УДК 004:37

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

МОДЕЛЬ ВЕЛИКИХ ДАНИХ “СУТНІСТЬ-ХАРАКТЕРИСТИКА”

The problem that led to a Big database has been described in the article. The NoSQL databases features and categories are outlined. Big data model “entity-characterization” is introduced. This model allows to determin the distance between the source data on the availability of information about a particular entity.

Key words: big data, NoSQL, document-oriennted database, BigTable.

Описано задачі, що призвели до появи Великих даних. Описано особливості баз даних NoSQL та їх категорії. Введено модель Великих даних “сутність-характеристика”, що дає змогу визначити відстань між джерелами даних стосовно наявності інформації про певну сутність.

Ключові слова: великі дані, NoSQL, документно-орієнтована база даних, BigTable.

Вступ

Уведення терміну “великі дані” належить Кліффорду Лінчу, редакторові журналу Nature, який підготував у вересні 2008 р. спеціальний випуск журналу, де аналізував феномен Великих даних та їх значення для науки. Він зібрав матеріали про явище вибухового зростання обсягу і різноманітності даних, а також технологічних перспектив у парадигмі ймовірного переходу від “кількості до якості” [1].

Незважаючи на те, що термін уведений в академічному середовищі, первинною була проблема зростання кількості даних і збільшення їх різноманітності у практичних задачах. Станом на 2009 р. термін поширений у діловій пресі, а у 2010 р. з’явилася перша низка продуктів і рішень, що стосуються винятково проблем обробки Великих обсягів даних. До 2011 р. багато найбільших постачальників інформаційних технологій використовують Великі дані для формування бізнес-стратегії, зокрема IBM, Oracle, Microsoft, Hewlett-Packard, EMC [1].

Проблеми, що виникають під час опрацювання, інтерпретації, збору та організації Великих даних, з'явилися в численних секторах, а також бізнес, промисловість, некомерційні організації. Набори даних, такі як операції замовника у роздрібній торгівлі, моніторинг погоди, бізнес-аналіз, можуть швидко випереджати потужність традиційних методів та інструментів аналізу даних. Тому з’явилися нові методи та інструменти, зокрема бази даних NoSQL, MapReduce, обробка природної мови, машинне навчання, візуалізація тощо.

Загальна постановка проблеми

Великі дані є терміном, який використовується для ідентифікації наборів даних, з якими ми не можемо впоратися з використанням існуючих методології та програмних засобів через їх великий розмір і складність. Багато дослідників намагаються розробити методики і програмні засоби для передавання даних або видобування інформаційних гранул з Великих даних.

Особливості Великих даних, а саме:

- робота з неструктурованою та структурованою інформацією [10];
- орієнтація на швидке опрацювання даних [11];
- призводить до того, що традиційні мови запитів виявляються малоефективними для роботи з даними.

Тому метою статті є формальний опис різних моделей даних, виділення операцій та носіїв, а також способів їх сумісного використання.

Аналіз останніх досліджень та публікацій

Одним із способів опрацювання не тільки реляційних даних є Nosql [2].

Прихильники концепції мови NoSQL наголошують, що вона не є повним запереченням мови SQL і реляційної моделі, проект враховує те, що SQL – це важливий і доволі корисний інструмент, але при цьому його не можна вважати універсальним. Одними з проблем, які існують для класичних реляційних БД, є:

- неможливість опрацювання даних дуже великого обсягу (через використання операцій з'єднання);
- неможливість паралельної обробки;
- обмеження на застосування у проектах з великим навантаженням.

Основна мета підходу – розширити можливості БД там, де SQL недостатньо гнучка, і не витісняти її там, де вона справляється зі своїми завданнями.

В основі ідеї NoSQL є:

- нереляційна модель даних,
- розподіленість,
- відкритий вихідний код,
- можливість горизонтальної масштабованості.

У якості одного з методологічних обґрунтувань підходу NoSQL використовується евристичний принцип, відомий як теорема CAP (Consistence, Availability, Partition tolerance – «узгодженість, доступність, стійкість до поділу»), у якій стверджується, що в розподіленій системі неможливо одночасно забезпечити узгодженість даних, доступність (англ. availability, у розумінні наявності відгуку за довільним запитом) і стійкість до розщеплення розподіленої системи на ізольовані частини. У такий спосіб за необхідності досягнення високої доступності й стійкості до поділу неможливо забезпечити погодженість даних, отриману традиційними SQL-орієнтованими СУБД з транзакційними механізмами, побудованими на принципах ACID [1].

Нестроге доведення теореми CAP засноване на простих міркуваннях [4]. Нехай розподілена система складається з N серверів, кожен з яких обробляє запити деякого числа клієнтських додатків. Під час опрацювання запиту сервер повинен гарантувати актуальність інформації, для чого потрібно попередньо синхронізувати вміст його власної бази з іншими серверами. Отже, серверу необхідно чекати повної синхронізації або генерувати відповідь з урахуванням несинхронізованих даних. Можливий і третій варіант, коли з яких-небудь причин синхронізація здійснюється тільки з частиною серверів системи. У першому випадку виявляється не виконаною вимога стосовно доступності, у другому – узгодженості, у третьому – стійкості до поділу.

