**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ТАВРІЙСЬКИЙ ДЕРЖАВНИЙ АГРОТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ДМИТРА МОТОРНОГО**

**Факультет економіки та бізнесу**

**Кафедра «Маркетингу»**

|  |  |
| --- | --- |

**УПРАВЛІННЯ ВЕЛИКИМИ ДАНИМИ**

**BIG DATA MANAGEMENT**

КОНСПЕКТ ЛЕКЦІЙ

для здобувачів ступеня вищої освіти «Магістр»

зі спеціальності ***075 «Маркетинг»***

**Мелітополь, 2022р.**

**Управління великими даними (Big Data Management)**. Конспект лекцій для здобувачів ступеня вищої освіти «Магістр» зі спеціальності 075 «Маркетинг». – Мелітополь, ТДАТУ, 2022 р. – 51 с.

**Розробники:**

*Кюрчев В.М.,*д.т.н., професор кафедри експлуатації та технічного сервісу машин, ТДАТУ

*Куліш Т.В.,* к.е.н., доцент кафедри маркетингу ТДАТУ;

*Шевчук С.В.,* к.е.н., доцент кафедри обліку і економічного аналізу, Національний університет кораблебудування імені адмірала Макарова

**Рецензенти:**

*Марина Тропманн-Фрік* -д.т.н., проф., Університет прикладних наук Гамбургу (Німеччина)

*Мироненко Д.С.* к.т.н., доцент кафедри інформатики ПДТУ

Розглянуто та рекомендовано до друку на засіданні кафедри маркетингу

Протокол № *10* від « *09*» травня 2022 року

Завідувач кафедри «Маркетинг»

д.е.н, професор \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дар’я ЛЕГЕЗА

Схвалено методичною комісією факультету економіки та бізнесу для здобувачів освітнього рівня «Магістр» за спеціальності 075 «Маркетинг»

Протокол № *7* від «*20*»  *червня* 2022 року

Голова, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_ Анна КОСТЯКОВА

Схвалено Навчально-методичною радою Таврійського державного агротехнологічного університету імені Дмитра Моторного для здобувачів освітнього рівня «Магістр» за спеціальності 075 «Маркетинг»

Протокол № *1* від «*26*»  *серпня* 2022 року

Голова, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_ Олександр ЛОМЕЙКО

©Шевчук С.В., 2022 рік

ЗМІСТ

| Тема 1. Основні положення концепції Big Data | 3 |
| --- | --- |
| Тема 2. Типи даних та їх вимірювання | 7 |
| Тема 3. Управління даними. База даних та СУБД | 11 |
| Тема 4. Дизайн бази даних. Розподілена система баз даних | 19 |
| Тема 5. Основні технології та інструменти роботи з великими даними | 24 |
| Тема 6. Особливості Apache Hadoop. Компонента MapReduce | 32 |
| Список літератури | 41 |

ТЕМА 1

ОСНОВНІ ПОЛОЖЕННЯ КОНЦЕПЦІЇ BIG DATA

*1. Основні поняття та визначення.*

«Великі дані (Big Data) – позначення структурованих и неструктурованих даних величезних обсягів і значного розмаїття, що піддаються ефективній обробці програмних інструментів, які горизонтально масштабуються та з’явились у кінці 2000-х років, і альтернативних традиційних систем управління базами даних і рішенням класу рішень Business Intelligence».

Як бачимо, у цьому визначені присутні такі неоднозначні терміни, як «величезних», «значного», «ефективній» та «альтернативних». Навіть сама назва доволі суб’єктивна. Наприклад, 4 Терабайти (ємкість сучасного зовнішнього жорсткого диску для ноутбуку) – це вже великі дані чи ні? До цього визначення в широкому сенсі про «великі дані» говорять, як про соціально-економічний феномен, пов’язаний з появою технологічних можливостей аналізувати величезні масиви інформації, у деяких проблемних галузях – весь світовий об’єм даних, і трансформаційні наслідки, які з цього випливають».

Аналітики компанії IBS «весь світовий обсяг даних» оцінили такими величинами:

2003 г. — 5 ексабайтів даних (1 ЭБ = 1 млрд гігабайтів)

2008 г. — 0,18 зеттабайта (1 ЗБ = 1024 ексабайта)

2015 г. — более 6,5 зеттабайтів

2020 г. — 40–44 зеттабайта (прогноз)

2025 г. — цей об’єм збільшується ще у 10 разів.

У доповіді також підкреслюється, що більшу частину даних генерувати будуть не звичайні споживачі, а підприємства (згадаймо Промисловий інтернет речей).1

Можна користуватись і більш простим визначенням, яке цілком відповідає усталеним і більш простим визначенням, що цілком відповідають думці журналістів і маркетологів. «Великі дані – це сукупність технологій, покликаних здійснювати три операції:

1. Обробляти більші, у порівнянні зі «стандартними» сценаріями, об’єми даних.

2. Уміти працювати з даними, що швидко надходять у дуже великих об’ємах. Тобто даних не просто багато, а їх постійно стає все більше й більше.

3. Вміти працювати зі структурованими і мало структурованими даними паралельно і у різних аспектах.»2

Вважається, що ці «вміння» дозволяють виявляти приховані закономірності, що вислизають від обмеженого людського сприйняття. Це дає безпрецедентні можливості оптимізації багатьох сфер нашого життя: державного управління, медицини, телекомунікацій, фінансів, транспорту, виробництва і так далі. Не дивно, що журналісти і маркетологи так часто використовували словосполучення Big Data, що багато експертів вважають цей термін дикредитованим і пропонують від нього відмовитись.3

Більш того, у жовтні 2015 року компанія Gartner виключила Big Data з числа популярных трендів. Своє рішення аналітики компанії пояснили тим, що до складу поняття «великі дані» входить значна кількість технологій, які вже активно застосовуються на підприємствах, вони частково стосуються інших популярних сфер і тенденцій і стали повсякденним робочим інструментом.4

Визначальними характеристиками для великих даних є, окрім їх фізичного об’єму, й інші, які підкреслюють складність задачі обробки і аналізу цих даних. Набір даних VVV (volume, velocity, variety — фізичний об’єм, швидкість приросту даних і необхідність їх швидкої обробки, здатність обробляти дані різних типів) був розроблений компанією Meta Group у 2001 році з метою вказати на рівну значимість управління даними по всім трьом аспектам.

У подальшому з’явилась інтерпретація з чотирьома V (додалась veracity – достовірність), п’ятю V (viability – життєздатність і value – цінність), семи V (variability – змінність та visualization – візуалізація). Але компанія IDC, наприклад, інтерпретує саме четверте V як value (цінність), пікреслюючи економічну доцільність обробки великих об’ємів даних у відповідних умовах.5

Виходячи з вищеназваних визначень, основні принципи роботи з великими даними такі:

* Горизонтальна масштабованість. Це — базовий принцип обробки великих данных. Як вже було зазначено, великих даних з кожним днем стає все більше. Відповідно, необхідно збільшувати кількість обчислювальних вузлів, за якими розподіляються ці дані, при чому обробка має відбуватись без погіршення продуктивності
* Відмовостійкість. Цей принцип витікає з попереднього. Оскільки обчислювальних вузлів у кластері може бути багато (іноді десятки тисяч) та їх кількість, не виключено, буде збільшуватись, зростає ймовірність виходу машин з ладу. Методи роботи з великими даними мають враховувати ймовірність таких ситуацій і передбачати превентивні заходи
* Локальність даних. Оскільки дані розподілені по великій кількості обчислювальних вузлів, то, якщо вони фізично знаходяться на одному сервері, а обробляються на іншому, витрати на передачу даних можуть бути невиправдано великими. Тому обробку даних бажано проводити на тій же машині, на якій вони зберігаються

Ці принципи відрізняються від тих, які характерні для традиційних, централізованих, вертикальних моделей зберігання добре структурованих даних. Власне, для роботи з великими даними розробляються підходи і технології.

*2. Історія розвитку і джерела великих даних*

Початково у сукупність підходів і технологій включались засоби масово-паралельної обробки невизначено-структурованих даних, такі як СУБД NoSQL, алгоритми MapReduce і засоби проекту Hadoop. У подальшому до технологій великих даних почали відносити й інші рішення, що забезпечують схожі за характеристиками можливості обробки надвеликих масивів даних, а також деякі апаратні засоби.

* MapReduce — модель розподілених обчислювань у комп’ютерних кластерах, представлена компанією Google. Згідно з цією моделлю, додаток розділяється на значну кількість однакових елементарних завдань, що виконуються на вузлах кластера і потім, природнім шляхом зводяться у кінцевий результат.
* NoSQL (від англ. Not Only SQL, не лише SQL) — загальний термін для різних нереляційних баз даних і сховищ, не означає якусь конкретну технологію чи продукт. Звичайні реляційні бази даних добре підходять для досить швидких і однотипних запитів, а на складних і гнучко побудованих запитах, характерних для великих даних, навантаження перевищує розумні межі і використання СУБД стає неефективним.
* Hadoop — набор утилітів, бібліотек і фреймворків, що вільно розповсюджується, для розробки і виконання розподілених програм, які працюють на кластерах із сотень і тисяч вузлів. Вважається однією з основоположних технологій більшості даних.
* R — мова програмування для статистичної обробки даних і роботи з графікою. Широко використовується для аналізу даних і фактично став стандартом для статистичних програм.
* Апаратні рішення. Корпорації Teradata, EMC та ін. др. пропонують апаратно-програмні комплекси, призначені для обробки великих даних. Ці комплекси поставляються як готові до установки телекомунікаційні шафи, що містять кластер серверів і керівне програмне забезпечення для масово-паралельної обробки. Сюди іноді відносять апаратні рішення для аналітичної обробки в оперативній пам’яті, зокрема, апаратно-програмні комплекси Hana компанії SAP і комплекс Exalytics компанії Oracle, незважаючи на те, що така обробка початково не є масово-паралельною, а об’єми оперативної пам’яті одного вузлаобмежуються кількома терабайтами6.

Консалтингова компанія McKinsey, окрім технологій NoSQL, MapReduce, Hadoop, R, які розглядає більшість аналітиків, включає у контекст придатності для обробки великих даних також технології Business Intelligence і реляційні системи управління базами даних з підтримкою мови SQL.

Згідно звіту компанії McKinsey «Global Institute, Big data: The next frontier for innovation, competition, and productivity», дані стали таким само важливим фактором виробництва, як трудові ресурси чи виробничі активи. За рахунок використання великих даних, компанії можуть отримувати відчутні конкурентні переваги. Технології Big Data можуть бути корисними при вирішенні наступних задач:

* прогнозування ринкової ситуації
* маркетинг і оптимізація продажів
* вдосконалення продукції
* ухвалення управлінських рішень
* підвищення продуктивності праці
* ефективна логістика
* моніторинг стану основних фондів.8,9

На виробничих підприємствах великі дані генеруються також внаслідок впровадження підприємства, великі дані генеруються також внаслідок впровадження технологій Промислового інтернету речей. У ході цього процесу основні вузли і деталі станків і машин оснащуються датчиками, виконавчими пристроями, контролерами та, іноді, недорогими процесорами, здатними виробляти граничні (туманні) обчислення. В процесі виробничого процесу здійснюється постійний збір даних і, можливо, їх попередня обробка (наприклад, фільтрація). Аналітичні платформи обробляють результати у найбільш зручному для сприйняття вигляді і зберігають для подальшого використання. На основі аналізу отриманих даних робляться висновки про стан обладнання, ефективність внесених змін у технологічні процеси і т.д..

Завдяки моніторингу інформації у режимі реального часу персонал підприємства має змогу:

* скорочувати кількість простоїв
* підвищувати продуктивність обладнання
* зменшувати витрати на експлуатацію обладнання
* запобігати нещасним випадкам.

Останній пункт особливо важливий. Наприклад, оператори, що працюють на підприємствах нафтопереробної промисловості, отримують у середньому біля 1500 аварійних повідомлень на день, тобто більше одного повідомлення у хвилину. Це призводить до підвищеної втоми операторів, яким доводиться постійно приймати миттєві рішення про те, як реагує платформа на той чи інший сигнал. Але аналітична платформа може відфільтровувати другорядну інформацію, і тоді оператори отримують можливість зосередитись у першу чергу а критичних ситуаціях. Це дозволяє їм більш ефективно виявити чи попередити аварії і, можливо, нещасні випадки. В результаті підвищуються рівні надійності виробництва, промислової безпеки, готовності технологічного обладнання, відповідності нормативним вимогам.10

Крім того, за результатами аналізу великих даних можна розрахувати строки окупності обладнання, перспективи змін технологічних режимів, скорочення обслуговуючого персоналу – тобто приймати стратегічні рішення стосовно подальшого розвитку підприємства.11

ТЕМА 2

ТИПИ ДАНИХ ТА ЇХ ВИМІРЮВАННЯ

*1. Методи і техніка аналізу великих даних*

Міжнародна консалтингова компанія McKinsey, що спеціалізується на розв’язанні задач, пов’язаних зі стратегічним управлінням, виділяє 11 методів і технік аналізу, що застосовуються до великих даних.

