Educational Matters
Научно-методически статии

ЗА ДАННИТЕ И ПЪТЯ КЪМ ГОЛЕМИТЕ ДАННИ

Павел Азълов
САЩ

Резюме. В последните две десетилетия данните прераснаха в „големи данни“. Това е термин, с който най-общо се обяснява тенденцията за огромното нарастване на данните във всичките им разновидности, и особено на полуструктурираните и неструктурираните данни. Големите масиви от данни, тяхното разнообразие и скоростта, с която се генерират някои от тях, провокират необходимостта от въвеждането на нови технологии за събирането, предаването, съхранението и обработката на данни.

Настоящата статия представя хронологията на основните постижения в областта на базите от данни от най-първите им години до възникването на големите данни. Успоредно с това се дава и кратка справка за авторите, чиито научни и практически резултати изградиха теорията и практиката на системите за обработване на данните – малки и големи. Представени са и типични примери на „срещи“ с големите данни от ежедневието. Накратко са описани основните технологии, използвани при големите данни, включително екосистемата Hadoop и NoSQL базите от данни. Специално се отбелязва и ролята на аналитиката на данните като подход за трансформиране на данните във факти, модели и знания.

Ключови думи: големи данни; история на бази данни; NoSQL база данни; структурирани бази данни; неструктурирани бази данни; полуструктурирани бази данни; екосистемата Hadoop

„Живеем в ерата на информацията“
е популярен израз, но всъщност
ние живеем в ерата на данните.
Jiawei Han, Micheline Kamber, Jian Pei[6)]

Живеем в *ерата на данните*. Всичко, което ни заобикаля, започна да се дигитализира. Текстове, графика и снимки, видеа и музика вече се трансформираха в числа. Животът ни се променя с всеки изминал ден и започваме постепенно да осъзнаваме, че ни управляват невидими знания, скрити в „дълбините“ на данните. А данните ни заобикалят отвсякъде. Самите ние вече сме част от тях. Айфонът ни знае всичко за нас и около нас, за семейството ни, за приятелите и колегите ни. Знае банковите ни сметки и данъците, които дължим, знае координатите ни във всеки момент, направлява

ни по непознатите пътища, избирайки най-подходящите от тях. Докато автомобилът ни навърта километър след километър на автономен режим, ние банкираме, информираме се за финансовите пазари, разглеждаме част от проекта, по който работим с колеги, и се чувстваме комфортно като в офиса си. Други предпочитат да използват това време, за да научат новините за локалната и световната икономика и политика, да отворят новите си писма, да проверят хода на придвижване на пощенската си пратка, а най-често и да си „побъбрят“ с приятел или да се „видят“ в някоя от социалните мрежи. А през това време видеокамери се грижат за безопасността на дома ни и следят за трафика по булевардите на големите градове. Хиляди сензори регистрират данни за околната среда – за качеството на въздуха, за нивата на реките и количеството на падналия дъжд, регистрират сеизмичната обстановка и промените на климата. Сървърите трупат терабайтове с данни, получени от научни експерименти и сателити. Лекари и медицински сестри сканират данните от измерванията и процедурите, които извършват на всеки пациент, добавяйки ги в здравното му досие. Постепенно в съвременните домове и градове се настанява изкуственият интелект и те стават все по-удобни и приятни за ползване. Всеки един от тези моменти от ежедневието ни пряко или косвено е свързан с големите данни.

Акцентът в тази статия е върху данните, и по-специално е насочен към *big data* като термин, за който у нас вече е приет дословният превод *големи данни*.

Какво всъщност са големите данни? Терминът е неясен и не звучи добре на български език. Но изглежда, че това не трябва много да ни смущава, защото в бизнес английския речник с прилагателното *голям* има свързани и други подобни съществителни, като например *big oil* (голям нефт), *big ore* (голяма руда). Засега като начало приемаме, че *големи данни* е термин, използван за означаване на големи масиви от данни, които е трудно или невъзможно да се обработват с традиционните системи за управление на бази от данни (СУБД).