Існує чотири категорії баз даних NoSQL [1].

Перша категорія – це Key-Value (Ключ-Значення) бази даних. Фактично, це дуже великі хеш-таблиці, де кожному ключеві поставлене у відповідність значення. Такі бази можуть дуже швидко оперувати колосальними обсягами інформації, але мають серйозні обмеження в потужності мови запитів (тільки пошук за ключем чи значенням). Як приклади Key-Value баз даних можна навести Dynomite, Voldemort, Tokyo, Redis.

Друга категорія – клони Bigtable. BigTable – це база даних, розроблена компанією Google для власних потреб. Ця база являє собою велику таблицю із трьома вимірами: колонки, рядки й часові мітки. Така архітектура дозволяє досягти дуже високої продуктивності, крім того, вона добре масштабується на багато комп'ютерів. Але це не реляційна база, і вона не підтримує багато можливостей реляційних баз. Зокрема в Bigtable немає операцій з'єднання (join), немає складних запитів і т.д. Компанія Google не поширює Bigtable, тому на ринку з'явилося кілька незалежно розроблених клонів цієї бази. Зокрема, це такі проекти, як Hadoop, Hypertable і Cassandra.

Наступна категорія баз – це документо-орієнтовані бази даних. Такі бази трохи нагадують Key-Value бази, але в цьому випадку база даних знає, що собою являють значення. Зазвичай, значенням є деякий документ або об'єкт, до структури якого можна робити запити. Прикладами таких баз є CouchDB і MongoDB.

Четверта категорія – це бази даних, побудовані на графах. Такі бази орієнтовані на підтримку складних взаємозв'язків між об'єктами і ґрунтуються на теорії графів. Структура даних у базах даних цього типу являє собою набір вузлів, зв'язаних між собою посиланнями. При цьому і вузли, і посилання можуть мати певну кількість атрибутів. Як приклад, можна навести такі бази даних, як: Neo4j, AllegroGraph, Sones graphDB.

Також існує ще й п'ята категорія, але її не відносять до Nosql. Йдеться про об'єктно-орієнтовані бази. Такі бази призначені, насамперед, для підтримки об'єктно-орієнтованої парадигми програмування.

Відомо кілька механізмів доступу до даних у БД NoSQL.

1. Restful інтерфейси. Це інтерфейс, схожий на основний протокол Інтернету – HTTP. У межах цього підходу передбачається, що кожний об'єкт, яким ми можемо маніпулювати, має свою унікальну адресу. Звертаючись за цією адресою, ми можемо запитувати, створювати, редагувати або знищувати зазначений об'єкт. При цьому на сервері не зберігається стан запиту, тобто кожний запит опрацьовується незалежно від інших запитів.

2. Мови запитів, відмінні від SQL.

- GQL – Sql-подібна мова для Google Bigtable,
- SPARQL – мова запитів Семантичного Веба,
- Gremlin – мова обходу графів,
- Sones Graph Query Language – мова запитів до Sones Graph.

3. API запити:

- Google Bigtable Datastore API,
- Neo4j Traversal API.

Головною перевагою NoSQL баз даних є продуктивність. Якщо ще недавно існував тільки один вид баз даних для всіх випадків – реляційні бази, то сьогодні ситуація змінилася. Для кожної конкретної ситуації необхідно підбирати свою базу даних. Іноді доводиться мати одночасно кілька баз даних, у кожної з яких використовуються її найсильніші аспекти. Наприклад, у web-застосуваннях Mongodb застосовується як основне місце зберігання даних, а за допомогою Redis організовується кешування запитів користувача. У результаті ми отримуємо систему з високою продуктивністю й з доволі зручними інтерфейсами для розробників. Ще одна важлива перевага NoSQL баз даних полягає в тому, що багато представників цього сімейства сховищ даних реалізовані як проекти з відкритим кодом. Загалом порівняння SQL і NoSQL подано на рис. 1.

Рис. 1. Порівняння SQL і Nosql

Отже, існування різних категорій баз даних NoSQL вимагає формального опису моделей даних, що ними опрацьовуються.

Аналіз згаданих вище джерел, науково-популярних журналів і блогів дають змогу виділити такі вектори роботи з Великими даними:

- джерела появи Великих даних;
- апаратне забезпечення та інфраструктура;
- програмне забезпечення й способи зберігання;
- інформаційні технології (методи і засоби обробки даних);
- використання Великих даних, бізнес-аналіз.