* Методи классу Data Mining (видобуток даних, інтелектуальний аналіз даних, глибинний аналіз данних) — сукупність методів виявлення у даних раніше невідомих, нетривіальних, практично корисних знань, необхідних для прийняття рішень. До таких методів, зокрема, належать: навчання асоціативним правилам (association rule learning), класифікація (разгалуження на категорії), кластерний аналіз, регресійний аналіз, виявлення і аналіз відхилень тощо.
* Краудсорсинг — класифікація і збагачення даних силами широкого, неозначеного кола особистостей, що виконують цю роботу без вступу у трудові стосунки.
* Змішання та інтеграція даних (data fusion and integration) — набір технік, що дозволяють інтегрувати різнорідні дані з розмаїття джерел з метою проведення глибинного аналізу (наприклад, цифрова обробка сигналів, обробка природньої мови, включно з тональним аналізом).
* Машинне навчання, включаючи навчання з учителем і без учителя – використання моделей, побудованих на базі статистичного аналізучи машинного навчання для отримання комплексних прогнозів на основі базових моделей.
* Штучні нейронні мережі, мережевий аналіз, оптимізація, у тому числі генетичні алгоритми (genetic algorithm — евристичні алгоритми пошуку, що використовуються для розв’язання задач оптимізації і моделювання шляхом випадкового підбору, комбінування і варіації потрібних параметрів з використанням механізмів, аналогічних натуральному відбору у природі)
* Розпізнавання образів
* Прогнозна аналітика
* Імітаційне моделювання (simulation) — метод, що дозволяє будувати моделі, що описують процеси так, як вони би проходили у дійсності. Імітаційне моделювання модна розглядати як різновид експериментальних випробувань.
* Просторовий аналіз (spatial analysis) — клас методів, що використовують топологічну, геометричну і географічну інформацію, що вилучається із даних.
* Статистичний аналіз — аналіз часових рядів, A/B-тестування (A/B testing, split testing — метод маркетингового дослідження; при його використанні контрольна група елементів порівнюється із набором тестових груп, у яких один чи кілька показників були змінені, щоб з’ясувати, які зі змін покращують цільовий показник.
* Візуалізація аналітичних даних — подання інформації у вигляді малюнків, діаграм, з використанням інтерактивних можливостей і анімації, як для отримання результатів, так і для використання у якості вихідних даних для подальшого аналізу. Дуже важливий етап аналізу великих даних, що дозволяє показати найважливіші результати аналізу у найбільш зручному для сприйняття вигляді.7

*2. Правильний вибір алгоритмів, структури даних, вибір інструментів.*

Текст - один з найпотужніших і широко використовуваних джерел великих даних. Є електронні листи, текстові повідомлення, твіт, коментарі в соціальних медіа, обмін миттєвими повідомленнями, чати і аудіо записи, переведені в текст. Текстові дані - один з найменш структурованих джерел даних. На щастя, на сьогоднішній день вже багато зроблено для того, щоб освоїти текстові дані і використовувати їх для прийняття більш ефективних бізнес-рішень.

Аналіз тексту зазвичай починається з його розбору і осмислення різних слів, фраз і компонентів, з яких він складається. Це може бути зроблено шляхом простого підрахунку частотності вживання або за допомогою більш складних методів. Існує дисципліна під назвою «Обробка природної мови», вона часто використовується в таких аналітичних методах. Але це не є предметом обговорення в даній книзі. Інструменти для інтелектуального аналізу тексту існують в якості як компонентів основних аналітичних систем, так і автономних додатків для аналізу тексту.

В основі одних інструментів для аналізу тексту лежить підхід, при якому користувачі повинні налаштувати програмне забезпечення для ідентифікації цікавлять їх закономірностей. Інші інструменти використовують машинне навчання та інші алгоритми, що дозволяють відшукувати моделі автоматично. Кожен підхід має свої переваги і недоліки, проте їх обговорення виходить за рамки цієї книги. Ми зосередимося нема на отриманні результатів, а на їх використанні.

Після розбору і класифікації приступають до аналізу. Результати, отримані в процесі аналізу тексту, часто використовуються в якості вхідних даних для інших аналітичних процесів. Наприклад, після визначення тону електронного листа клієнта генерується змінна, яка визначає тон замовника як негативний або позитивний. Тепер цей тег - частина структурованих даних, які можна використовувати в якості вхідних для аналітичного процесу. Створення структурованих даних на основі неструктурованого тексту часто називається витяганням інформації.

Як інший приклад припустимо, що ми знаємо, про які товари клієнт залишив коментарів в процесі спілкування з нашою компанією. Ми створюємо набір змінних, які визначають товари, обговорювані клієнтом. Ці змінні також є структуровані метрики, які можна використовувати в процесі аналізу. Ці приклади показують способи збору фрагментів неструктурованих даних і створення з них релевантних і структурованих даних.

Аналіз тексту - відмінний приклад того, як абсолютно неструктуровані дані можуть бути оброблені і перетворені в структуровані, які використовуються в традиційних аналітичних процесах. Один з основних аспектів процесу приборкання великих даних полягає в застосуванні творчого підходу до процесу підготовки неструктурованих і напівструктурованих даних для подальшого використання.

Інтерпретація текстових даних насправді досить складна. Сенс наших слів змінюється в залежності від того, яке з них ми акцентуємо, а також від контексту, в який ми їх розміщуємо. При погляді на простий текст ви напевно не знаєте, на якому слові зроблений акцент, і вам часто невідомий весь контекст. Це означає, що доведеться зробити деякі припущення. Ми поговоримо про це більш детально в розділі 6.

Аналіз тексту - це одночасно мистецтво і наука, і він завжди буде мати на увазі певний рівень невизначеності. При проведенні аналізу тексту будуть виникати проблеми, викликані помилками класифікації і неоднозначністю. Це нормально. Якщо знайдена в тексті закономірність дозволяє прийняти більш ефективне рішення, то її слід використовувати. Мета аналізу тексту - поліпшити прийняті рішення, а не досягти досконалості. Текстові дані дозволяють підвищити якість прийнятих рішень і надають більш цінну інформацію, навіть не дивлячись на що міститься в них шум і неоднозначність.

Один з найпопулярніших варіантів аналізу тексту на сьогоднішній день - дослідження настрою. Аналіз настрою дозволяє вивчити загальну думку великої кількості людей, щоб зрозуміти, про що говорить ринок, що він відчуває і думає про організацію. При цьому часто використовуються дані соціальних мереж

Ми вже говорили про те, що одна зі складностей аналізу тексту полягає в тому, що слова можуть мати позитивне або негативне значення в залежності від контексту. Це необхідно брати до уваги, однак загальний настрій безлічі людей повинно бути ясно. Знаючи про те, що говорять люди в соціальних мережах або при спілкуванні з відділом обслуговування клієнтів, можна більш впевнено планувати подальші дії.

Якщо організація вловить настрою окремого клієнта, вона зможе судити про його наміри і думках. Подібно веб-даними, які допомагають визначити наміри, думка споживача про товар є цінною інформацією. Це особливо вірно, якщо споживач раніше не купував цей продукт. Аналіз настроїв показує, наскільки легко чи важко буде переконати клієнта придбати даний продукт.

Текстові дані застосовуються для розпізнавання закономірностей. Аналізуючи скарги, заявки на ремонт та інші коментарі, зроблені клієнтами, організація зможе швидше виявляти і вирішувати питання, поки вони не перетворилися на серйозні проблеми. Після виведення нового продукту на ринок і почала надходження скарг аналіз тексту допоможе визначити, з якими труднощами стикаються клієнти. Іноді вдається навіть виявити назріваючу проблему і запобігти хвилю дзвінків до відділу обслуговування клієнтів. Це дозволить реагувати набагато швидше. Організація не тільки виправить дефекти в продуктах, які будуть випущені пізніше, але і допоможе клієнтам впоратися зі складнощами, які вони відчувають сьогодні.

Важливою сферою використання текстових даних є виявлення шахрайства. У США в галузі страхування здоров'я або працездатності, наприклад, аналіз тексту можна використовувати для розбору коментарів клієнтів і заяв на отримання страхової виплати. Потім виявляються закономірності, пов'язані з випадками шахрайства, щоб оцінити ступінь ризику того чи іншого заяви. Заяви, яким притаманний більш високий ризик, слід перевіряти ретельніше. З іншого боку, деякі заяви можна перевіряти автоматично. При наявності в заяві закономірностей, термінів і фраз, що відносяться до виправданим вимогам, його розглядають як низькоризиковий і проводять через систему в прискореному режимі, а основні ресурси зосереджують на заявах з більш високим ризиком.

Переваги аналізу тексту використовуються і в сфері юриспруденції. У судових справах часто аналізуються електронне листування та інші історії повідомлень з метою виявити інформацію, яка може мати відношення до справи. Наприклад, які з електронних листів можуть містити інсайдерську інформацію? Хто надавав завідомо неправдиву інформацію при взаємодії з іншими людьми? Що особливого в природі загроз?

Застосування такого способу аналізу в судовому діловодстві називають виявленням електронних даних (eDiscovery). Всі перераховані методи аналізу можуть допомогти в розкритті злочинів. Без аналізу тексту, «вручну», було б практично неможливо перевірити всі необхідні документи. Навіть якщо така спроба була б зроблена, висока ймовірність втратити ключову інформацію в зв'язку з монотонністю заняття.

Текстові дані можуть бути затребувані у всіх галузях. Це буде один з найбільш широко використовуваних джерел великих даних. Уміння збирати, розбирати і аналізувати текст має для організацій вирішальне значення. Текст - це одне з джерел великих даних, які необхідно приборкати.

ТЕМА 3

УПРАВЛІННЯ ДАНИМИ. БАЗА ДАНИХ ТА СУБД

*1. Методи і техніка аналізу Big Data*

Основна складова великих даних - інформація, тому доцільно почати з першої групи - власників даних. Вони не обов'язково є творцями вихідної бази даних, але в їх руках знаходиться доступ до інформації та можливість її використовувати або передати на правах ліцензування іншим користувачам, які зуміють витягти з неї вигоду. ITA Software, одна з чотирьох головних мереж бронювання авіаквитків (після Amadeus, Travelport і Sabre), надала свої дані компанії Farecast для прогнозування цін на квитки, але самостійний аналіз не проводила. Чому? ITA працювала з даними виключно за їх прямим призначенням. Зрештою, продаж авіаквитків - непросте завдання, так що аналіз не входив до компетенції компанії. Крім того, у неї не було інноваційної ідеї (а значить, довелося б шукати обхідні шляхи навколо патенту Ецион). Далі, компанія вирішила не міняти положення справ з огляду на своє місця в ланцюжку створення цінності інформації. ITA мала доступ до інформації особливої ​​важливості, яка була потрібна для надання послуг, і не могла дозволити собі поставити їх під загрозу ». Замість цього вона обережно тримала дані на відстані витягнутої руки, ліцензуючи їх, але не використовуючи. В результаті ITA продала дані за безцінь. Їх основна цінність дісталася Farecast: клієнтам - у вигляді більш дешевих квитків, а співробітникам і власникам Farecast – у вигляді доходів від реклами, комісій і, врешті-решт, продажу компанії.

Ми навіть можемо спостерігати, як характерні риси неточності проникають в одну з найменш терпимих до неї областей – проектування баз даних. Для звичайних механізмів системи управління базами даних (СКБД) потрібні точні і добре структуровані дані, які не просто зберігаються, а розбиваються на «записи» з полями. Кожне поле містить інформацію конкретного типу і довжини. Наприклад, в числове поле довжиною в сім цифр неможливо записати суму, рівну десяти мільйонам і більш. А в поле для телефонних номерів не вийде ввести «недоступний». Пристосуватися до таких змін можна, тільки змінивши структуру бази даних. Ми все ще воюємо з цими обмеженнями на комп'ютерах і смартфонах, коли програмне забезпечення відмовляється приймати дані, які ми хочемо ввести. Індекси теж зумовлені, і це обмежує можливості пошуку. А щоб додати новий індекс, його створюють з нуля, витрачаючи час. Звичайні реляційні бази даних призначені для роботи в області розріджених даних, які можна і слід ретельно перевіряти. У такій області питання, на які потрібно відповісти за допомогою даних, відомі спочатку, тому база даних служить саме для ефективної відповіді на них. Однак ця точка зору на збереження та аналіз даних все більше розходиться з реальністю. Тепер в нашому розпорядженні є великі обсяги даних різного типу і якості. Дані рідко вписуються в певні категорії, відомі спочатку. І питання, на які ми хотіли б отримати відповідь, теж часто виникають тільки в процесі збору даних або роботи з ними. Ці реалії призвели до створення нових структур баз даних. Старі принципи створення записів і зумовлених полів, що відображають чітко задану ієрархію інформації, залишилися в минулому. Довгий час найпоширенішою мовою доступу до бази даних був SQL ( «структурований мову запитів»). Сама назва говорить про його жорсткості. Але в останні роки відбувся перехід в сторону так званої технології NoSQL, при якій в базах даних не потрібно зумовлена структура записів. Допускаються дані різного типу і розміру. При цьому вони все так само доступні для пошуку. Безлад, який допускається в структурі таких баз даних, компенсується тим, що для їх зберігання і обробки потрібно більше ресурсів. І все ж, з огляду на різке падіння витрат на зберігання і обробку, цей компроміс ми можемо собі дозволити.

