відносно невеликих за обсягом баз даних, тому із сучасним суттєвим збільшенням обсягів даних підходи необхідно адаптувати до теперішніх реалій. Одночасно потрібно більше уваги приділяти засобам захисту даних.

Мета досліджень

Розробити підходи до побудови програмної системи формування агрегованих даних великих обсягів для підприємства, що відповідатиме наступним вимогам:
- забезпечувати можливість ефективного управління процесами збирання, автоматизованого завантаження, моніторингу, трансформації, агрегування даних великих обсягів;
- забезпечувати можливість формування звітів на основі агрегованих даних;
- забезпечити систему необхідними програмними засобами захисту інформації.

Основна частина роботи

Програмна система формування агрегованих даних переважно будується за технологіями бізнес-інтелекту і містить такі компоненти: джерело даних, базу даних, сховище даних, кіоск даних, аналітичну структуру даних, засоби агрегування даних, засоби створення звітів на основі отриманих агрегованих даних.

Для реалізації програмної системи формування агрегованих даних підприємства необхідно, щоб існувала підсистема введення даних для надійного та швидкого передавання даних до системи від різноманітних джерел даних. Джерело даних – це місце, звідки беруться дані, що використовуються для подальшого функціонування програмних систем. Первинним джерелом даних для програмних систем можуть бути файли різноманітних типів (docx, xlsx, pdf, xml, json, csv тощо), електронні таблиці, інформаційні листи, веб-сервіси тощо. Джерело даних як поняття найчастіше використовується в контексті з базами даних, системами управління баз даних або з будь-якими системами, що займаються опрацюванням даних [8].

Інформація в програмну систему може надходити з джерел даних цілодобово та великим потоком, тому слід розробити систему, що могла б успішно вирішувати такі завдання. Для реалізації таких систем використовують оперативні системи обробки транзакцій, побудовані на основі моделей баз даних. Існують декілька моделей бази даних:

1) ієрархічна модель – організована у вигляді деревоподібної структури з кореневим елементом, на який посилаються інші елементи даних. У цій моделі дані організовані деревоподібною структурою з відношенням «один до багатьох» між двома елементами даних [9];