През последните 80 години човечеството е навлязло в три технологични революции. Те са започнали по различно време, но продължават и до днес. Първата от тях се свързана с *компютърния хардуер* и конструирането на първия компютър. С усъвършенстването на изчислителната мощ на компютрите настъпват годините на разцвет на компютърния *софтуер*. Симбиозата на хардуера и софтуера създава условията за дигитализиране на обществото. Така стартира и третата технологична революция, която откри пътя към големите данни, и хората заживяха в *ерата на данните*.

Данните са навсякъде в нашето ежедневие. А те са нещо много повече от числа, текстове, музика и снимки. *Знанията*, вградени в тях, които постепенно се разкриват, усвояват и прилагат в живота, са истинското неоценимо богатство.

Започваме нашия разказ с разглеждането на „обикновените“ данни.

1. Данни, информация и знание

Всеки има добре изградена представа за понятията *данна* и *информация* и ежедневно те се използват с почти един и същ смисъл. Нека още в началото да уточним, че за удобство, а също и за да избегнем ненужни интерпретации от други области на науката, в тази статия разглеждаме понятията *данни* и *информация* само в контекста на използването им при работа с компютри.

Данни

Приемаме, че понятието *данна* (*данни*) е първично, с което избягваме необходимостта от формалното му дефиниране. Това по никакъв начин не намалява общността на разглежданата тема. Неформално *данните* са входен обект на компютърна програма, който подлежи на някаква обработка от нея. Данни например са числата, знаците, произволни текстове, фигури, чертежи, таблици, картини, звуци и други. Сами за себе си те нямат смисъл, ако такъв специално не им бъде присвоен по определен начин. Елементарен пример за това може да бъде редица от числа, представляваща вход за програма, която извършва определени операции с числата, без да познава техния смисъл.

Информация

Информацията се приема като резултат, получен от обработката на съответно множество от данни. Може да се каже, че това са нови данни, чийто смисъл е познат на човека. Например числата 182 и 176 са данни, но ако към всяко от тях се добави, че Иван е висок 182 см, а Петър е 176 см, тогава резултатът от сравняването на тези числа, т.е. че Иван е по-висок от Петър, вече е някаква информация.

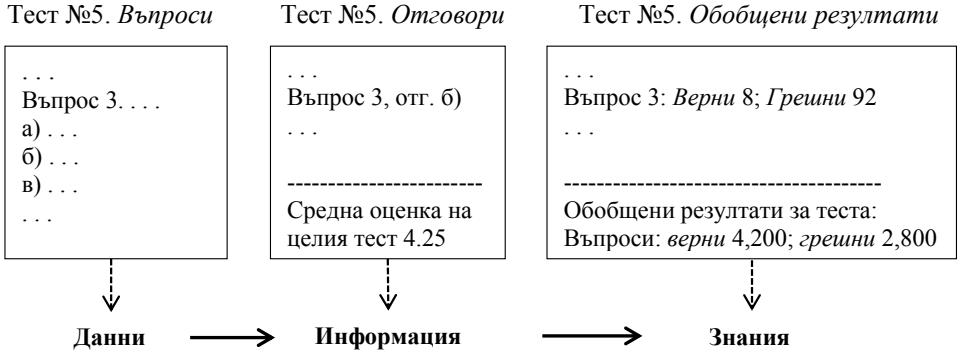
Като изходна точка приехме, че данните са вход на компютърна програма, а информацията е изходът от изпълнението на програмата. Но за по-доброто изясняване на разликата между тях да се опитаме да ги „погледнем“ от различни ъгли.

– Програмата се нуждае от данни. Човек се нуждае от информация.
– Данните не зависят от информацията. Информацията е зависима от данните.
– При изпълнението на една програма от входните ѝ данни се „извлича“ някаква информация.
– Решенията не се вземат на базата на данни, а на базата на информация.
– „Може да имате данни без информация, но не може да имате информация без данни“ (Daniel Keys Moran).
– Информацията, получена от едни данни, може да се използва като данни за получаване на нова информация.
– Данните са най-ниското ниво на някакво знание, което е незначимо за програмата, с която се обработват. Информацията е второто ниво на знание, извлечено от данните, на които е придаден смисъл.

– „Много данни, но малко информация“ – израз, с който се казва, че от данните не е извлечена полезна информация.

Знание

За *знанието* се казва, че е едно трето ниво, което може да се получи от данните. То представлява определена комбинация от *информация* и *опит*, които позволяват на човек да *прави изводи* и да *взема решения*.