*2. База даних та СУБД*

[База даних](http://lib.mdpu.org.ua/e-book/vstup/glos.htm#%D0%91%D0%B0%D0%B7%D0%B0%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) (БД) – упорядкований набір логічно взаємопов'язаних даних, що використовується спільно, та призначений для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система управління БД.

[Система управління базами даних](http://lib.mdpu.org.ua/e-book/vstup/glos.htm#%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%20%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F%20%D0%B1%D0%B0%D0%B7%D0%BE%D1%8E%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) (СУБД) - це комплекс програмних і мовних засобів, необхідних для створення баз даних, підтримання їх в актуальному стані та організації пошуку в них необхідної інформації. Централізований характер управління даними в базі даних передбачає необхідність існування деякої особи (групи осіб), на яку покладаються функції адміністрування даними, що зберігаються в базі.

За способом доступу до даних бази даних поділяються на бази даних з локальним доступом і бази даних з віддаленим (мережевим) доступом.

Системи централізованих баз даних з мережевим доступом припускають різні архітектури подібних систем:

файл-сервер;

клієнт-сервер.

Файл-сервер. [Архітектура систем](http://lib.mdpu.org.ua/e-book/vstup/glos.htm#%D0%90%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B8) БД з мережевим доступом передбачає виділення однієї з машин мережі в якості центральної (сервер). На такій машині зберігається спільно використовувана централізована БД. Усі інші машини мережі виконують функції робочих станцій, за допомогою яких підтримується доступ користувальницької системи до централізованої бази даних. Файли бази даних відповідно до призначених для користувача запитів передаються на робочі станції, де в основному і проводиться обробка. При великій інтенсивності доступу до одних і тих же даних продуктивність інформаційної системи падає. Користувачі можуть створювати також на робочих станціях локальні БД, які використовуються ними монопольно.

Клієнт-сервер. У цій концепції мається на увазі, що крім зберігання централізованої бази даних центральна машина (сервер бази даних) повинна забезпечувати виконання основного обсягу обробки даних. Запит на дані, який видається клієнтом (робочою станцією), породжує пошук і вилучення даних на сервері. Витягнуті дані (але не файли) транспортуються по мережі від сервера до клієнта. Специфікою архітектури клієнт-сервер є використання мови запитів SOL.

Історично системи управління базами даних орієнтувалися на вирішення завдань, пов'язаних у першу чергу з транзакційною обробкою структурованої інформації. Безумовно, найкращим, перевіреним часом рішенням тут була і залишається реляційна модель СУБД. Однак в останні роки область застосування баз даних незмінно розширювалася. З одного боку, потрібно керувати більш широким набором форматів даних, переходячи до вирішення спільних проблем управління корпоративною інформацією. З іншого - саме СУБД беруть на себе основні функції інтеграції даних і додатків корпоративних систем. (За даними Gartner Group, інформаційні відділи підприємств витрачають до 40% свого бюджету на вирішення завдань інтеграції діючих компонентів баз даних.) Саме цим пояснюється активний інтерес до обговорення архітектурних принципів і можливостей реалізації баз даних різних моделей – пост реляційних, об'єктно-реляційних, XML.

Якщо постаратися класифікувати існуючі області застосування баз даних, а так само оцінити перспективи їхнього розвитку в даний час, то можна отримати приблизний список найбільш поширених класів:

* документографічні й документальні застосовуються у всіх базах органів влади та управління;
* бази даних з промислової, будівельної та сільськогосподарської продукції;
* бази даних з економічної та кон'юнктурної інформації (статистична, кредитно-фінансова, зовнішньоторговельна);
* фактографічні бази соціальних даних, які включають відомості про населення і про соціальні середовища;

Останнім часом утворилися нові важливі області застосування баз даних, і кожна з них представляє принципово нове середовище, до якого необхідно адаптувати технології СУБД. Ці області отримали на ринку назви інтелектуального аналізу даних (data mining), сховищ даних (data warehousing), репозитаріїв даних (data repository).

Інтелектуальний аналіз даних. Ідея інтелектуального аналізу даних (data mining), тобто добування інформації з величезних масивів даних, накопичених зовсім для інших цілей, викликає сьогодні підвищений ентузіазм. Наприклад, авіакомпанії домагаються оптимального заповнення рейсів за рахунок аналізу накопичених раніше даних про резервування квитків. Можна навести одну історію про те, як була виявлена несподівана кореляція між покупками пива і покупками серветок у післяобідній період. Власник магазину наблизив один до одного відділи, які торгують пивом і серветками, а між ними помістив ще прилавки з картопляними чіпсами. У результаті збільшилися продажі усіх трьох видів товару.

Із запитами, характерними для систем інтелектуального аналізу даних, пов'язана низка незвичайних проблем. Вони включають, як правило, агрегацію величезних обсягів даних. Вони мають нерегламентований характер; їх формулюють особи, відповідальні за прийняття рішень, коли їм необхідно виявити будь-які неочевидні взаємозв'язки.

У додатках, пов'язаних, наприклад, з торгівлею цінними паперами, дуже важливий малий час відповіді. Суть проблеми полягає тут у тому, щоб скоротити загальний час, необхідний для написання, налагодження та виконання запиту.

Досить часто користувач не в змозі точно сформулювати запит - йому просто потрібно виявити "що-небудь цікаве".

Таким чином, з видобутком даних пов'язані такі дослідницькі напрями:

* методи оптимізації складних запитів, які включають, наприклад, агрегацію та групування;
* методи підтримки "багатовимірних" запитів, що відносяться до даних, організованих у вигляді "куба", в осередках якого знаходяться потрібні дані;
* методи оптимізації використання третинної пам'яті;
* мови запитів дуже високого рівня, а також інтерфейси для підтримки користувачів, які не є експертами, і яким потрібні відповіді на нерегламентовані запити.

Сховища даних. У сховищі даних накопичуються дані з однієї або більше баз даних. Існує безліч потенційних застосувань, а також підходів до організації сховищ даних. Наприклад, великий магазин може підтримувати сховище даних на основі транзакційних даних про касові операції для цілей видобутку даних. У сховищі даних може зберігатися інформація з багатьох баз даних для використання в надзвичайних ситуаціях. Наприклад, в єдиному сховищі даних підтримуються відомості про цивільну інфраструктуру (дороги, мости, трубопроводи тощо), оскільки, наприклад, після землетрусу навряд чи вдасться отримати цю інформацію з міст, що знаходяться поблизу епіцентру. Ще один приклад - використання сховища даних як "матеріалізованого уявлення" інтегрованої інформації. Альтернативою медіаторних систем, які дають цілісне уявлення даних, витягнутих з безлічі джерел, можуть служити сховища даних, що забезпечують фізичне зберігання інтегрованих даних. На відміну від сховищ, медіатори надають інформацію, розсилаючи запити декільком джерелам, подібно до того, як це відбувається при реалізації уявлень.

Деякі дослідницькі проблеми, що стосуються сховищ даних, збігаються з тими, які характерні для інтеграції даних у цілому, але є і деякі специфічні проблеми.

Інструменти для створення насосів даних (data pump), тобто модулів, що функціонують за середовищем джерел даних і поставляють у сховище ті зміни, які істотні з точки зору сховища; при цьому дані мають транслюватися у відповідності з глобальною моделлю і схемою сховища.

Методи "чистки даних" (data scrubbing), які забезпечують узгодження даних, видалення елементів, що відповідають різним уявленням одного й того ж об'єкта (наприклад "Sally Tones" і "SA Tones"), а також видалення неправдоподібних значень.

Засоби для створення і підтримки метасловника, який інформує користувачів про способи отримання даних.

Репозитарії.

Програми, що відносяться до категорії репозитаріїв, характеризуються тим, що вони призначені для зберігання і управління як даними, так і метаданими, тобто інформацією про структуру даних. Приклади репозитаріїв - бази даних для підтримки комп'ютерного проектування, включаючи CASE (системи проектування програмного забезпечення), а також системи управління документами. Відмінна риса цих систем - часті зміни метаданих, характерні для будь-якого середовища проектування.

У репозитарії необхідно підтримувати безліч уявлень однієї й тієї ж або схожої інформації. Наприклад, програмний модуль має уявлення у вигляді вихідного коду, об'єктного коду, проміжного коду, готової програми, таблиць використань/визначень, документації. Зв'язки між усіма цими уявленнями повинні відстежуватися репозитарієм так, щоб зміни в одному з них автоматично поширювалися на інші уявлення того ж об'єкта.

Репозитарії повинні підтримувати поняття версій (моментальних знімків елементів даних, що змінюються в часі) і конфігурацій (версіонних колекцій версій). Наприклад, різні релізи програмної системи будуть зазвичай формуватися як конфігурації з певних версій файлів вихідного коду.

Сховище повинне підтримувати еволюцію структури інформації та її метаданих таким чином, щоб при додаванні нових властивостей даних або нових зв'язків не була потрібна повна перекомпіляція.

*3. Інструменти обробки великих даних*

Коротка характеристика деяких [СУБД](http://lib.mdpu.org.ua/e-book/vstup/glos.htm#%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%20%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F%20%D0%B1%D0%B0%D0%B7%D0%BE%D1%8E%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)

MySQL - вільна система управління базами даних. MySQL є власністю компанії Oracle Corporation, що отримала її разом з поглиненою Sun Microsystems, яка здійснює розробку і підтримку додатку. Розповсюджується під GNU General Public License і під власною комерційною ліцензією, на вибір. Крім цього розробники створюють функціональність на замовлення ліцензійних користувачів, саме завдяки такому замовленню майже в найраніших версіях з'явився механізм реплікації.

Цю систему управління базами даних з відкритим кодом було створено як альтернатива комерційним системам. MySQL із самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL - одна з найпоширеніших систем управління базами даних. Вона використовується, у першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL є рішенням для малих і середніх додатків. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або віддалені клієнти, проте до дистрибутиву входить бібліотека внутрішнього сервера, що дозволяє включати MySQL до автономних програм. Вихідні коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNІХ-системах, де є підтримка багатонитевості, що підвищує продуктивність системи в цілому.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі й GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць. MySQL характеризується великою швидкістю, стійкістю і простотою використання.

Для некомерційного використання MySQL є безкоштовною. Можливості сервера MySQL:

* простота у встановленні та використанні;
* підтримується необмежена кількість користувачів, що одночасно працюють із БД;
* кількість рядків у таблицях може досягати 50 млн.;
* висока швидкість виконання команд;
* наявність простої та ефективної системи безпеки.

PostgreSQL - об'єктно-реляційна система управління базами даних. Є альтернативою як комерційним СУБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СУБД з відкритим кодом (MySQL, Firebird, SQLite).

Порівняно до інших проектів з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СУБД та впроваджувати у неї найновіші досягнення.

СУБД Oracle – це найпотужніший програмний комплекс, що дозволяє створювати додатки будь-якої складності. Ядром цього комплексу є база даних, що зберігає інформацію, кількість якої за рахунок наданих засобів масштабування практично безмежна. З високою ефективністю працювати з цією інформацією одночасно може практично будь-яка кількість користувачів (за наявності достатніх апаратних ресурсів), не проявляючи тенденції до зниження продуктивності системи при різкому збільшенні їхньої кількості.

Механізми масштабування в СУБД Oracle останньої версії дозволяють безмежно збільшувати потужність і швидкість роботи сервера Oracle і своїх додатків, просто додаючи нові й нові вузли кластеру. Це не вимагає зупинки працюючих додатків, не вимагає переписування старих додатків, розроблених для звичайної одномашинної архітектури. Крім того, вихід з ладу окремих вузлів кластера також не призводить до зупинки програми.

Вбудовування до СУБД Oracle JavaVM, повномасштабної підтримки серверних технологій (Java Server Pages, Java-сервлети, модулі Enterprise JavaBeans, інтерфейси прикладного програмування CORBA), призвели до того, що Oracle на сьогоднішній день де-факто є стандартом СУБД для Internet.

Ще однією складовою успіху СУБД Oracle є те, що вона поставляється практично для всіх існуючих на сьогодні операційних систем. Працюючи під Sun Solaris, Linux, Windows або на інший операційній системі з продуктами Oracle не буде виникати ніяких проблем у роботі. СУБД Oracle однаково добре працює на будь-якій платформі. Таким чином, компаніям, які розпочинають роботу з продуктами Oracle не доводиться міняти мережеве оточення. Існує лише невелика кількість відмінностей при роботі з СУБД, обумовлених особливостями тієї або іншої операційної системи. У цілому ж це завжди та ж сама безпечна, надійна і зручна СУБД Oracle.

Microsoft SQL Server - система управління реляційними базами даних, розроблена корпорацією Microsoft. Основна використовувана мова запитів - Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів (SQL) із розширеннями. Використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства, конкурує з іншими СУБД у цьому сегменті ринку.