В якості джерел появи Великих даних можна виділити пристрої та людей. Приклади перших джерел: національні та міжнародні проекти, такі як Великий адронний коллайдер (LHC) в ЦЕРН; промисловість (SCADA, фінанси і т.д.). Приклади другого типу джерел: соціальні мережі, охорона здоров'я, роздрібна торгівля, особисті дані про місцезнаходження, управління громадським сектором тощо. Наприклад, за даними аналітика Тіма Суонсона кількість операцій, що здійснюється щодня в криптовалюті Bitcoin, перевищила 100 000 операцій (оригінал у «Щоденний обсяг транзакцій в Bitcoin подолав 100-тисячний бар'єр»: [Режим доступу] http://vkurse.ua/ua/business/ezhednevnyy-obem-tranzakciy-v-bitcoin.html). За дослідженням IDC Digital Universe Study сумарний обсяг світових даних у 2005 р. становив 130 ексабайт (1018), до 2011 р. він зріс до 1227 EB, а за 2013 р. потроївся, досягши 4,4 ZB (зетабайт – 1021). Прогноз, здійснений у тому ж дослідженні, показує, що до 2020 р. обсяг цифрових даних зросте до 44 ZB (зростання щорічно на 40 %): джерело «Data Growth, Business Opportunities, and the IT Imperatives» [Режим доступу] http://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm. Розмір окремих баз даних зростає так само швидко і подолав петабайтний бар’єр (наприклад, бази даних соціальних мереж). Усе це свідчить про те, що онлайн опрацювання таких обсягів даних у розподіленому режимі онлайн практично неможливе (рис. 2).

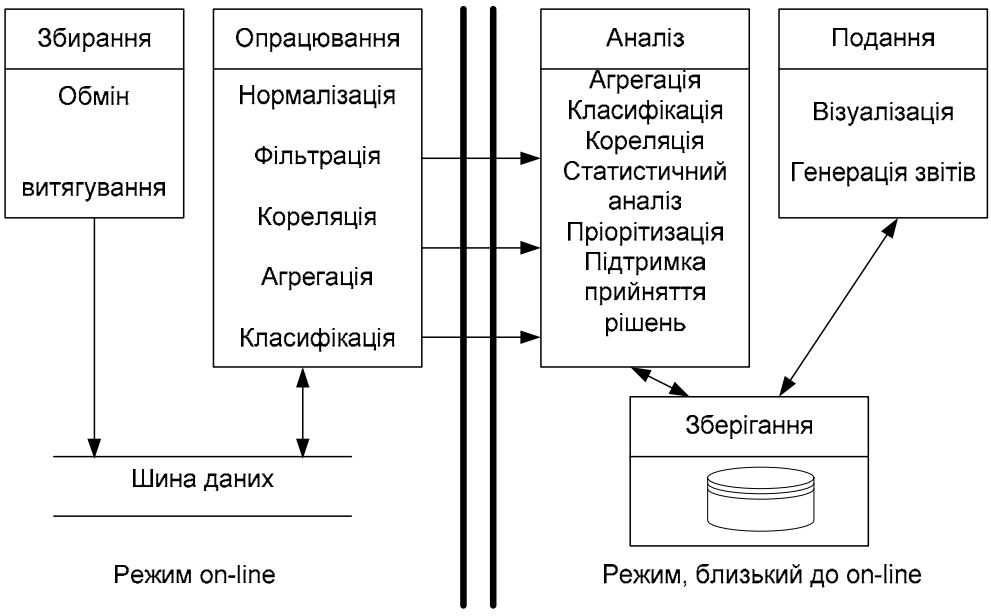

Рис. 2. Порівняльна характеристика OLAP та Великих даних

У таблиці [1] описано деякі інструменти опрацювання Великих даних з відкритим вихідним кодом, які надаються через інфраструктури хмарних обчислень. Більшість інструментів забезпечується Apache і випущені під її ліцензією. Усі ці продукти погруповано залежно від задач, що виникають під час опрацювання Великих даних.

Засоби роботи з Великими даними [1]