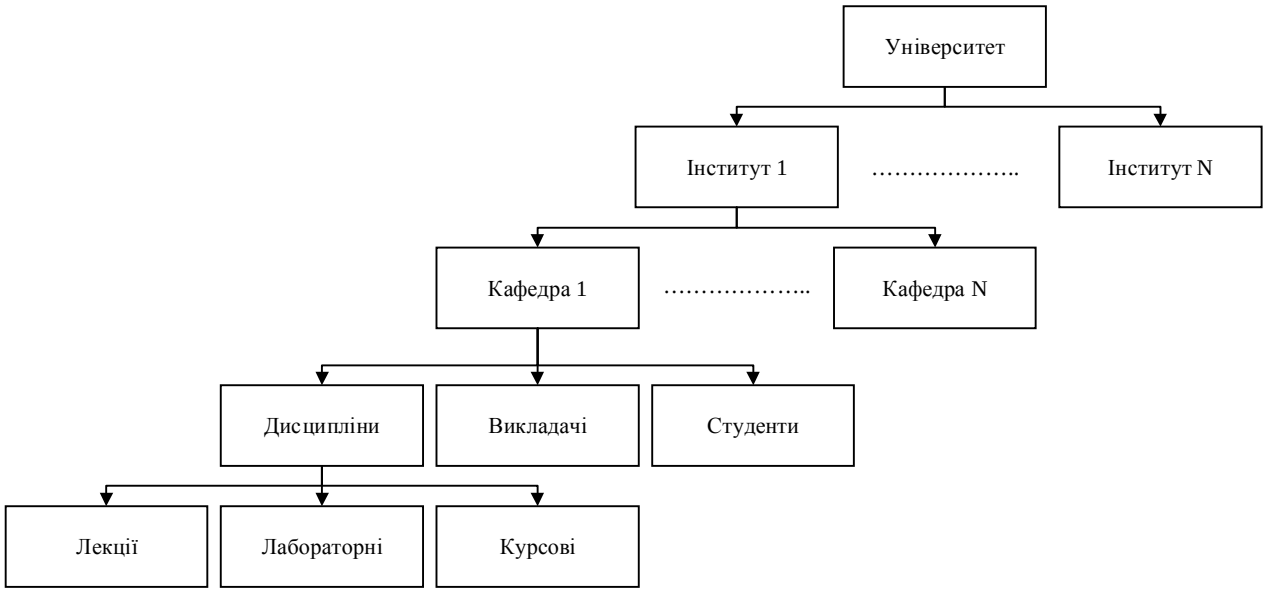

Рис. 1. Приклад ієрархічної моделі бази даних

2) мережева модель – дані організовані у вигляді графподібної структури. На відміну від ієрархічної моделі, в мережевій модель елемент даних може мати більше одного батька. Іншою суттєвою перевагою мережевої моделі є простіший і швидший доступ до даних завдяки використанню більшої кількості зв’язків між елементами даних [9];

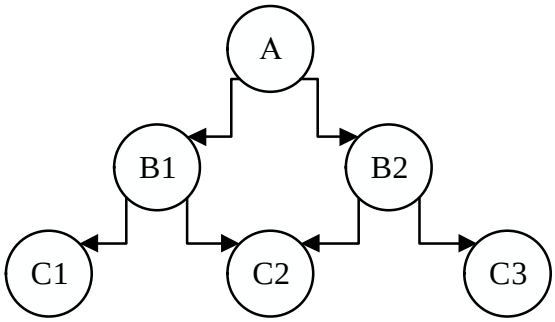

Рис. 2. Приклад мережевої моделі бази даних

3) реляційна модель – дані, організовані у вигляді сукупності таблиць, що пов'язані між собою спільними даними. Ця модель найпопулярніша. Переваги: багатокористувацький доступ, можливість надавати привілеї користувачам, простота мережевого доступу, швидкодія, надійність [10];

4) об'єктно-орієнтована модель – дані організовано навколо об'єктів, а не команд. Цю модель активно розвивають і покращують розробники. Зокрема об'єктно-орієнтована модель надає можливість описувати в межах одного інформаційного поля об'єкти, що мають різнорідні внутрішні структури та склад елементів;

5) NoSQL модель покращує продуктивність БД під час роботи з даними великих обсягів. Найбільшої продуктивності досягають у випадках, коли організація повинна проаналізувати великі фрагменти неструктурованих даних або тих даних, що зберігаються на декількох віртуальних серверах на хмарі [11];

6) хмарна модель – оптимізована або побудована для роботи у віртуальному середовищі [11]. Переваги хмарних моделей БД: можливість платити за пропускну здатність на основі кожного користувача, забезпечуючи високу масштабованість та доступність;

7) граф-орієнтована модель функціонує за теорією графів. Граф-орієнтована модель представлена у вигляді вузлів та ребер, де вузол – це об'єкт граф-орієнтованої БД, а ребро вказує на зв'язок між вузлами.

Сховища даних можна досліджувати із використанням двох основних моделей:

1) модель «сутність-зв'язок» – сховище даних перебуває в третій нормальній формі, використовується велика кількість таблиць, зв'язків між ними та мала кількість колонок у таблицях (в середньому 3);

2) просторова модель – сховище даних є максимально денормалізованим, використовується мінімізована кількість таблиць з великим обсягом даних. У такій моделі даних є два типи таблиць: таблиця фактів (містить числові дані та посилання на інші таблиці) та таблиця вимірів (містить описову інформацію).