При взаємодії з мережею Microsoft SQL Server і Sybase ASE використовують протокол рівня додатків під назвою Tabular Data Stream (TDS, протокол передачі табличних даних). Протокол TDS також був реалізований у проекті FreeTDS з метою забезпечити різні додатки можливістю взаємодії з базами даних Microsoft SQL Server і Sybase.

Для забезпечення доступу до даних Microsoft SQL Server підтримує Open Database Connectivity (ODBC) - інтерфейс взаємодії додатків з СУБД. SQL Server надає можливість підключення користувачів через веб-сервіси, що використовують протокол SOAP. Це дозволяє клієнтським програмам, не призначеним для Windows, кросплатформно з'єднуватися з SQL Server.

Microsoft Office Access або просто Microsoft Access - реляційна СУБД корпорації Microsoft. Має широкий спектр функцій, включаючи зв'язані запити, зв'язок із зовнішніми таблицями і базами даних. Завдяки вбудованій мові VBA, в самому Access можна писати застосунки, що працюють з базами даних.

Вони можуть викликати скрипти мовою VBA, тому MS Access дозволяє розробляти програми і бази даних практично "з нуля" або написати оболонку для зовнішньої БД. MS Access є файл-серверною СУБД і тому застосовується лише до маленьких додатків. Відсутній ряд механізмів, необхідних у багатокористувацьких БД, таких, наприклад, як тригери.

Істотно розширює можливості MS Access з написання додатків механізм зв'язку з різними зовнішніми СУБД: "зв'язані таблиці" (зв'язок з таблицею СУБД) і "запити до сервера" (запит на діалекті SQL, який "розуміє" СУБД). Також MS Access дозволяє будувати повноцінні клієнт-серверні додатки на СУБД MS SQL Server. При цьому є можливість поєднати з притаманною MS Access простотою інструменти для управління БД і засоби розробки.

Принципи створення баз даних зазнали серйозних змін, з тих пір, як в 1962 році була реалізована перша з них. Спочатку застосовувалися навігаційні БД, до яких відносяться типи з деревовидної і мережевою структурою.

З плином часу реляційні бази даних замінили мережеві БД, які тепер використовуються тільки для спеціальних завдань. Графові системи, в свою чергу, є свого роду подальшим розвитком реляційних БД. Вони використовують графіки для відображення та зберігання мережевої інформації. Протягом декількох десятиліть на ринку домінували три основні виробники СУБД (RDBMS): Oracle, IBM і Microsoft. Однак поточна динаміка ринку привела до змін, які привели в рух ринок і створили більш широку конкуренцію.

За інформацією на перший квартал 2020 року, в світі налічується більше 250 різних продуктів, які пропонують реляційні або графові типи організації БД. Причому більше 50 продуктів використовують нові типи – NoSQL, куди відносяться як хмарні рішення, так і БД з автономної структурою і підтримкою машинного інтелекту.

ТЕМА 4

ДИЗАЙН БАЗИ ДАНИХ. РОЗПОДІЛЕНА СИСТЕМА БАЗ ДАНИХ

*1. Дизайн бази даних*

Для управління базами даних потрібно певний софт. Без нього БД – це просто файли. Які, до того ж, не можна відкрити звичайними програмами, на зразок блокнота (найчастіше). Цей софт, необхідний для маніпуляцій з базами даних, називається СУБД – Системи Управління Базами Даних. Вони дозволяють створювати бази, структурувати, змінювати / видаляти / додавати дані, захищати їх і так далі.

Головним завданням [БД](http://lib.mdpu.org.ua/e-book/vstup/glos.htm#%D0%91%D0%B0%D0%B7%D0%B0%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) є гарантоване збереження значних обсягів інформації та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином БД складається з двох частин: збереженої інформації та системи управління нею. З метою забезпечення ефективності доступу записи даних організовують як множину фактів (елемент даних). Існує величезна кількість різновидів баз даних, що відрізняються за критеріями (наприклад, в Енциклопедії технологій баз даних визначаються понад 50 видів БД).

Відзначимо тільки основні класифікації. Класифікація БД за моделлю даних:

ієрархічні,

мережеві,

реляційні,

об'єктні,

об'єктно-орієнтовані,

об'єктно-реляційні.

Класифікація БД за технологією фізичного зберігання:

БД у вторинній пам'яті (традиційні);

БД в оперативній пам'яті (in-memory databases);

БД у третинній пам'яті (tertiary databases).

Класифікація БД за вмістом:

географічні.

історичні.

наукові.

мультимедійні.

Класифікація БД за ступенем розподіленості:

централізовані (зосереджені);

розподілені.

Окреме місце в теорії та практиці займають просторові (англ. spatial), тимчасові, або темпоральні (temporal) і просторово-часові (spatial-temporal) БД.

Ієрархічні бази даних можуть бути представлені як дерево, що складається з об'єктів різних рівнів. Верхній рівень займає один об'єкт, другий - об'єкти другого рівня і т.д.

Між об'єктами існують зв'язки, кожен об'єкт може включати в себе декілька об'єктів більш низького рівня. Такі об'єкти перебувають у відношенні предка (об'єкт більш близький до кореня) до нащадка (об'єкт більш низького рівня), при цьому можлива ситуація, коли об'єкт-предок не має нащадків або має їх декілька, тоді як у об'єкта-нащадка обов'язково тільки один предок. Об'єкти, що мають загального предка, називаються близнюками.

Мережеві бази даних подібні до ієрархічних, за винятком того, що в них є покажчики в обох напрямках, які з'єднують споріднену інформацію.

До основних понять мережевої моделі бази даних відносяться: рівень, елемент (вузол), зв'язок.

Вузол - це сукупність атрибутів даних, що описують деякий об'єкт. На схемі ієрархічного дерева вузли представляються вершинами графа. У мережній структурі кожен елемент може бути пов'язаний з будь-яким іншим елементом.

Незважаючи на те, що ця модель вирішує деякі проблеми, пов'язані з ієрархічною моделлю, виконання простих запитів залишається досить складним процесом.

Також, оскільки логіка процедури вибірки даних залежить від фізичної організації цих даних, то ця модель не є повністю незалежною від програми. Іншими словами, якщо необхідно змінити структуру даних, то потрібно змінити і додаток.

Реляційна модель орієнтована на організацію даних у вигляді двовимірних таблиць. Кожна реляційна таблиця являє собою двовимірний масив і має наступні властивості:

кожен елемент таблиці - один елемент даних;

всі осередки в стовпчику таблиці однорідні, тобто всі елементи в стовпчику мають однаковий тип (числовий, символьний тощо);

кожен стовпчик має унікальне ім'я;

однакові рядки в таблиці відсутні;

порядок проходження рядків і стовпчиків може бути довільним.

Об'єктна СУБД ідеально підходить для інтерпретації складних даних, на відміну від реляційних СУБД, де додавання нового типу даних досягається ціною втрати продуктивності або за рахунок різкого збільшення термінів і вартості розробки додатків. Об'єктна база, на відміну від реляційної, не вимагає модифікації ядра при додаванні нового типу даних. Новий клас і його екземпляри просто надходять у зовнішні структури бази даних. Система управління ними залишається без змін.

Об'єктно-орієнтована база даних (ООБД) - база даних, в якій дані оформлені у вигляді моделей об'єктів, що включають прикладні програми, які управляються зовнішніми подіями. Результатом поєднання можливостей (особливостей) баз даних і можливостей об'єктно-орієнтованих мов програмування є об'єктно-орієнтовані системи управління базами даних (ООСУБД). ООСУБД дозволяють працювати з об'єктами баз даних також, як з об'єктами у програмуванні в об'єктно-орієнтованих мовах програмування. ООСУБД розширює мови програмування, прозоро вводячи довготривалі дані, управління паралелізмом, відновлення даних, асоційовані запити й інші можливості.

Об'єктно-орієнтовані бази даних звичайно рекомендовані для тих випадків, коли потрібна високопродуктивна обробка даних, які мають складну структуру.

Система, яка забезпечує об'єктну інфраструктуру і набір реляційних розширювачів, називається "об'єктно-реляційною".

Об'єктно-реляційні системи поєднують переваги сучасних об'єктно-орієнтованих мов програмування з такими властивостями реляційних систем як множинні представлення даних і високорівневі непроцедурні мови запитів.

За технологією обробки даних бази даних поділяються на централізовані й розподілені.

Централізована база даних зберігається у пам'яті однієї обчислювальної системи. Якщо ця обчислювальна система є компонентом мережі ЕОМ, можливий розподілений доступ до такої бази. Такий спосіб використання баз даних часто застосовують у локальних мережах ПК.

Розподілена база даних складається з декількох, можливо пересічних або навіть дублюючих одна одну частин, які зберігаються в різних ЕОМ обчислювальної мережі. Робота з такою базою здійснюється за допомогою системи управління розподіленою базою даних (СУРБД).

*2. Розподілена система баз даних*

Розподілені бази даних (РБД) - сукупність логічно взаємопов'язаних баз даних, розподілених у комп'ютерній мережі.

РБД складається з набору вузлів, пов'язаних комунікаційної мережею, в якій:

* кожен вузол - це повноцінна БД сама по собі;
* вузли взаємодіють між собою таким чином, що користувач будь-якого з них може отримати доступ до будь-яких даних у мережі так, як ніби вони знаходяться на його власному сайті.

Кожен вузол сам по собі є системою бази даних. Будь-який користувач може виконати операції над даними на своєму локальному вузлі точно так само, як якщо б цей вузол зовсім не входив у розподілену систему. Розподілену систему баз даних можна розглядати як партнерство між окремими локальними БД на окремих локальних вузлах.

Фундаментальний принцип створення розподілених баз даних («правило 0»): Для користувача розподілена система повинна виглядати так само, як нерозподілений система.

Фундаментальний принцип має наслідком певні додаткові правила чи мети. Таких цілей всього дванадцять:

Локальна незалежність. Вузли в розподіленій системі повинні бути незалежні, або автономні. Локальна незалежність означає, що всі операції на вузлі контролюються цим вузлом.

Відсутність опори на центральний вузол. Локальна незалежність припускає, що всі вузли в розподіленій системі повинні розглядатися як рівні. Тому не повинно бути ніяких звернень до «центрального» або «головному» вузлу з метою отримання певної централізованого сервісу.

Безперервне функціонування. Розподілені системи повинні надавати більш високу ступінь надійності і доступності.

Незалежність від розташування. Користувачі не повинні знати, де саме дані зберігаються фізично і повинні діяти так, як якщо б всі дані зберігалися на їх власному локальному вузлі.

Незалежність від фрагментації. Система підтримує незалежність від фрагментації, якщо дана зміна-відношення може бути розділена на частини чи фрагменти при організації її фізичного зберігання. У цьому випадку дані можуть зберігатися в тому місці, де вони найчастіше використовуються, що дозволяє досягти локалізації більшості операцій та зменшення мережевого трафіку.

Незалежність від реплікації. Система підтримує реплікацію даних, якщо дана збережена мінлива-ставлення - або в загальному випадку даний фрагмент даної збереженої змінної-відносини - може бути представлена кількома окремими копіями або репліками, які зберігаються на декількох окремих вузлах.

Обробка розподілених запитів. Суть в тому, що для запиту може знадобитися звернення до кількох вузлів. У такій системі може бути багато можливих способів пересилання даних, що дозволяють виконати розглянутий запит.

Управління розподіленими транзакціями. Існує 2 головних аспекти управління транзакціями: управління відновленням і керування паралельністю обробки. Що стосується управління відновленням, то щоб забезпечити атомарность транзакції в розподіленому середовищі, система повинна гарантувати, що всі безліч відносяться до даної транзакції агентів (агент - процес, який виконується для даної транзакції на окремому компоненті) або зафіксувало свої результати, або виконало скидання. Що стосується управління паралельністю, то воно в більшості розподілених систем базується на механізмі блокування, точно так, як і в нерозподілених системах.

Апаратна незалежність. Бажано можливість запускати одну і ту ж БД на різних апаратних платформах і, більше того, домогтися, щоб різні машини брали участь в роботі розподіленої системи як рівноправні партнери.

Незалежність від операційної системи. Можливість функціонування БД під різними операційними системами.

Незалежність від мережі. Можливість підтримувати багато принципово різних вузлів, що відрізняються обладнанням та операційними системами, а також ряд типів різних комунікаційних мереж.

Незалежність від типу БД. Необхідно, щоб екземпляри БД на різних вузлах всі разом підтримували один і той же інтерфейс, і зовсім необов'язково, щоб це були копії однієї і тієї ж версії БД.

Як правило, розрізняють два рівні абстракції представлення даних у вигляді інформаційної та фізичної моделей. Користувач мало звертає увагу на організацію фізичного зберігання інформації - його цікавить логічне представлення даних.