Засоби для Великих даних	Опис
Засоби аналізу даних	
Ambari http://ambari.apache.org	Інструмент веб для надання послуг, управління та моніторингу Apache Hadoop кластерів
Avro http://avro.apache.org	Система серіїзації даних
Chukwa http://incubator.apache.org/chukwa	Система колекціонування даних для керування великими розподіленими системами
Hive http://hive.apache.org/	Інфраструктура сховища даних, яка забезпечує агрегацію даних
Pig http://pig.apache.org	Високорівнева мова потоків даних і виконуваний framework для паралельних обчислень
Spark http://spark.incubator.apache.org	Швидкий і генеральний обчислювач для даних Hadoop. Забезпечує просту і виразну модель програмування, яка підтримує широкий спектр додатків, у тому числі ETL, машинне навчання, опрацювання потоків
ZooKeeper http://zookeeper.apache.org/	Високо продуктивна служба координації для розподілених додатків
Actian http://www.actian.com/about-us/#overview	Забезепечує зберігання «сирих» даних і готує дані для подальшого аналізу
HPCC http://hpccsystems.com	Забезпечує швидке перетвореня, паралельне опрацювання для застосувань з Великими даними
Засоби Data Mining	
Orange http://orange.biolab.si	Візуалізація та аналіз даних
Mahout http://mahout.apache.org	Бібліотека засобів машинного навчання та видобування даних
KEEL http://keel.es	Еволюційний алгоритм для задач видобування даних
Засоби соціальних мереж	
Apache Kafka	Платформа з високою пропускною здатністю для опрацювання даних в режимі реального часу
Засоби BI	
Talend http://www.talend.com	Інтеграція даних, управління, інтеграція застосувань, засоби і сервіси для Великих даних
Jedox http://www.jedox.com/en	Функції аналізу, звітності, планування
Pentaho http://www.pentaho.com	Інтеграція даних, бізнес-аналіз, візуалізація даних, прогнозування
Rasdaman http://rasdaman.eecs.jacobs-university.de/	Багатовимірні расторові дані (масив) без обмежень на розмір, наявність мови запитів
Засоби пошуку	
Apache Lucene http://lucene.apache.org	Застосування для повнотекстового індексування і пошуку
Apache Solr http://lucene.apache.org/solr	Повнотекстовий пошук, фасетний пошук, динамічна кластеризація, формати документів типу Word, PDF, просторовий пошук
Elasticsearch http://www.elasticsearch.org	Засіб розподіленого повнотекстового пошуку з веб-інтерфейсом і JSON докуметами
MarkLogic http://developer.marklogic.com	NOSQL і XML база даних
mongoDB http://www.mongodb.org	Крос-платформенна документо-орієнтована система управління базами даних з підтримкою JSON та динамічних схем
Cassandra http://cassandra.apache.org	Маштабована база даних без єдиної точки відмови
HBase http://hbase.apache.org	Маштабована розподілена база даних з підтримкою структуровано зберігання даних великого обсягу
InfiniteGraph http://www.objectivity.com	Розподілена графова база даних

Аналіз отриманих наукових результатів

Формальний опис структури Великих даних

Прикладом Великих даних є масив даних, що описує процеси функціонування регіону. Отже, у підсумку існує:

- набір сутностей: особи, місця, організації (фізичні, юридичні), дати, природні ресурси (річки, ліси, озера), рекреаційний фонд (історичні пам'ятки, санаторії), законодавчі акти та звіти;
- база даних описів сутностей: документи для інтелектуального аналізу даних, онологічні терміни, словники даних, які дозволяють зв'язати деякі об'єкти.

Грунтуючись на цій інформації, ми повинні визначити, які сутності і в який спосіб пов'язані між собою.

Формально поділимо усі об'єкти на такі категорії:

- сутності e,
- характеристики f,
- асоціації між сутностями e та характеристиками f.

Наприклад:

- ім'я e згадується у документі f,
- термін f з'явився у документі e.

Нехай також визначено:

- множину сутностей E;
- множину характеристик F;
- для кожних e і f зазначено номер асоціацій між e і f як $n_{e,f}$.

Загальна кількість сутностей визначається як $|E|$, загальна кількість характеристик є потужністю множини F: $|F|$. Також опишемо:

- для кожної характеристки f множину $e(f) = \{e \in E : n_{e,f} > 0\}$ усіх асоційованих з f сутностей;
- для кожної сутності e множину $f(ef) = \{f \in EFn_{e,f} > 0\}$ усіх асоційованих з e характеристик.

Опишемо ці якісні представлення у кількісному вигляді.

Коли у нас є кілька сутностей, пов'язаних з одною харакетристикою, використаємо кількісне подання інформації, тобто визначимо кількість бінарних запитань (так, ні), які необхідно задати, щоб знайти потрібний об'єкт. Загалом, якщо ми знаємо, що невідомий об'єкт належить до множини, що складається з N елементів, то можемо розділити цей набір на дві половини і, задаючи двійкові питання, з'ясувати, до якої половини належить шуканий об'єкт (принцип алгоритму бінарного пошуку в масиві). Отже, тоді кількість проаналізованих об'єктів становитиме $\frac{N}{2}$. Продовжимо далі таку ж процедуру: задамо друге питання, для чого поділимо виділену половину ще на дві половини. Отже, після двох запитань матимемо $\frac{N}{4}$ об'єктів, серед яких є шуканий. Після трьох запитань матимемо $\frac{N}{8}$. Загалом, після відповіді на q бінарних запитань матимемо множину з $N \cdot 2^{-q}$ елементів, що містить необхідний об'єкт. Отже, для *N* альтернатив відповідна інформація (кількість бінарних запитань) становитиме $N \cdot 2^{-q} = 1$, отже, буде $q = \log_2(N)$.