Пропонується для побудови програмної системи формування агрегованих даних великих обсягів альтернативний підхід на таких принципах:

– виконувати декомпозицію даних на ранніх етапах завантаження за об'єктно-орієнтованими ознаками;
– використовувати реляційну модель даних;
– реалізовувати сховище даних на основі моделі "сутність – зв'язок";
– опрацьовувати дані із використанням коротких транзакцій;
– реалізувати високий рівень захисту даних.

На рис. 3 наведено схему алгоритму формування агрегованих даних.

На рис. 4 наведено приклад структурної схеми програмної системи формування агрегованих даних.

Використання реляційної моделі даних у програмній системі є доцільним, оскільки вона є одним із найбільш простих та ефективних засобів опрацювання даних. Робота з даними має відбуватися цілодобово та з використанням коротких транзакцій. Такі системи належать до класу OLTP (Online Transactional Processing) систем. OLTP-системи забезпечують ефективне розв'язання задач інформаційно-пошукового запиту. Типовим прикладом виконання операції OLTP-системою є процедура зняття грошей з банківського рахунку через банкомат. Важливими особливостями OLTP-систем є такі:

- опрацювання коротких транзакцій, що надходять великими потоками;
- велика кількість користувачів;
- постійне виконання операцій читання і запису даних з невеликою кількістю рядків.

Продуктивність систем баз даних підвищується за умови коротких транзакцій. Це пояснюється тим, що для усунення можливих негативних наслідків одночасного доступу до даних транзакції використовують блокування. Якщо виконання транзакцій займає багато часу, кількість блокувань та їх кількість збільшується, знижуючи рівень доступності даних для інших транзакцій і, як наслідок, знижуючи продуктивність системи [12].