Концепція реляційної моделі бази даних призначена забезпечити незалежність представлення та опису даних від прикладних програм. В основі цієї моделі лежать поняття відношення (relations), подане у вигляді таблиці з дотриманням деяких обмежувальних умов. Основні взаємопов’язані поняття фізичного, спеціального прикладного та математичного рівнів побудови реляційної бази даних та їх взаємовідношення показані в таблиці, в рядках якої знаходяться еквівалентні поняття:

Основні поняття реляційної бази даних та їх взаємозв’язок.

|  | Фізичний рівень | Спеціальний прикладний рівень | Математичний рівень |
| --- | --- | --- | --- |
| 1 | Файл | Таблиця | Відношення |
| 2 | Запис | Рядок | Кортеж |
| 3 | Поле | Стовпець | Атрибут |

Таблиця зрозуміла оглядово і звична для людей. Оригінальність такого підходу полягає в застосовані до відношень впорядкованої системи операцій, що дозволяє отримувати складені відношення з бази (виводити, обчислювати, проводити арифметичні операції). Використання відношень дозволяє ділити інформацію на таку, що зберігається постійно і таку, яка отримується в результаті визначених перетворень над постійною.

ТЕМА 5

ОСНОВНІ ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ РОБОТИ З ВЕЛИКИМИ ДАНИМИ

*1. Загальні підходи обробки великих даних*

До найбільш поширених підходів обробки великих даних є:

* SQL
* NoSQL
* MapReduce
* Hadoop
* SAP HANA

Модель проштовхування SQL SQL - рідна мова для MPP-системи, і він відповідає широкому спектру вимог. Це особливо стосується агрегування, з'єднання і перетворення даних. Багато основні завдання, пов'язані з підготовкою даних, можуть бути переведені на мову SQL користувачем. Або аналітичний інструмент може генерувати SQL-код від імені користувача та «проштовхнути» його в базу даних. Код SQL також легко генерується багатьма аналітичними алгоритмами, що володіють досить простою логікою. До цієї категорії відносяться лінійна регресія, логістична регресія і дерева рішень. Аналітичні інструменти часто переводять логіку моделі на мову SQL. Іноді користувачі самостійно пишуть SQL-сценарій після створення моделі. У будь-якому випадку підготовка даних або процеси скорингу в кінцевому рахунку виконуються тільки з використанням SQL.

Функції, визначені користувачем (UDF), - відносно нова особливість реляційних баз даних. Можливості UDF виходять за рамки можливостей мови SQL. Певні користувачем функції розширюють функціональність мови SQL, дозволяючи користувачеві визначити логіку завдання, яка виконується так само, як і рідна SQL-функція. Запит загального обсягу продажів по кожному споживачеві може бути записаний у такий спосіб: «Select Customer, Sum (Sales) ...» При використанні певних користувачем функцій запит на проведення скорингу втрат може бути записаний так: «Select Customer, Attrition\_Score ...» В останньому прикладі «Attrition\_Score» - певна користувачем функція, наявна в системі бази даних. UDF застосовує до даних необхідну логіку, і ця логіка може бути значно складнішою, ніж в разі використання тільки мови SQL. Функції, визначені користувачем, пишуться на таких мовах, як C ++ або Java. В результаті в них можуть бути вбудовані деякі процедурні можливості мови. Це розширює ядро SQL, яке не має таких можливостей. Складність полягає в тому, що багато професійних аналітики не вміють писати програми на цих мовах. І тоді на допомогу приходять аналітичні інструменти, які автоматично генерують відповідні певні користувачем функції і завантажують їх у базу даних, готуючи до використання. вбудовані процеси Концепція вбудованих процесів – абсолютно новий спосіб обробки в базі даних. Вбудований процес являє собою ще більш високий рівень інтеграції, ніж певні користувачем функції. Останні мають на увазі компіляцію коду у вигляді нових функцій, які можуть бути викликані з запиту SQL, як і будь-які інші. З точки зору користувача аналітична функція буде вести себе так само, як вихідний код аналітичного інструменту, і ефективно працювати в базі даних, проте вона більше не представлена у вигляді коду аналітичного інструменту. По суті, аналітичний інструмент перевів те, що йому було необхідно, на мову бази даних. Проте вбудований процес являє собою варіант роботи движка аналітичного інструменту в самій базі даних. Таким чином, вбудований процес може запускати програми безпосередньо всередині бази даних. Вбудовані процеси використовують код аналітичного інструменту, який був поміщений в движок бази даних. Коли необхідно виконати фрагмент коду та використовувати переваги паралелізму системи, цей фрагмент передається аналітичних інструментів, які працюють на кожному з процесорів бази даних. У перекладі немає необхідності. Цей метод вимагає найменших змін у вихідному коді, проте реалізувати його складно. Постачальники інструментів і баз даних тільки починають розширювати сферу застосування вбудованих процесів. У найближчі роки цей метод, ймовірно, стане найпоширенішим.

Причина, по якій масивно-паралельна система може виявитися настільки корисною в сфері передової аналітики, полягає в тому, що бо'льшая частина незручностей, пов'язаних з обробкою, доводиться на етап підготовки даних, яка включає з'єднання, агрегування, деривацію і перетворення. З'єднання має на увазі об'єднання різних джерел даних для збору всієї необхідної для аналізу інформації. Агрегація передбачає комбінування відомостей з декількох записів. Один із прикладів - обчислення загальної та середньої обсягу продажів, що припадає на одного клієнта, з кількох транзакцій. До деривації і перетворенню відносяться такі дії, як обчислення співвідношень, наприклад обсяг продажів за одну транзакцію, і застосування функцій, наприклад логарифм або квадратний корінь, до змінних для створення додаткових змінних. Логіка більшості завдань, пов'язаних з підготовкою даних, відносно проста. Для вирішення саме таких задач і призначені реляційні бази даних і їх рідна мова, відомий як мову структурованих запитів (structured query language - SQL). Сучасний SQL може впоратися з більшістю, якщо не з усіма, завдань підготовки даних, необхідними для аналізу. Саме використання SQL поклало початок розвитку аналітики, вбудованої в базу даних, в MPP-системах. Аналітики просто помістили завдання в базу даних, написавши їх на SQL, замість того щоб писати їх на аналітичному мовою, що вимагало б вилучення інформації з бази даних. Ще 10 років тому SQL мав деякі обмеження, коли мова йшла про певні складних обчисленнях, необхідних для підтримки передової аналітики. Сьогодні це мова набагато більш потужний. Старе правило, згідно з яким при роботі з конкретною рядком запит не враховує дані в інших рядках, вже не діє. Наприклад, існують SQL-функції, звані «віконними» (windowed aggregates), які при обробці конкретної рядки дозволяють запитом врахувати дані, що знаходяться в іншому місці. За допомогою таких функцій запит, наприклад, може дізнатися, чи є дана транзакція першої або останньої для клієнта, і робити різні дії при обробці даних. Така функціональність дозволяє SQL вирішувати безліч додаткових складних завдань обробки, які ставлять кошти передової аналітики в процесі підготовки даних. До того як SQL обзавівся більш широкою функціональністю, для обробки інформації необхідно було отримувати її з бази даних. На щастя, все це в минулому, оскільки мова SQL значно еволюціонував. Більшість операцій, пов'язаних з підготовкою даних, тепер можуть бути зроблені за допомогою SQL прямо в базі даних. Багато фахівців до цих пір пишуть код на SQL замість використання аналітичних інструментів, проте в даний час існує і безліч вбудованих варіантів.

NoSQL (not only SQL, не тільки SQL)- термін, що позначає ряд підходів, спрямованих на реалізацію сховищ баз даних, що мають суттєві відмінності від моделей, що використовуються в традиційних реляційних СКБД з доступом до даних засобами мови SQL. Застосовується до баз даних, в яких робиться спроба вирішити проблеми масштабованості і доступності за рахунок атомарності і узгодженості даних.

Основні риси:

базова доступність – запити гарантовано завершується (успішно чи безуспішно) NoSQL дає розробникам більше свободи, прискорює їх роботу, а також дозволяє змінювати схеми і запити відповідно до вимог до даних. Зберігання даних в вигляді агрегатів допомагає швидко вносити ітеративні поліпшення, що не проектуючи схему заздалегідь;

гнучкий стан – стан системи може змінюватися з часом, навіть без введення нових даних, для досягнення узгодження даних; завдяки можливості реагувати на незаплановані ситуації бази даних NoSQL підходять для інтенсивних циклів випуску програмного забезпечення, а також для швидкої і гнучкої розробки додатків.

узгодженість в кінцевому рахунку – дані можуть бути деякий час неузгодженими, але приходять до узгодження через деякий час, бази даних NoSQL надають привабливі експлуатаційні переваги і допомагають економити завдяки можливості горизонтального масштабування, що дозволяє скоротити кількість дорогих серверів без необхідності в оновленні. Ви можете масштабувати їх для обробки великих обсягів даних або зберігати єдину велику базу даних та легко розподіляються у кластерах серверів.

MapReduce технологія була впроваджена великими організаціями, такими як Google, Yahoo, Facebook, а також була прийнята Apache Hadoop, як HDFS, PIG, HIVE, а також для зберігання даних або виконання та обробки великих даних за допомогою HBase, який також відомий як NoSQL. Ця технологія має дві переваги.

1) Паралельна обробка. Найперша перевага – паралельна обробка. За допомогою MapReduce ми завжди можемо обробляти дані паралельно. Відповідно до наведеної діаграми, існує п’ять підпорядкованих машин, і деякі дані містяться на цих машинах. Тут дані обробляються паралельно за допомогою Hadoop MapReduce і, отже, обробка стає швидкою.

Насправді тут відбувається те, що весь шматок даних ділиться Hadoop HDFS на блок HDFS, і MapReduce обробляє ці шматки даних, і таким чином обробка стає швидкою.

2) Місцезнаходження даних Це одна універсальна річ, яку надає Hadoop MapReduce, тобто ми можемо обробляти дані там, де вони є. Що це означає?Обробка виконується на меншій кількості даних у кількох місцях паралельно. Це економить багато часу, а також пропускну здатність мережі, необхідну для переміщення великих даних з одного місця в інше. Тільки пам’ятайте, що дані, які ми обробляємо, - це великі дані, розбиті на шматки, і якщо ми почнемо переміщати великі дані безпосередньо через відведені канали мережі у централізовану машину та обробляти їх, то це не дасть нам переваг, оскільки ми збираємось споживати всю пропускну здатність при переміщенні даних на централізований сервер.

Отже, використовуючи Hadoop MapReduce ми не просто робимо “Паралельна обробка”, ми також обробляємо дані до відповідних підпорядкованих вузлів або вузлів даних, де є фрагменти даних, а отже, ми також “Збереження великої пропускної здатності мережі” що дуже вигідно.

Нарешті, SlaveMachines закінчують з обробкою даних, що зберігаються на SlaveMachines, і вони відправляють результати назад на головну машину, оскільки результати не такі великі, як блоки, що зберігалися на SlaveMachines. Отже, він не буде використовувати велику пропускну здатність.

Підпорядковані машини відправляють результат назад на Головну машину, ці результати агрегуються разом, а кінцевий результат відправляється назад на Клієнтську машину, яка подала завдання. Тут виникає одне питання – хто вирішує, які дані слід обробляти в якому вузли даних?

Клієнт передає завдання менеджеру ресурсів, а менеджер ресурсів – це той, хто надає вказівки для виконання завдання у відповідних вузлах даних, де перебувають дані, і вирішує на основі найближчого вузла даних, який доступний, так що багато мереж Пропускна здатність не використовується.

Apache Hadoop – це фреймворк з відкритим кодом для управління всіма типами даних (структурованими, неструктурованими та напівструктурованими).

Як ми всі знаємо, якщо ми хочемо обробляти, зберігати та управляти нашими даними, то СУБД є найкращим рішенням. Але дані повинні мати структурований формат для обробки з RDBMS. Крім того, якщо обсяг даних збільшується, RDBMS не здатний обробляти їх, і нам потрібно регулярно проводити очищення бази даних.

Це може спричинити втрату даних у минулому та не може дати точних та надійних результатів у деяких галузях, таких як прогноз погоди, банківська справа, страхування, продаж тощо. Інша проблема RDBMS полягає в тому, що якщо основний сервер не працює, ми можемо втратити важливі даних і страждають багато. Hadoop – це розподілена файлова система, яка може зберігати великі обсяги даних (дані у петабайтах та терабайтах). Швидкість обробки даних також дуже швидка і забезпечує надійні результати, оскільки вона має дуже високу систему відмовостійкості. Hadoop – це платформа програмування з відкритим кодом на основі Java, яка підтримує зберігання та обробку наборів великих даних у розподіленому обчислювальному середовищі. Hadoop базується на кластерній концепції з використанням товарного обладнання. Це не вимагає складної конфігурації, і ми можемо створити середовище Hadoop за допомогою дешевого, простого та легкого апаратного забезпечення.

Концепція кластеру простими словами - це дані, які зберігаються у форматі реплікації на декількох машинах, так що, коли будь-яка проблема або катастрофа трапляється в одному з місць, де перебувають дані, тоді повинна бути копія цих даних, безпечно доступна в іншому місці.