Так само можна описати сутності. Маємо $|E|$ сутностей з кількістю інформації $\log_2(|E|)$. Коли ми знаємо, що якась сутність асоційована з характеристикою (маємо $|e(f)|$ сутностей), то

кількість інформації буде $\log_2(|e(f)|)$. Отже, факт, який свідчить, що сутність пов'язана з характеристикою f, дає змогу зменшити кількість запитань до

$$\log_2(|E|) - \log_2(|e(f)|) = \log_2\left(\frac{|E|}{|e(f)|}\right).$$

Визначення кількості асоціацій також здійснюється за допомогою бінарних питань, які потрібно задати для того, щоб і надалі асоціація з необхідною сутністю була відома. Почнемо з $n_{e,f}$. Кожне бінарне запитання зменшує кількість об'єктів наполовину; q питань зменшують кількість до $n_{e,f} \cdot 2^{-q}$. Ми продовжуємо мати асоціацію до того часу, поки кількість об'єктів ≥ 1. Найбільша кількість q, для якої ми ще маємо асоціацію, визначиться як $n_{e,f} \cdot 2^{-q} = 1$, що, своєю чергою, визначається як $q = \log_2(n_{e,f})$. Задавання додаткового запитання визначається як $1 + \log_2(n_{e,f})$.

Загальна важливість характеристики f для сутності e визначається як $\log_2\left(\frac{|E|}{|e(f)|}\right)$з фактором важливості $1 + \log_2(n_{e,f})$. Результуюча кількість інформації визначається як

$$I(e, f) = (1 + \log_2(n_{e,f})) \cdot \log_2\left(\frac{|E|}{|e(f)|}\right) \quad (1).$$

Формула (1) є однією з варіацій частоти термінів – так званою зворотною частотою документа tf-idf. Для кожної сутності e маємо важливість $I(e, f)$ для різних характеристик f. Значення важливості необхідно нормалізувати:

$$V(e, f) = \frac{(1 + \log_2(n_{e,f})) \cdot \log_2\left(\frac{|E|}{|e(f)|}\right)}{\sqrt{\sum \left((1 + \log_2(n_{e,f})) \cdot \log_2\left(\frac{|E|}{|e(f)|}\right)\right)^2}}$$

Для кожної сутності e є вага $V(e, f)$. Отже, в якості міри близькості між двома об'єктами E_1 і E_2, ми можемо вважати відстань між відповідними векторами $(V(e, f_1), V(e, f_2), ...)$.

У звичайній Евклідовій відстані $d(a,b) = \sqrt{(a_1 - b_1)^2 + ...}$ додаються квадрати різниць. Отже, для кожної ваги $V(e, f)$, що репрезентує кількість бітів, матимемо

$$d(e_1, e_2) = \sum_{f \in F} |V(e_1, f) - V(e, f_2)|.$$

Ця відстань залежить від кількості характеристик: наприклад, якщо на додаток до документів ми зберігаємо їхні копії, то відстань збільшується вдвічі. Щоб уникнути цієї залежності, відстань $d(e_1, e_2)$, як правило, нормалізується в інтервалі [0,1] шляхом ділення на максимально можливе значення цієї відстані.

Коли реальні значення A і B невідомі, а відомі тільки верхні межі цих величин $\bar{a}$, $\bar{b}$, то найбільша можлива величина різниці $|\bar{a} - \bar{b}|$ дорівнює $\max(\bar{a}, \bar{b})$. Тоді за [2]:

- якщо $\bar{a} \leq \bar{b}$, то $|\bar{a} - \bar{b}| = \bar{b} - \bar{a} \leq \bar{b}$ і тому, $|\bar{a} - \bar{b}| \leq \max(\bar{a}, \bar{b})$;
- якщо $\bar{b} \leq \bar{a}$, то $|\bar{a} - \bar{b}| = \bar{a} - \bar{b} \leq \bar{a}$ і тому, $|\bar{a} - \bar{b}| \leq \max(\bar{a}, \bar{b})$.

В обох випадках маємо $\left|\bar{a}-\bar{b}\right| \le \max\left(\bar{a},\bar{b}\right)$.

Межа $\max\left(\bar{a},\bar{b}\right)$ досягається у випадках:

якщо $\bar{a} \le \bar{b}$, то для $a = 0, b = \bar{b}$;

якщо $\bar{b} \le \bar{a}$, то для $a = \bar{a}, b = 0$.

Уведемо поняття моделі асоціацій між об'єктами та характеристиками для різних категорій NoSQL баз даних.

Моделі асоціацій між сутностями та характеристиками для різних категорій NoSQL баз даних

Носій даних у моделі «ключ-значення» (інша назва – колонкова БД) описується кортежами вигляду:

$$KV = \{< f, e >\},$$

де f – ключ, який приймає унікальні значення у кожній парі, e – значення, що відповідає цьому ключеві.