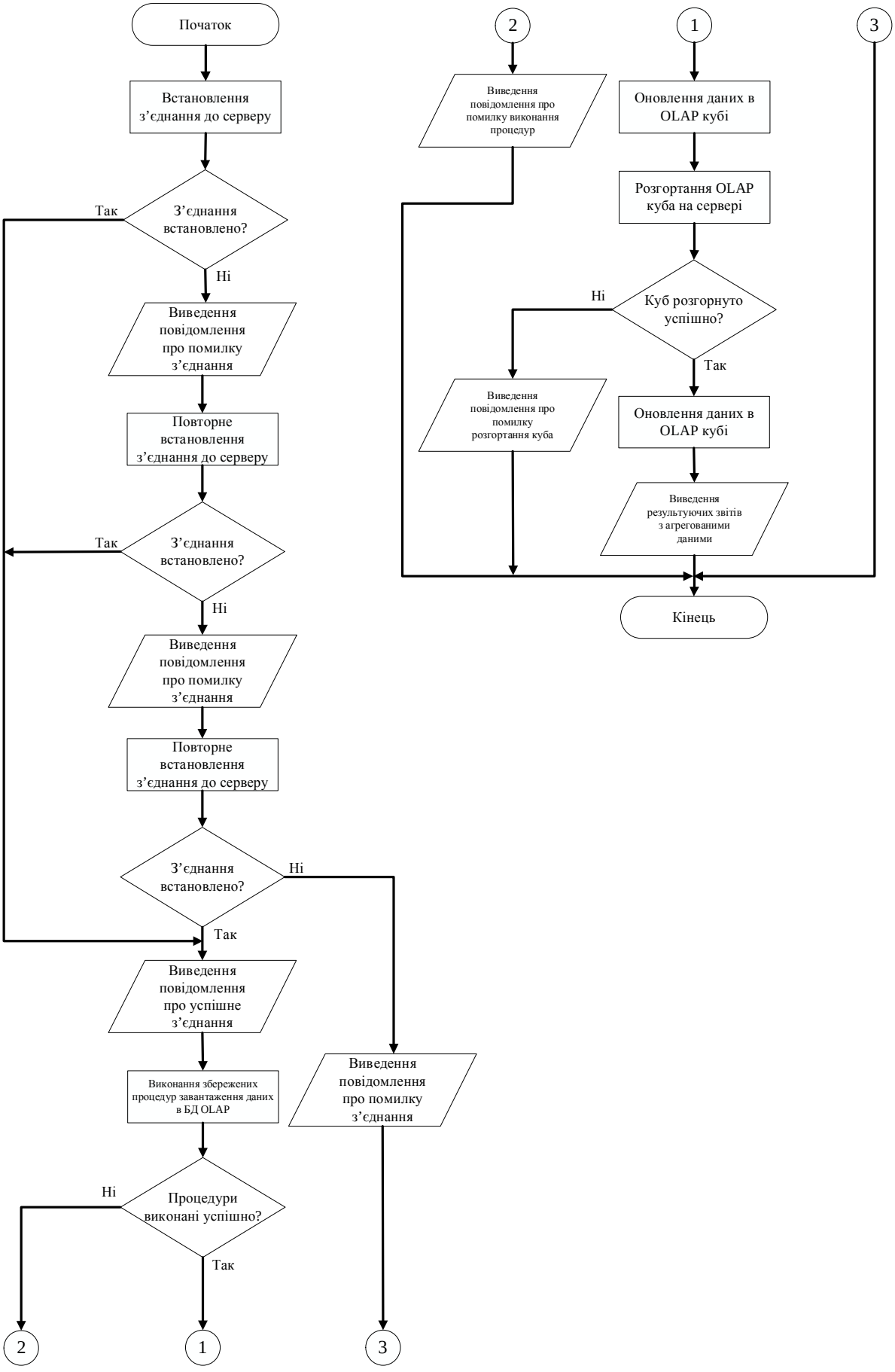

Рис. 3. Схема алгоритму формування агрегованих даних

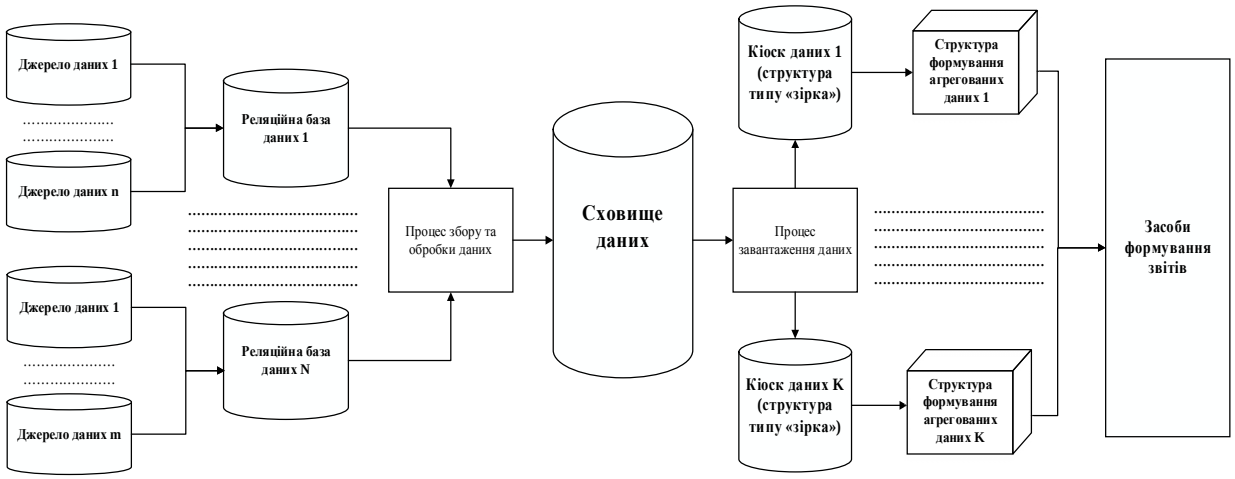

Рис. 4. Приклад структури програмної системи формування агрегованих даних

Дані OLTP-систем можуть постійно змінюватися, тобто бути динамічними, що є важливим фактором для багатьох прикладних застосувань. Усі операції в базі даних зазвичай виконуються з невеликим обсягом даних, хоча не виключено, що система бази даних повинна мати доступ до багатьох записів однієї або декількох таблиць, що зберігаються в базі даних [12].