*2. Корпоративне сховище даних у хмарі (warehouses and cloud)*

В даний час концепції хмарних обчислень приділяється багато уваги. Як і багато інших технології, хмарні обчислення розвиваються відповідно до циклом зрілості технологій. Для початку визначимо, що собою представляють хмарні обчислення і як вони можуть бути використані в сфері передової аналітики і великих даних. Існує безліч визначень хмарних обчислень. Ми розглянемо два з них. Перше запропоновано в доповіді компанії McKinsey в 2009 году, в якому вказані три критерії для хмарної середовища: 1. Підприємства не мають капітальних витрат або витрат на інфраструктуру, зате несуть експлуатаційні витрати без будь-яких контрактних зобов'язань. 2. Ємність може бути збільшена або зменшена динамічно і негайно. Це відрізняє хмари від традиційних хостинг-провайдерів, які можуть обмежувати масштабування. 3. Базове обладнання може перебувати де завгодно. Користувач абстрагується від архітектурної специфікації. Крім того, апаратні засоби працюють в розрахованому на багато користувачів режимі, при якому кілька користувачів з різних організацій одночасно можуть отримати доступ до однієї і тієї ж інфраструктурі. Справжнє хмара має відповідати всім трьом перерахованим критеріям. Воно повинно приховувати базову інфраструктуру від користувача, дозволяти проводити масштабування на вимогу і мати на увазі оплату в міру використання. При використанні хмари немає необхідності турбуватися про ресурсних обмеженнях. Користувачі можуть своєчасно отримати те, що їм необхідно. Вони, звичайно, заплатять, але тільки за те, що використовували. Їм більше не доведеться воювати з системними адміністраторами за ресурси.

Інше визначення запропоновано Національним інститутом стандартів і технологій (NIST), що входить до Міністерства торгівлі США. Воно спирається на п'ять основних характеристик хмар хмарної среди:

1. Самообслуговування на вимогу.

2. Широкий доступ до мережі.

3. Створення пулу ресурсів.

4. Швидка еластичність.

5. Вимірюваний сервіс.

Кожен з цих критеріїв повинен бути врахований. Легко помітити схожості між визначеннями McKinsey і NIST. На сайті NIST ви можете більше дізнатися про роботу, виконану цим інститутом в сфері хмарних обчислень. У хмарних обчислень, як будь-яке явище, є хороші і погані сторони, плюси і мінуси, переваги і недоліки. Щоб зробити усвідомлений вибір, організаціям необхідно отримати достатньо інформації. Безсумнівно, в майбутньому хмара буде знаходити все більш широке застосування в сфері поглибленої аналітики. Це особливо актуально в сфері розробки; майбутнє сфери виробництва менш ясно. Тепер розглянемо два основних типи хмарних середовищ: публічні хмари і приватні хмари.

Публічним хмар приділяється найбільша увага. При роботі з публічним хмарою користувачі просто завантажують свої дані в хост-систему, а потім, коли їм потрібно буде використовувати ці дані, їм будуть виділені необхідні ресурси. І вони заплатять за те, що використовували. Такий підхід має певні переваги:

Забезпечується необхідна пропускна здатність, а користувачі платять тільки за те, що вони використовували.

Немає необхідності купувати систему, розраховану на максимальне навантаження, яка більшу частину часу буде використовувати тільки половину потужності.

Короткі періоди, коли потрібно обробляти великий обсяг даних, невикличуть будь-яких турбот. Потрібно просто заплатити за використання додаткових ресурсів

Наростити потужності можна, як правило, дуже швидко. Після отримання доступу до хмарної середовищі користувачі завантажують свої дані і приступають до аналізу.

Можна легко обмінюватися даними з іншими людьми незалежно від їх місцезнаходження, оскільки публічне хмара за визначенням знаходиться за межами корпоративного брандмауера. Будь-яка людина може отримати доступ до створеному середовищі.

У публічного хмари є і деякі недоліки:

Зазвичай при використанні публічного хмари існує мало гарантій хорошою продуктивності. Будь-яка кількість людей може одночасно претендувати на використання одних і тих же ресурсів. Звичайно, можна доплатити за ексклюзивне використання хмарного сервера.

Це може привести до високої нестійкості рівня продуктивності. Швидкість виконання завдань не буде відома до її відправки. Можна робити деякі прогнози на підставі минулих показників продуктивності, однак немає гарантій того, що вони будуть реалізовані.

Існують побоювання щодо безпеки даних. Хоча є люди, які стверджують, що в реальності проблеми менш серйозні, ніж в теорії, питання безпеки є серйозною проблемою.

Якщо хмара використовувати нераціонально, це може дорого обійтися, оскільки користувачі платять за все, що вони роблять. Некоректні запити, які використовують значні системні ресурси, дратують і у власній системі, проте з ними не пов'язані реальні прямі витрати. При використанні хмари через некоректне запиту ви можете понести досить великі витрати.

Працюючи в публічному хмарі, неможливо перевірити аудиторський слід і місце розташування даних. Буває, що неможливо навіть визначити, чи знаходяться всі дані в одній і тій же країні.

Апаратні ресурси хмари описуються як еластичні, тобто вони в будь-який час можуть досить легко збільшуватися і зменшуватися. Це означає легкість нарощування обчислювальної потужності за рахунок використання додаткових процесорів, систем зберігання та обсягу пам'яті сервера. Крім того, в будь-який час можна легко оплатити і використовувати 10 додаткових серверів. Однак цей вид масштабованості відрізняється від масивно-паралельних систем. Більшість серверів публічного хмари працюють незалежно один від одного. MPP-система являє собою єдину велику систему. Це означає, що, якщо організація управляє безліччю малих і середніх процесів, хмара може бути для неї великою підмогою. Проте хмара мало чим зможе допомогти, коли мова заходить про управління процесами, які виходять за рамки масштабів окремих серверів, що працюють в хмарному середовищі. Хоча програмне забезпечення масивно-паралельної системи може працювати в хмарі, той факт, що базове устаткування невідомо і може змінитися в будь-який час, створює численні проблеми, пов'язані з продуктивністю програмного забезпечення масивно-паралельної системи. Ймовірно, найкраще застосування публічних хмар - робота, пов'язана з дослідженням і розробкою, при якій не доводиться турбуватися про нестабільність рівня продуктивності. Якщо аналітикам необхідно по експериментувати з деякими новими даними, щоб зрозуміти, чим вони можуть бути корисні, то їм слід розглянути можливість використання хмарної середовища. У більшості випадків при проведенні аналітичних досліджень і розробок продуктивність не викликає особливих проблем аж до розгортання процесу. Для розгортання некритичних аналітичних процесів хмара - цілком підходяще середовище. Якщо ви турбуєтеся про захист даних, то публічне хмара може створювати проблеми. При використанні публічного хмари необхідно застосовувати хороші протоколи безпеки і інструменти, а також забезпечувати надійний захист свого середовища. Єдине, що знаходиться поза зоною вашого контролю, - це співробітники компанії-постачальника хмарного сервісу. Якщо серед них є хакери чи злодії, вони можуть завдати такої ж шкоди, як і співробітники всередині компанії. Така ймовірність вкрай низька, проте порушення безпеки співробітником компанії-постачальника хмарного сервісу викличе набагато сильніший громадський резонанс, ніж якби це був співробітник самої компанії. Розміщення чутливої персональної інформації в публічному хмарі вимагає вжиття заходів для забезпечення безпеки. В іншому разі не робіть цього.

Існує ще один нюанс, пов'язаний з публічним хмарою, про який люди не замислюються. Дійсно, працювати з хмарою дешевше, ніж купувати систему, однак якщо вам необхідно запускати кілька процесів, то вартість використання хмари може насправді перевищити вартість внутрішньофірмової системи. Чим більше користувачів вдаються до публічного хмарі і платять по мірі його використання, тим вигідніша покупка внутрішньофірмової системи. Крім економії витрат це може привести і до підвищення продуктивності системи, оскільки організація матиме над нею повний контроль. Згодом публічні хмари, ймовірно, забезпечать виконання найважливіших функцій корпоративного рівня за доступною ціною. Постачальники, які сьогодні пропонують більш високий рівень продуктивності, стягують за додаткові послуги набагато більше, ніж за базові хмарні сервіси. Цілком можливо, що постачальники вирішать проблеми безпеки. До тих пір публічні хмари будуть використовуватися компаніями здебільшого для розробки процесів.

В даний час хмарних обчислень приділяється багато уваги, і у хмарної архітектури, безумовно, є переваги. Організаціям необхідно розібратися в способах використання хмари, а також зрозуміти їх плюси і мінуси. Найближчим часом або в середньостроковій перспективі публічні хмари будуть застосовуватися в основному для розробки з використанням нечутливих даних. Приватні хмари або їх близькі «родичі» - аналітичні пісочниці - стануть все частіше застосовуватися для вирішення всіх видів аналітичних задач. Суть в тому, що для проведення досліджень і сприяння інноваціям організаціям має сенс мати гнучку, слабко структурироване і менш жорстко контрольоване середовище. Хмари - один із способів створення такого середовища.

ТЕМА 6

ОСОБЛИВОСТІ APACHE HADOOP. КОМПОНЕНТА MAPREDUCE

*1.* [*Особливості та архітектура Hadoop*](https://uk.myservername.com/what-is-hadoop-apache-hadoop-tutorial#Hadoop_Features)

Apache Hadoop – це фреймворк з відкритим кодом для управління всіма типами даних (структурованими, неструктурованими та напівструктурованими).

Як ми всі знаємо, якщо ми хочемо обробляти, зберігати та управляти нашими даними, то СУБД є найкращим рішенням. Але дані повинні мати структурований формат для обробки з RDBMS. Крім того, якщо обсяг даних збільшується, RDBMS не здатний обробляти їх, і нам потрібно регулярно проводити очищення бази даних.

Це може спричинити втрату даних у минулому та не може дати точних та надійних результатів у деяких галузях, таких як прогноз погоди, банківська справа, страхування, продаж тощо. Інша проблема RDBMS полягає в тому, що якщо основний сервер не працює, ми можемо втратити важливі даних і страждають багато. Hadoop – це розподілена файлова система, яка може зберігати великі обсяги даних (дані у петабайтах та терабайтах). Швидкість обробки даних також дуже швидка і забезпечує надійні результати, оскільки вона має дуже високу систему відмовостійкості. Hadoop – це платформа програмування з відкритим кодом на основі Java, яка підтримує зберігання та обробку наборів великих даних у розподіленому обчислювальному середовищі. Hadoop базується на кластерній концепції з використанням товарного обладнання. Це не вимагає складної конфігурації, і ми можемо створити середовище Hadoop за допомогою дешевого, простого та легкого апаратного забезпечення.

Нижче наведено кілька пунктів, які описують переваги Hadoop перед **RDBMS.**

| **Особливості** | **Hadoop** | **RDBMS** |
| --- | --- | --- |
| **Архітектура** | Hadoop базується на HDFS, MapReduce та YARN. | СУБД базується на властивостях ACID. |
| **Гучність** | Може обробляти великий обсяг даних. | СУБД не може обробляти великий обсяг даних. |

| **Різноманітність / типи даних** | Може обробляти структуровані, напівструктуровані та неструктуровані дані, такі як відео, зображення, файли CSV, xml тощо. | Обробляйте лише структуровані дані. |
| --- | --- | --- |
| **Швидкість** | Швидка обробка великих обсягів даних. | Дуже повільно під час обробки великої кількості даних. |
| **Пропускна здатність** | Висока пропускна здатність. | Низька пропускна здатність. |
| **Відмовостійкість** | Дуже добре | Не здатний відновити втрачені дані, якщо основний сервер не працює. |
| **Зберігання** | Дуже висока місткість сховища. | Не вдається зберегти Bigdata. |
| **Надійний** | Дуже надійні та генерують точні історичні та поточні звіти. | Не надійний з точки зору Bigdata. |

Hadoop - це основа для запуску програм на великих кластерах, побудованих з товарного обладнання. Рамка Hadoop прозоро забезпечує додаткам і надійність, і рух даних. Hadoop реалізує обчислювальну парадигму під назвою Map / Reduce, де додаток розділено на безліч невеликих фрагментів роботи, кожен з яких може бути виконаний або повторно виконаний на будь-якому вузлі кластера. Крім того, вона забезпечує розподілену файлову систему (HDFS), яка зберігає дані про обчислювальні вузли, забезпечуючи дуже високу сукупну пропускну здатність у кластері. І Map / Reduce, і розподілена файлова система розроблені таким чином, що збої вузлів автоматично обробляються рамкою.

Map / Reduce - парадигма програмування, популярна [Google,](http://research.google.com/archive/mapreduce.html) де завдання розбивається на невеликі порції та розподіляється на велику кількість вузлів для обробки (карта), а результати потім підсумовуються до остаточної відповіді (зменшити ). Google та Yahoo використовують це для своїх пошукових технологій, серед іншого.

Hadoop – це загальна основа для реалізації подібної схеми обробки. Що стосується того, чому він б'є попку, здебільшого тому, що він надає акуратні функції, такі як відмовостійкість і дозволяє зблизити майже будь-яке обладнання для обробки. Він також масштабує надзвичайно добре, якщо ваша проблема відповідає парадигмі.