Сигнатура моделі така:

$$O = \langle \pi, \sigma \rangle,$$

де π – операція проекції за атрибутами (ключ або значення), σ – селекції атрибутів (вибір значення за ключем, ключів за значенням, ключів за значенням предків). Перераховані операції належать до категорії читання [5, 6].

Прикладом СУБД колонкового типу є Cassandra.

Для версійного розподіленого зберігання великих обсягів даних була спроектована модель, що використовується у системі BigTable компанії Google з такими характеристиками:

- не повна реляційна модель даних,
- підтримка динамічного контролю над розміщенням даних.

Основа моделі даних Bigtable проста: рядки, стовпці й тимчасові мітки.

$$BigTable = \{< r, c, t >\}$$

Наприклад, у базі даних пошукової машини іменами рядків можуть слугувати адреси документів з Інтернету, а іменами стовпців – особливості цих документів (наприклад, зміст документа може зберігатися в стовпці «content:», а посилання на дочірні сторінки – в шпальтах «anchor:»). Інший приклад – карти Google, що складаються з мільярдів зображень, кожне з яких деталізує ту чи іншу географічну ділянку планети. У Bigtable карти Google структуруються так: кожному рядку відповідає один географічний сегмент, а стовпцями є зображення, з яких цей сегмент складається, у різних стовпцях зберігаються зображення з різною деталізацією.

Якщо в кількох стовпцях зберігаються дані одного типу, такі стовпці, згідно з моделлю Bigtable, утворюють сімейство: $colF = \{c_i, c_j \mid dom(c_i) \in T \wedge dom(c_j) \in T\}$. Використовувати сімейство стовпців зручно хоча б для того, щоб стиснути однорідні дані, тим самим зменшивши обсяг. Саме сімейства стовпців є одиницею доступу до даних.

Рядки Bigtable (їх максимальна довжина може досягати 64 кілобайти) теж важливі. Операція звернення до рядка є атомарною (це означає, що поки одна програма звертається до рядка, жодна інша не має права змінювати дані в сімействах стовпців цього рядка). А ще рядки зручно сортувати. У прикладі з URL документа, зробивши його запис реверсивним, легко впорядкувати всі рядки за іменем домена третього рівня.

Вміст сторінок в Інтернеті постійно змінюється. Щоб врахувати ці зміни, кожній копії даних, що зберігаються в стовпці, присвоюється тимчасова мітка (timestamp). У Bigtable тимчасовою міткою слугує 64-розрядне число, яким можна кодувати час і дату так, як це потрібно клієнтським програмам. Наприклад, timestamp для копій веб-сторінки в стовпці «contents:» є датою і часом створення цих копій. Використовуючи тимчасові мітки, додатки можуть задати в Bigtable пошук, наприклад, тільки найостанніших копій даних [4].

Отже, для будь-якої предметної області в сервісі Google можна створити власну карту даних Bigtable, що містить задану кількість рядків і унікальний для цієї предметної області набір сімейств стовпців. Повтори даних у стовпцях упорядковуються за значеннями тимчасових міток. Усе це вказує на повну відсутність підтримки властивостей ACID.

Головною перевагою цього підходу є те, що таку базу неважко порізати на незалежні шматочки і розподілити по множині серверів. Відсортовані за алфавітом рядки діляться на діапазони, які називають «таблет» (tablet) – несамостійними таблицями. Оскільки рядки в кожному таблеті відсортовані за ключовим іменем, то клієнтським додаткам просто знайти потрібний таблет, а в ньому – потрібний рядок [4].

Для синхронізації таблетів використовують сервіс Chubby. Його роль в Bigtable можна порівняти з роллю транзакцій в звичайних СУБД. Для кожного таблет-сервера Chubby створює спеціальний chubby-файл. Завдяки цьому файлу Bigtable може визначити, які з серверів працездатні. Ще один chubby-файл містить посилання на розташування кореневого таблета (Root-tablet) з даними про розташування усіх інших. Цей файл повідомляє майстру, який з серверів якими таблетами керує.

Безумовно, використання сервісу Chubby в Bigtable якоюсь мірою виконує завдання підтримки несуперечності даних у розподіленому середовищі з безліччю реплік. Але несуперечливість буває різною. Bigtable стала першою спробою досягти балансу між продуктивністю системи, її масштабованістю і непротиріччям даних, що тут зберігаються [9]. Результатом стала підтримка так званої слабкої несуперечності, яка, в принципі, задовольняла вимоги більшості працюючих з Bigtable сервісів.

На рис. 3 показано як користувач шукає свій таблет [4].