OLTP-системи реалізовуються за допомогою СУБД, найбільш популярною і гнучкою системою для реалізації OLTP є СУБД Microsoft SQL Server, рекомендована для багатьох прикладних застосувань.

Використання Microsoft SQL Server зумовлено також тим, що система надає надійні засоби захисту інформації шифруванням симетричними/асиметричними ключами, сертифікатами. При цьому СУБД використовує алгоритм шифрування AES-256, який є доволі поширеним та надійним.

Збирання, трансформацію і завантаження даних (із OLTP) до сховища даних доцільно здійснювати ETL (extract, transformation, load) засобами. В програмній системі пропонується використання ETL-засобу компанії Microsoft за назвою SQL Server Integration Services. ETL засоби очищують дані та узгоджують їх. Цей автоматизований процес доцільно виконувати один раз на добу.

Сховище даних у програмній системі доцільно будувати на основі моделі “сутність – зв’язок”. Це означає, що сховище буде строго структурованим і спрощуватиме реалізацію ETL-процесу.

На основі структури сховища даних формуються кіоски даних – вони матимуть схему типу “зірка”. Це означає, що структурна схема складається з однієї таблиці фактів та декількох таблиць вимірів. Таблиця фактів містить числові дані, які можна агрегувати (наприклад: кількість проданих товарів, сума витрат) та даних, що містять посилання на таблиці вимірів. Таблиці вимірів містять описову інформацію виміру, наприклад, час, інформація про продукт, облікові дані співробітників тощо.

Завантаження даних зі сховища в кіоск даних пропонується виконувати за допомогою конструкцій збережених процедур мови SQL і залежно від того, наскільки часто змінюються дані в таблиці, виконувати повне або інкрементальне завантаження даних.

Повне завантаження даних: дані повністю видаляються з таблиці кіоску даних за допомогою SQL-інструкції TRUNCATE із завантаженням усіх записів, що знаходяться в таблиці сховища даних. Процес повного завантаження даних вимагає багато часу, якщо робота відбувається з великими обсягами даних, проте він простий у реалізації.

Інкрементальне завантаження даних: дані не видаляються повністю з таблиці кіоску даних. Відбуваються оновлення існуючих даних або вставка нових записів. Процес інкрементального завантаження даних займає менше часу порівняно з повним завантаженням під час роботи з великими обсягами даних, проте складніший у реалізації. Конструкції збережених процедур для інкрементального завантаження є набагато складнішими в розробленні, ніж для повного завантаження.

Схема типу “зірка” дає змогуприскорити доступ до даних, оскільки таблиця є максимально денормалізованою з великим обсягом даних і на основі “зірки”, з погляду продуктивності, вигідніше будувати системи аналізу даних. Найпоширенішою системою аналізу даних є OLAP-куби. Дані в кубі можна представити у багатовимірній формі, що доволі зручно для формування агрегованих даних.

Формується структура OLAP-кубів інструментом SQL Server Analyses Services. Вибірка даних з OLAP-куба здійснюється за допомогою багатовимірної мови запитів MDX.

MDX (Multidimensional Expressions) – SQL-подібна мова запитів, що орієнтована на доступ до багатовимірних структур даних. На відміну від SQL, що орієнтована на роботу з даними скалярних типів та реляційними моделями баз даних, мова MDX на синтаксичному рівні містить такі поняття, як виміри, ієрархії та міри, які характерні для багатовимірних моделей. Показники міри в розрізі з вимірами в мові MDX дозволяють формувати звіти про дані підприємства, проте використання лише мови MDX не дає змоги формувати звіти в графічному вигляді. Для цього можна використовувати спеціалізовані інструменти формування звітів. Найпоширенішим інструментом для формування звітів є Microsoft SQL Server Reporting Services. Зручним інструментом для формування звітів є Microsoft Excel.