*2. Компонента (модель) MapReduce*

MapReduce - це фреймворк для паралельного програмування. MapReduce не є ні базою даних, ні прямим її конкурентом. Однак дехто вважає, що ця модель може замінити бази даних і все інше. Фактично MapReduce є доповненням до існуючих технологій. Безліч завдань, що вирішуються в середовищі MapReduce, можуть бути вирішені в реляційної базі даних. Все зводиться до того, щоб вибрати найбільш підходящу середовище для вирішення тієї чи іншої проблеми. Якщо інструмент або технологія на щось здатні, то це ще не означає, що цей інструмент або технологія найкраща. Зосередивши увагу на те, для яких цілей модель MapReduce підходить найкраще, а не на тому, на що теоретично вона здатна, можна максимізувати отримані переваги. Робота MapReduce складається з двох основних процесів, що створюються програмістом: кроку «map» і кроку «reduce». Звідси і назва MapReduce. Ці кроки передаються в фреймворк MapReduce, який паралельно запускає програми в наборі робочих вузлів. Згадайте, що при використанні масивно-паралельної системи дані розподіляються по вузлах, яким потім можна адресувати запит. У випадку з MapReduce використовується велика кількість недорогого обладнання, на яке у міру необхідності запуску процесу передаються дані. Кожен з робочих вузлів MapReduce застосовує один і той же код до своєї частини даних. Проте робочі вузли не взаємодіють і навіть не знають про існування один одного. При наявності постійного потоку даних веб-журналів вони можуть бути роздані фрагментами різних робочих вузлів. Найпростіший спосіб - циклічна процедура (round-robin), при якій записи послідовно знову і знову передаються в вузли. Часто застосовуються деякі види хеширования, в цьому випадку записи передаються робочим вузлам на основі формули, щоб схожі записи відправлялися одному і тому самому вузла. Наприклад, при хешуванні ідентифікатора клієнта все записи, що стосуються даного клієнта, передаються одному і тому ж робочому вузлу. Це дуже важливо, якщо планується проведення аналізу за ідентифікатором клієнта.

Сайт Mapreduce.org визначає модель MapReduce як програмний каркас для спрощення обробки масивних наборів даних. Hadoop - популярна версія MapReduce з відкритим вихідним кодом, що поставляється організацією Apache. Hadoop є найвідомішу реалізацію фреймворка MapReduce. В даному розділі ми будемо використовувати загальний термін MapReduce, проте вся подальша інформація стосується будь-якої його реалізації. Організації починають усвідомлювати важливість швидкого аналізу величезної кількості даних, які вони створюють, щоб приймати більш зважені рішення. MapReduce допомагає цим організаціям впоратися з неструктурованими і частково структурованими джерелами, які складно аналізувати за допомогою традиційних інструментів. Більшість підприємств має справу з численними типами даних крім даних з реляційних баз даних. До них відносяться текст, дані, згенеровані машинами, наприклад веб-журнали, або дані, отримані від датчиків, зображення і т. Д. Організаціям необхідно швидко і ефективно обробляти всі ці дані, щоб отримувати цінні відомості. За допомогою моделі MapReduce обчислювальні операції проводяться над даними, що зберігаються в файлової системі без завантаження в базу даних. Пізніше ми повернемося до цієї ключової особливості. Велика перевага середовища MapReduce полягає в можливості обробки неструктурованого тексту. У реляційній базі даних вся інформація вже представлена у вигляді таблиць, що складаються з рядків і стовпців. Взаємозв'язку між даними вже добре визначені. Це не завжди вірно у випадку з потоками необроблених даних. Ось де модель MapReduce дійсно може бути корисною! Завантаження великих фрагментів тексту в BLOB-поле в базі даних можлива, але це далеко не кращий метод використання бази даних і не кращий спосіб обробки таких даних. Тут може допомогти модель MapReduce.

Використання MapReduce часто служить відправною точкою, а результат застосування даного алгоритму використовується в якості вхідних даних для іншого аналітичного процесу. Тисячі процесів MapReduce можуть виконуватися на тисячах комп'ютерів. Саме в такій ситуації виявляється вся міць даної моделі. У тих випадках, коли є величезні потоки даних і процес їх обробки може бути розбитий на частини, модель MapReduce може застосовуватися з великим успіхом. Якщо для ефективної роботи одного робочого вузла необов'язково знати про те, що відбувається з іншим, можна забезпечити повністю паралельну обробку. У нашому прикладі кожне слово може бути розібрано само по собі, а зміст інших слів не має значення для даного робочого вузла. Попередній пункт не можна ігнорувати, оскільки він дуже важливий для розуміння того, коли і як варто застосовувати алгоритм MapReduce. Коли дані передаються робочим вузлам, кожен з них знає тільки про тих даних, які він бачить. Якщо процес обробки вимагає того, щоб робочий вузол знав про дані, що знаходяться на інших вузлах, то слід використовувати не MapReduce, а інший фреймворк. На щастя, є багато випадків, коли дані можуть бути оброблені таким способом. Розбір одного веб-журналу або однієї RFID-записи на фрагменти ні від чого не залежить. Коли необхідно розібрати текст за ідентифікатором клієнта, при розподілі даних їх просто потрібно хешировать так, щоб всі записи, що стосуються даного клієнта, виявилися на одному і тому ж робочому вузлі. Концептуально алгоритм MapReduce розбиває проблему на частини так само, як це робить паралельна реляційна база даних. Однак MapReduce - це не база даних, бо у неї відсутня певна структура. Кожен окремий процес нічого не знає про те, що відбувається до або після нього. Існують певні подібності в тому, що ви можете робити з допомогою MapReduce і бази даних. База даних може забезпечити вхідні дані для процесу MapReduce, як і процес MapReduce - для бази даних. Ключовий фактор - вибір найкращого способу для вирішення конкретного завдання. Якщо щось можна зробити за допомогою даного набору інструментів, це не означає, що цей спосіб найкращий. Інші набори інструментів можуть бути набагато більш придатними. Бази даних та модель MapReduce повинні використовуватися для вирішення тих завдань, для яких вони найкраще підходять. сильні і слабкі сторони моделі MapReduce Процес MapReduce працює на стандартному недорогому обладнанні. Це означає, що його можна налагодити з невеликими витратами. Крім того, його розширення обійдеться досить дешево. Збільшити ємність легко, оскільки все, що потрібно, - це додаткові сервери і їх інтеграція. Ми вже говорили про те, що з рішенням деяких завдань MapReduce справляється набагато краще, ніж реляційні бази даних. Це розбір тексту, обробка даних веб-журналів і читання інформації з величезного джерела необроблених даних. Модель MapReduce найбільш ефективна у випадках, коли існує великий обсяг вхідних даних, більша частина яких не потрібна для аналізу. Якщо велике значення має мала частина даних, але заздалегідь неясно, яка саме, то MapReduce може допомогти. Цей алгоритм дозволяє розібратися в великому обсязі даних і витягти важливі фрагменти.

Немає сенсу витрачати багато часу і ресурсів на завантаження величезної кількості необробленої інформації в корпоративне сховище даних, якщо в процесі їх обробки більша частина буде відкинута. Якщо дані потрібні тільки на короткий період, не варто поміщати їх в сховище даних. Модель MapReduce ідеально підходить в таких випадках. Відкиньте надлишки інформації до її завантаження в базу даних. У багатьох ситуаціях модель MapReduce використовується як інструмент вилучення, перетворення і завантаження даних (ETL). ETL- інструменти прочитують набір вихідних даних, виробляють комплекс дій, пов'язаних з форматуванням або реорганізацією, а потім завантажують результати в підсумковий джерело даних. Для забезпечення аналізу ETL-інструменти беруть дані з систем взаємодії підрозділів підприємства і завантажують їх у реляційну базу даних, щоб забезпечити до них доступ. Модель MapReduce часто використовується для обробки джерел великих даних, вилучення з них мають цінність відомостей і передачі результатів в базу даних або аналітичний процес. У попередньому прикладі необроблений текст перетворюється в набір слів із зазначенням частоти їх використання, який можна проаналізувати, а результати передати в базу даних, щоб об'єднати цю інформацію з додатковими даними. Модель MapReduce НЕ база даних, тому не має вбудованої системи безпеки, засобів індексування, оптимізатора запитів або процесів, історії виконаних завдань і відомостей про інших існуючих даних. Вона забезпечує не тільки максимальну гнучкість при обробці різних видів даних, але також несе відповідальність за визначення того, що собою представляють дані, в кожному створеному процесі. Практично завжди буде застосовуватися для користувача код, в тому числі і для структури даних. Кожне завдання являє собою окремий елемент, якому нічого не відомо про те, що відбувається за межами моделі.

*3.* [*Зберігання та обробка даних у Hadoop*](https://uk.myservername.com/comprehensive-hadoop-testing-tutorial-big-data-testing-guide#Storing_And_Processing_Data_In_Hadoop)

Для виконання цих процесів у системі Hadoop ми маємо робочу силу, яка розділена на чотири секції.

* Адміністратори Hadoop відповідають за налаштування навколишнього середовища та мають адміністративні права на доступ до систем Hadoop.
* Розробники Hadoop розробити програми щодо витягування, зберігання та обробки даних з різних місць до централізованих місць.
* Тестувальники Hadoop для перевірки та перевірки даних перед витяганням з різних місць і після витягування з централізованого місця, а також перевірка та перевірка здійснюється під час завантаження даних у клієнтське середовище.
* Аналітики Hadoop функціонують, коли виконується завантаження даних і коли дані надходять на склад у місцезнаходження клієнта. Вони використовують ці дані для створення звітів та інформаційної панелі. Аналітики проводять аналіз даних для зростання та розвитку бізнесу.

Ми знаємо, що Hadoop - це не єдина система; він містить безліч систем і машин. Дані поділяються та зберігаються на декількох машинах, і якщо ми хочемо отримати до них доступ знову, нам потрібно об’єднати та втягнути дані у звіти тощо. Розробник відповідає за написання програм на JAVA та Python для вилучення даних та їх зберігання. Інша робота розробника - це обробка даних. Існує два шари Hadoop, один призначений для зберігання, тобто Hadoop HDFS, а інший - для обробки, тобто Hadoop MapReduce.

Зберігання означає, що всі дані, які ми маємо у джерелі, просто зберігаються / вставляються в систему. Обробка означає, що нам потрібно розділити її на кілька машин і знову об’єднати та надіслати клієнту. Таким чином, зберігання та обробка здійснюються за допомогою сценаріїв програмування, а розробник відповідає за написання сценаріїв. Окрім програмування, іншим методом зберігання та обробки даних у Hadoop є використання програм баз даних, таких як Hive, Impala, HBase тощо. Ці інструменти не потребують знань з програмування. Після зберігання та обробки розробником дані надходять на формування звіту. До цього нам потрібно перевірити оброблювані дані на точність і перевірити, чи дані точно завантажені та оброблені правильно чи ні.

Отже, програму та / або сценарії, створені розробником, потрібно перевірити Hadoop або BigData Tester. Тестувальник повинен знати базове програмування, таке як Mapper, Hive, Pig Scripts тощо, щоб перевірити сценарії та виконати команди.

Отже, перед тестуванням тестувальники повинні знати, які всі програми та сценарії працюють, як написати код, а потім подумати, як їх протестувати. Тестування може проводитися як вручну, так і за допомогою засобів автоматизації. Hadoop має різні види тестування, такі як модульне тестування, тестування регресії, тестування системи та тестування продуктивності тощо. Отже, це загальні типи тестування, які ми використовуємо в нашому звичайному тестуванні, а також тестування Hadoop та BigData.

Тестування BigData означає перевірку та перевірку даних при зберіганні та обробці їх у сховищі даних. Під час тестування BigData нам потрібно перевірити обсяг та різноманітність даних, вилучених з різних баз даних та завантажених, а також оброблених у сховищі даних або системі Hadoop, це тестування підлягає функціональному тестуванню. Нам потрібно протестувати швидкість даних, завантажених з різних баз даних і завантажених до системи Hadoop, яка є частиною тестування продуктивності.

Отже, як план або стратегія, нам потрібно зосередитись на функціональному, а також на тестуванні продуктивності тестування BigData. У тестуванні BigData тестувальник повинен перевірити обробку величезної кількості даних за допомогою товарного обладнання та відносних компонентів. Отже, якість даних також відіграє важливу роль у тестуванні BigData. Важливо перевірити та перевірити якість даних.

Функціональне тестування та тестування продуктивності відіграють життєво важливу роль у тестуванні BigData, окрім тестування BigData, нам потрібно провести ще кілька видів тестування, таких як тестування баз даних, а також архітектурне тестування.

Ці типи тестування також настільки важливі, як тестування функціональності та продуктивності.

1) Архітектурне тестування. Це тестування проводиться для того, щоб переконатися, що обробка даних правильна та відповідає вимогам. Насправді, система Hadoop обробляє величезні обсяги даних і є всебічною.Якщо архітектура неправильна, це може погіршити продуктивність, через що обробка даних може перерватись і може статися втрата даних.

2) Тестування бази даних. Тут перевірка процесу вкладається в картину, і нам потрібно перевірити дані з різних баз даних, тобто нам потрібно переконатися, що дані, отримані з вихідних баз даних або локальних баз даних, повинні бути правильними та належними.

Крім того, нам потрібно перевірити, щоб дані, наявні у вихідних базах даних, відповідали даним, що вводяться в систему Hadoop. Подібним чином нам потрібно перевірити, чи дані в системі Hadoop правильні та правильні після обробки або, скажімо, після перетворення, і щоб бути завантаженими в середовище клієнта з відповідною валідацією та верифікацією.