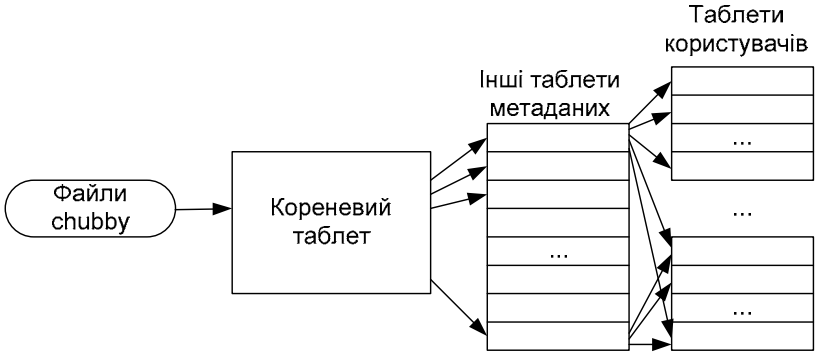

Рис. 3. Ієрархія таблетів

Носій моделі *«об'єкт-документ»* описується кортежами вигляду:

$$OD = \left\langle f_0, \left\langle f_1 : e_1,\ f_2 : e_2, ...,\ f_n : e_n \right\rangle, \left\langle f_1 : d_1,\ f_1 : d_2, ...,\ f_{n+l} : d_l \right\rangle \right\rangle,$$

де f_0 – ідентифікатор документа, $f_1 .. f_n$ – характеристики (атрибути) документа, $e_1 .. e_n$ – атомарні значення характристик $f_1 .. f_n$, $d_1 .. d_l$ – посилання на інші документи, $d_i = e(f_i)$.

Операції цієї моделі є об'єктні.

Операція визначення вузлів елемента

$$v(f_i) = \{C\} \cup \{od_i \mid i = \overline{1,n}\} \cup \{e(f_i) \mid i = \overline{0,n+l}\},$$

де C – колекція документів od_i.

Операція визначення значень вузлів:

$$v(f_i) = \{e_{ij} \mid i = \overline{1,n},\ j = \overline{0,m+l}\},$$

де e_{ij} – значення атрибутів f_i.

Також визначено відношення над елементами носія.

Відношення «елемент-елемент» визначаються між документами та колекцією:

$$OD \times C \to EE .$$

Відношення «елемент-атрибут»:

$$f_i \times OD \rightarrow EA \cdot$$

Відношення «елемент-посилання»:

$$f_i \times d_j \rightarrow ER \cdot$$

Відношення «елемент-дані» визначаються так:

$$f_i \times e_j \rightarrow ED \cdot$$

Прикладами СУБД цього типу є MongoDB та CouchDB.

Графова модель даних подана як:

$$O = \langle ID, A, z, r \rangle,$$

де *ID* – множина ідентифікаторів, вузлів графа; *A* — множина позначених спрямованих дуг (*p*,*l*,*c*), $p, c \in ID$, *l* — « рядок-мітка»; запис (*p*,*1*,*c*) означає, що між вузлами *p* та *c* є зв'язок *l;* z — функція, що відображає кожний вузол $n \in ID$ в конкретне значення складеного або атомарного типу, $z : n \rightarrow v$; *V* — особливий кореневий вузол графа.

Структура XML-документа, що складається з вкладених елементів-тегів добре відома, її відмінність від розглянутої вище графової моделі полягає, в основному, у трактуванні тегів і міток: в графах мітки використовуються як позначення зв'язків між елементами схем даних, і мітки не потрібні для позначення елемента, а в XML документно-орієнтованій моделі потрібно, щоб кожний (нетекстовий) елемент даних мав ідентифікуючу ознаку. Також XML транслюється в структуру даних «дерево», що є частковим випадком графової моделі.

У графовій моделі XML для слабкоструктурованих даних необхідно використовувати спеціалізовані типи атрибутів, такі як *ID, IDREF, IDREFS* [5]. Зазначені типи дають змогу організувати зберігання перехресних посилань в *XML* -елементах виду <*eid,vahie*> (<ідентифікатор елемента, значення>) і атрибутах виду < *label, eid*> (<мітка, значення>).

Існує кілька видів *RDF*-даних як графової моделі: *RDF* / *XML, N3, Turtle, RDF* / *JSON*. Опис ресурсів у вигляді *RDF* -набору даних – це трійка «суб'єкт»- «предикат»-«об'єкт», тобто для трійки:

- множини *U* (*Universal Resource Identifier, URI,* уніфікований ідентифікатор ресурсів) – елементи *f*,
- множини *B {Black nodes,* порожніх вузлів),
- множини *L {Literal, RDF*-літералів), $B \in e, L \in e$,

визначено набір (*f, e*(*f*), *e*), де *f* – «суб'єкт»; *e*(*f*) – «предикат»; *e* – «об'єкт».

Отже, Великі дані поєднують дані, представлені у різних моделях даних. Для цього повинні існувати методи їх перетворення з мінімальною втратою даних [6].

Інформаційна структура Великих даних подана на рис. 4 [8].