Запропоновані принципи побудови програмної системи формування агрегованих даних реалізовані на конкретному застосуванні для завдань супермаркета. Програмна система коректно виконує збирання, перетворення та завантаження даних у сховище даних, усуває проблеми масштабування та продуктивності, реалізовує достатній рівень безпеки даних.

Висновки

Розглянуто основні підходи до побудови програмних систем для аналізу даних та формування звітів. Сформульовано вимоги для розроблення програмної системи формування агрегованих даних. Запропоновано для побудови програмної системи формування агрегованих даних великих обсягів альтернативний підхід, оснований на таких принципах: виконувати декомпозицію даних на ранніх етапах завантаження за об’єктно-орієнтованими ознаками; використовувати реляційну модель даних; реалізовувати сховище даних на основі моделі “сутність – зв’язок”; опрацьовувати дані із використанням коротких транзакцій; реалізувати високий рівень захисту даних. Розроблено узагальнену схему алгоритму формування агрегованих даних та приклад структурної схеми програмної системи. Така програмна система забезпечує можливість ефективного управління процесами збирання, автоматизованого завантаження, моніторингу, трансформації, агрегування даних великих обсягів, формування якісних звітів, підтримує високий рівень захисту інформації.

1. Kornilov Y. G. Sovremennoye primeneniye OLAP i OLTP tekhnologiy v ekonomike / Kornilov Y. G., Dolgova T. G. – Krasnoyarsk: Siberian State Aerospace University, Sektsiya “Informatsionno-ekonomicheskiye sistemy”, 2010. – P. 419–420. 2. Marie-Aude Aufaure – Business Intelligence // Marie-Aude Aufaure, Esteban Zimanyi (Eds.): Second European Summer School, eBISS 2012, Brussels, Belgium, July 2012. – 234 p. 3. Bat'kov V. O. – Analiz problem sovremennykh khranilishch dannykh / Bat'kov V. O. –Moscow: Moscow State University of Economics, Statistics, and Informatics, 2013. – 3 p. 4. Data aggregation [Electronic resource] / ibm – Access mode: https://www.ibm.com/support/knowledgecenter/en/SSBNJ7_1.4.4/dataView/Concepts/ctnpm_dv_use_data_aggreg.html. 5. M. A. Poltavtseva – Bezopasnost' baz dannykh: problemy i perspektivy / M. A. Poltavtseva, A. R Khabarov – Tver: Nauchno-issledovatel'skiy institut «Tsentrprogrammsistem», Programnyye produkty i sistemy No3 tom 29, 2016. – P. 36–41. 6. Aarthi Raman –Conceptual Data Vault Modeling and its Opportunities for the Future // Aarthi Raman, Teuta Cata – Dallas, TX: Decision Sciences Institute, 2017 – 10 p. 7. Data Warehouse: The Choice of Inmon versus Kimball [Electronic resource] / Ian Abramson. – Access mode: https://www.ismll.uni-hildesheim.de/lehre/bi-10s/script/Inmon-vs-Kimball.pdf. 8. Data Source [Electronic resource] / techopedia. – Access mode: https://www.techopedia.com/definition/30323/data-source. 9. DBMS Database Models [Electronic resource] / studytonight. – Access mode: https://www.studytonight.com/dbms/database-model.php. 10. The Advantages of a Relational Database Management System [Electronic resource] / techwalla. – Access mode: https://www.techwalla.com/articles/the-advantages-of-a-relational-database-management-system 11. Database [Electronic resource] / Margaret Rouse. – Access mode: https://searchsqlserver.techtarget.com/definition/database. 12. Dusan Petkovic. Microsoft SQL Server 2012. A beginner’s guide // Dusan Petkovic – New York: The McGraw-Hill Companies, 2012. – 833 p.