3) Тестування продуктивності. В рамках тестування продуктивності нам потрібно перевірити швидкість завантаження та обробки даних, тобто як IOPS (Input Output Per Second). Потрібно перевірити швидкість введення даних або даних як вхідних даних з різних баз даних до сховища даних або системи Hadoop та від системи Hadoop або сховища даних до середовища клієнта. Потрібно також перевірити швидкість даних, що надходять з різних баз даних та зі Складу даних, як вихідні дані. Це те, що ми називаємо вхідним виходом на секунду або IOPS.

Окрім цього, ще одним аспектом є перевірка ефективності поглинання та розподілу даних, а також наскільки швидко дані споживаються сховищем даних з різних баз даних та системою клієнта з системи Hadoop. Також, як тестувальник, нам потрібно перевірити ефективність розподілу даних, наприклад, наскільки швидко дані розподіляються між різними файлами, доступними в системі Hadoop або в сховищі даних. Подібним чином той самий процес відбувається під час розподілу даних до клієнтських систем.

Система Hadoop або сховище даних складається з декількох компонентів, тому тестер повинен перевірити ефективність усіх цих компонентів, таких як Завдання MapReduce, вставка та споживання даних, час відгуку запитів та їх продуктивність, а також продуктивність пошуку операцій. Все це включено до тестування продуктивності.

4) Функціональне тестування. Функціональне тестування містить тестування всіх підкомпонентів, програм та скриптів, інструментів, що використовуються для виконання операцій зберігання або завантаження та обробки тощо. Для тестера це чотири важливі типи та етапи, через які дані потрібно відфільтрувати, щоб клієнт отримав ідеальні дані без помилок.

Інструменти для тестування BigData Hadoop:

Існують різні інструменти, які використовуються для тестування BigData:

* Файлова система розподілу HDFS Hadoop для зберігання BigData.
* Зниження карти HDFS для обробки BigData.
* Для NoSQL або HQL Cassandra DB, ZooKeeper та HBase тощо.
* Хмарні серверні інструменти, такі як EC2.

Тестування середовищ та налаштувань

Для будь-якого типу тестування тестувальник потребує належних налаштувань та середовища.

Нижче наведено перелік вимог:

1. Тип даних та програми, які будуть перевірені.
2. Зберігання та обробка вимагає великого простору для величезного обсягу даних.
3. Правильний розподіл файлів на всіх вузлах даних загалом у кластері.
4. Під час обробки даних використання апаратного забезпечення має бути мінімальним.
5. Запущені програми та сценарії відповідно до вимог Програми.

Ролі та обов'язки тестування Hadoop

Як тестувальник Hadoop, ми несемо відповідальність за розуміння вимог, підготовку кошторисів випробувань, планування тестових кейсів, отримання деяких даних тестування для тестування деяких тестових кейсів, участь у створенні випробувального стенду, виконанні планів випробувань, звітуванні та повторному тестуванні дефектів. ісля того, як ми запропонуємо рішення нашої проблеми, нам потрібно продовжити і спланувати або розробити стратегію нашого плану випробувань, ми можемо обговорити стратегію автоматизації, яку ми можемо використовувати там, план про графік випробувань, який залежить від наших термінів постачання, також ми може обговорювати планування ресурсів.

Стратегія автоматизації допоможе нам зменшити ручні зусилля, необхідні для тестування продукту. Графік випробувань важливий, оскільки він забезпечить своєчасну доставку товару.

Планування ресурсів буде мати вирішальне значення, оскільки нам потрібно спланувати, скільки людських годин нам потрібно на нашому тестуванні та скільки ресурсів Hadoop потрібно для виконання нашого планування тестів.

Після того, як ми розробимо стратегію тестування, нам потрібно продовжувати створювати плани розробки тестів, які включають створення планів тестування, створення тестових сценаріїв, які допоможуть нам автоматизувати тестування, а також визначити деякі дані тестування, які будуть використовуватися в планах тестування. і допомагає нам виконувати ці тестові плани. Коли ми закінчимо розробку тестів, що включає створення тестових планів, тестових сценаріїв та тестових даних, ми продовжуємо і починаємо виконувати ці тестові плани. Коли ми виконуємо тестові плани, можуть бути певні сценарії, коли фактичний результат не відповідає очікуваному, і ці речі називаються дефектами. Всякий раз, коли є дефект, нам також потрібно перевірити ці дефекти, і нам потрібно створити і підтримувати матриці для них.

Управління дефектами складається з відстеження помилок, виправлення помилок та перевірки помилок. Щоразу, коли план тестування виконується щодо будь-якого з наявних у нас продуктів, і як тільки виявляється конкретна помилка або виявляється дефект, про цей дефект потрібно повідомляти розробнику або призначати розробнику.

Тож розробник може розглянути це і почати над ним працювати. Як тестувальник, нам потрібно відстежувати хід помилки та відстежувати, чи помилку виправлено. Якщо помилка була виправлена, як повідомляється, тоді нам потрібно повторно протестувати її та перевірити, чи вона вирішена.

**Список літератури**

1. Кузьменко О., Козьменко О. Економіко-математичні методи і моделі. Економетрика : навч. посіб. К. : «Університетська книга», 2019. 406 с.
2. Pang-Ning Tan, Steinbach M., Karpatne A., Kumar V. Introduction to Data Mining. 2nd edition. New York: Pearson, 2018. 1450 p.
3. 3 Zak Cameron. Data Mining Concepts and Techniques: Complete Guide to a Comprehensive Understanding of Data Mining. Independently published, 2020. 366 p.
4. Фрэнкс Б. «Революция в аналитике. Как в эпоху Big Data улучшить ваш бизнес с помощью операционной аналитики» / Билл Фрэнкс. — Москва: Альпина Паблишер, 2017. — 320 c.
5. Глущенко Н. Большие данные большого города: как Big Data меняет жизнь Киева [Электронный ресурс] / Нина Глущенко // интернет-журнал AIN.UA. — Электронные данные. — [Киев: AIN.UA, 2017]. — Режим доступа: <https://ain.ua/special/big-data-in-kyiv/>
6. Силен Д., Мейсман А., Али М. Основы Data Science и Big Data. Python и наука о данных. – СПб.: Питер, 2017. – 336 с.
7. Дэви С. Основы Data Science и Big Data. Python и наука о даннях // С. Дэви, М. Арно, А. Мохамед. – СПб.: Питер, 2017. – 336 c.: ил.
8. C. Chen, M. Lin, and X. Guo, “High-level modeling and synthesis of smart sensor networks for Industrial Internet of Things,” Computers & Electrical Engineering, vol. 61, pp. 48–66, 2017.
9. Harrison, Guy. Next Generation Databases: NoSQL, NewSQL and Big Data. 2016.
10. Daniel Keim, Jörn Kohlhammer, Geoffrey Ellis und Florian Mansmann. „Visual Analytics“. 2010
11. Dimitri P. Bertsekas and John N. Tsitsiklis. Introduction to Probability. Charles Wheelan. Naked Statistics: Stripping the Dread from the Data. W. W. Norton and Company, 2013.
12. F. Liu, Y. Liu, D. Jin, X. Jia, and T. Wang, “Research on Workshop-Based Positioning Technology Based on Internet of Things in Big Data Background,” Complexity, vol. 2018, Article ID 875460, 11 pages, 2018.
13. H. Mora, M. Signes-Pont, D. Gil, and M. Johnsson, “Collaborative Working Architecture for IoT-Based Applications,” Sensors, vol. 18, no. 6, p. 1676, 2018.
14. H. Tahaei, R. Salleh, S. Khan, R. Izard, K.-K. R. Choo, and N. B. Anuar, “A multi-objective software defined network traffic measurement,” Measurement, vol. 95, pp. 317–327, 2017.
15. Hariri, R.H., Fredericks, E.M. & Bowers, K.M. Uncertainty in big data analytics: survey, opportunities, and challenges. J Big Data 6, 44 (2019). <https://doi.org/10.1186/s40537-019-0206-3>.
16. [Smart Big Data Management,](https://books.google.de/books?id=A_u3oQEACAAJ&dq=Big+Data+Management&hl=de&newbks=1&newbks_redir=1&sa=X&ved=2ahUKEwiZh-vomq3vAhWMDewKHYk7CgEQ6AEwAnoECAMQAg) Frank Keuper, ‎Dietmar Schmidt, ‎Marc Schomann, 2014.
17. Barrow C., Barrow P., Brown R. The Business Plan Workbook: A Step-By-Step Guide to Creating and Developing a Successful Business. 9th Edition. Kogan Page, 2018. 407 p.
18. Behera H.S. (ed.) et al. Computational Intelligence in Data Mining. Springer, 2018. 895p.
19. Bhatia P. Data Mining and Data Warehousing: Principles and Practical Techniques. Cambridge University Press, 2019. 513 p.
20. Business Result Pre-Intermediate. Teacher’s book. 2nd edition. Rachel Appleby, Mark Bartan, David Grant. Oxford University Press, UK, 2017. 97 p.
21. Byrd M.J. Small Business Management: An Entrepreneur’s Guidebook. 8th Ed. McGraw-Hill, 2017. 494 p.
22. Carlberg C. Business Analysis with Microsoft Excel. 5th edition. Que Publishing, 2018. 576 p.
23. Cicala G. Project Management Using Microsoft Project 2019: A Training and Reference Guide for Project Managers Using Standard, Professional, Server, Web Application and Project Online for Office 365. Project Assistants Inc., 2019. 446 p.
24. Dr. Eckroth J. AI Blueprints: How to build and deploy AI business projects. Packt, 2018. 241p.
25. Esposito A., Esposito A.M., Jain L.C. (Eds.) Innovations in Big Data Mining and Embedded Knowledge. Springer, 2019. 286 p. (Intelligent Systems Reference Library 159).
26. Fitzgerald Brian, Stol Klaas-Jan. Scaling a Software Business. Springer, 2017. 264 p.
27. Harris Tom. Start-up: A Practical Guide to Starting and Running a New Business. Springer, 2018. 153 p.
28. Harvard Business Review – Entrepreneur’s Handbook. Boston: Harvard Business School Publishing Corporation, 2018. 303 p.
29. Hastie Trevor, Tibshirani Robert, Friedman Jerome. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2009. Corrected 12th printing, 2017. 745 p.
30. Graph Databases, 2nd Edition, by Ian Robinson, Jim Webber, Emil Eifrem, 2015, O'Reilly Media
31. Kreinovich V., Sriboonchitta S., Chakpitak N. (eds.) Predictive Econometrics and Big Data. Springer, 2018. 788 p.
32. Kryvinska N., Gregus M. (eds.) Data-Centric Business and Applications Evolvements in Business Information Processing and Management (Vol.2). Springer, 2020. 468 p.
33. Lee C.F., Chen H.Y., Lee J. Financial Econometrics, Mathematics and Statistics: Theory, Method and Application. Springer, 2019. 656 p.
34. Olson D.L., Wu D. Predictive Data Mining Models. 2nd.ed. Springer, 2020. 125 p. Computational Risk Management.
35. Olson David L. Descriptive Data Mining. 2nd.ed. Springer, 2019. 130 p. (Computational Risk Management).
36. Project Management Institute. A Guide to the Project Management Body of Knowledge: PMBOK Guide-Sixth Edition + Agile Practice Guide. 6th Edition. Project Management Institute, 2017. 800 p.
37. Project Management Institute. Agile: Practice Guide. Project Management Institute, 2017. 115 p.
38. Rojas I., Pomares H., Valenzuela O. (eds.) Advances in Time Series Analysis and Forecasting: Selected Contributions from ITISE 2016. Springer, 2017. 412 p.
39. Sul D. Panel Data Econometrics: Common Factor Analysis for Empirical Researchers. New York: Routledge, 2019. 165 p.
40. Wilcox J. Excel: The Ultimate Statistics Guide. Mobo Publications, 2017. 251p.

Список інтернет джерел:

1. <https://rb.ru/howto/chto-takoe-big-data/>
2. <https://www.mysql.com/>
3. <https://app.diagrams.net/>
4. <https://cloudstack.apache.org/>
5. <https://projects.apache.org/projects.html?category>
6. <https://cloud.google.com/bigtable>
7. <https://aws.amazon.com/de/dynamodb/>
8. <https://neo4j.com/>
9. https://www.datacenterknowledge.com/archives/2015/03/30/big-data-bubble-set-burst
10. [https://ru.wikipedia.org/wiki/Большие\_данные](https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5)
11. <https://intellect.ml/big-data-6821>
12. [http://sewiki.ru/index.php?title=Большие\_данные&oldid=3075](http://sewiki.ru/index.php?title=%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5&oldid=3075)
13. ttp://www.mckinsey.com/insights/business\_technology/big\_data\_the\_next\_fro ntier\_for\_innovation
14. <http://engjournal.ru/articles/1228/1228.pdf>
15. http://www.ogcs.com.ua/index.php/articles/121-big-data-v-promyshlennosti-innovatsii-k-kotorym-pridetsya-privykat