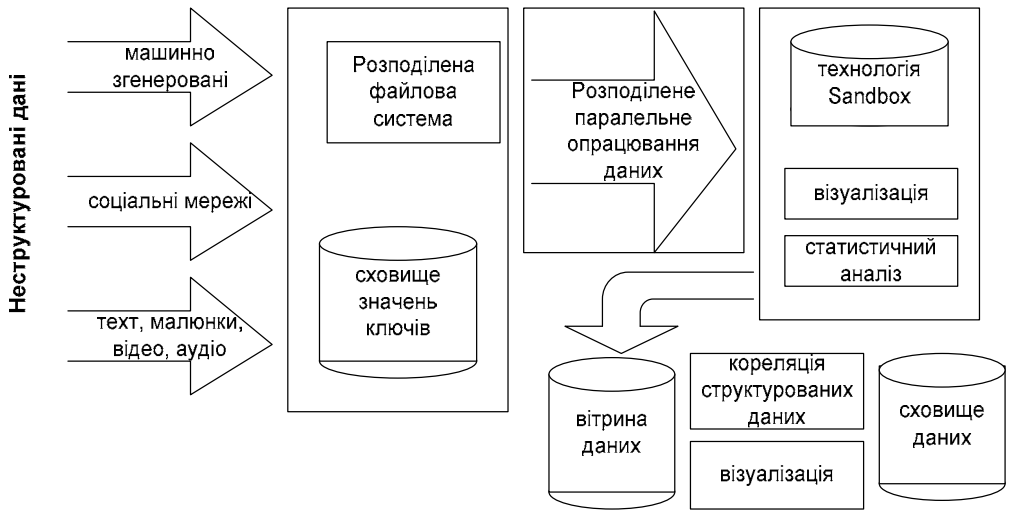

Рис. 4. Інформаційна структура Великих даних.

Висновки та перспективи подальших наукових розвідок

У статті описано структуру Великих даних. Визначено моделі асоціацій об'єктів та характеристик основних представлень даних у NoSQL. Побудовано інформаційну структуру Великих даних. Уес це стало основою для продовження досліджень і зосередженню на проблемі опрацювання різнорідних даних без їх попередньої інтеграції.

1. Pedrycz W., Chen S.-M. (eds.), Information Granularity, Big Data, and Computational Intelligence, Studies in Big Data 8, DOI: 10.1007/978-3-319-08254-7, Springer International Publishing Switzerland 2015. 2. Srinivasa, S., Bhatnagar, V. (eds.): Big data analytics. In: Proceedings of the First International Conference on Big Data Analytics BDA'2012. Lecture Notes in Computer Science, vol. 7678. Springer, New Delhi, 24–26 Dec 2012. 3. Бутакова М. А., Климанская Е. В., Янц В. И. Мера информационного подобия для анализа слабоструктурированной информации // Современные проблемы науки и образования. – 2013. – № 6; URL: http://www.science-educa- tion.ru/113-11307. 4. Chang, Fay; Dean, Jeffrey; Ghemawat, Sanjay; Hsieh, Wilson C; Wallach, Deborah A; Burrows, Michael 'Mike'; Chandra, Tushar; Fikes, Andrew; Gruber, Robert E (2006), "Bigtable: A Distributed Storage System for Structured Data", Research (PDF), Google. 5. Papakonstantinou Y. Object exchange across heterogeneous information sources / Y. Papakonstantinov, FI. Garcia-Molina, J. Widom // 11-th International Conference on Data Engineering (ICDE'05). – 2005. P. 251–261. 6. Zhou Feng, W. Hsu, Mong Li Lee. Efficient pattern discovery for sem- istructured data // 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-05). – 2005. – P. 301–309. 7. Шаховська Н. Б. Програмне та алгоритмічне забезпечення сховищ та просторів даних : монографія / Н. Б. Шаховська. – Львів: Вид-во Львівської політехніки, 2010.– 194 с. 8. Шаховська Н. Особливості моделювання просторів даних / Н. Шаховська // Вісник Нац. ун-ту «Львівська політехніка». – Львів: Вид-во Нац. ун-ту "Львівська політехніка", 2008. – № 608 : Комп'ютерна інженерія та інформаційні технології. – С. 145–154. 9. Shakhovska N., Medykovsky M., Stakhiv P. Application of algorithms of classification for uncertainty reduction/ Przeglad Elektrotechniczny. – 2013, Vol 89, 4. – P. 284–286. 10. Кут В. І. Модель консолідованих даних дистанційного навчально-консультаційного центру осіб із особливими потребами / В. І. Кут // Вісник Нац. ун-ту "Львівська політехніка". – 2014. – № 783: Інформаційні системи та мережі. – С. 120–127. 11. Кушнірецька І. І. Аналіз інформаційних ресурсів системи динамічної інтеграції слабоструктурованих даних у Web-середовищі / І. І. Кушнірецька, О. І. Кушнірецька, А. Ю. Берко // Вісник Нац. ун-ту "Львівська політехніка". Інформаційні системи та мережі. – 2014. – № 805. – С. 162–169.