Фигура 1. Илюстрация на последователността: данни, информация, знание

Нека да се опитаме да доизясним понятието *знание*, като го съпоставим с понятието *информация*.

– Информацията е градивен елемент на всяко знание, но сама за себе си информацията все още не е достатъчна, за да се говори за знание.
– С отделна информация могат да се обяснят частично някои факти и събития, но тя невинаги е достатъчна за пълното разбиране на темата, свързана с тези факти.

Да разгледаме и един познат пример от практиката, илюстриран на фиг. 1. Текущият контрол е използван подход в обучението. Често пъти той се провежда чрез тестове с избираеми отговори. Въпросите с избираемите отговори са данни за програмата, която проверява отговорите на всеки студент. Отговорите на въпросите и крайната оценка представляват добра информация за отделния студент. Обобщените резултати, които преподавателят получава за всички въпроси и поотделно за всеки един от тях, както и за всеки индивидуален обучаем, е информация, от която той може да направи редица изводи, като например „*Кои са въпросите*, *на които резултатите са отлични*, *добри*, *незадоволителни или слаби*?“. Ако след повторението на подобен тест в продължение на няколко семестъра се установи, че студентите имат незадоволителни или слаби резултати на въпросите от една и съща тема, то тогава преподавателят навярно ще се замисли дали това не е резултат от начина на представяне на тема, или пък въпросите не са формулирани добре, или са

много трудни. В този смисъл, част от информацията от тези тестове може да се приеме като знание. Преподавателят е осъзнал и вече знае, че тази тема е проблемна за студентите, и би могъл да използва тези изводи при управление на учебния процес.

В информатиката има редица важни понятия, свързани с данните, като две от фундаменталните са *тип на данните* и *структура на данните*. Те са близки по смисъл, но между тях има съществена разлика.

Тип на данните

Основна характеристика на данните в езиците за програмиране и в специализираните езици при базите от данни е техният *тип*. Обикновено данните се групират в отделни категории според специфични техни характеристики. За дан-ните от една и съща категория се казва, че са от един и същ *тип*. Множеството на целите числа, представими в една система за програмиране, е пример за такава категория от данни. Други примери са множеството на т.нар. реални числа (в компютрите се работи само с рационални числа) и множеството от знакови низове.

Представянето на данните в компютърната памет, както и на съответното множество от операции на най-ниско ниво са въпроси от компетенцията на съответния процесор. На това ниво се говори за електронни схеми, битове и байтове. Типът на данни, или както е прието да се нарича, *типът данни* е абстракция, която „скрива“ тези подробности и директно предоставя възможности за използването на множество от данни и съответните му операции, без да е необходимо да се познават детайлите, вградени в процесора. Типовете данни, които непосредствено надграждат процесора, обикновено се приемат като *примитивни* (*атомарни*) *типове*. Няколко типични примера за най-често използвани типове данни в езиците за програмиране са дадени на таблицата от фиг. 2.

Име на типа данни	Примери на стойности
int (цяло число)	19; -100; 0; 29128
double (реално число)	12.345; 0.0; 0.982; 17.0
string (знаков низ)	„GS9032“; „Бързи криле“; „Мг.“

Фигура 2. Примери за типове данни

На ниво компютърна памет всички данни са представени като редица от нули и единици. Как ще бъде интерпретирано съдържанието на едно поле от паметта, зависи от типа данни, който е определен за съответното поле от паметта. Ето един пример.

Съдържанието на поле от 32 бита е: 00111111110000000000000000000000.

Стойността на полето е: 1069547520 като цяло число или 1.5 като реално число.

Езиците за програмиране разполагат и с езикови конструкции, с които от примитивните типове данни могат да се изграждат нови типове данни. Нари-

чат се *съставни* (*непримитивни*) *типове данни*, защото съдържат в себе си отделни компоненти. Те могат да бъдат както примитивни, така и съставни типове данни. Новите типове данни са едно още по-високо ниво на абстракция. За тях трябва да се опишат необходимите операции за достъп до компонентите на новия тип данни, които ще „скрият“ неговата структура.

Структура на данните

Важна характеристика на данните е тяхната структура. Ако се приеме, че типът данни представлява логическото описание (дефиницията) на някакви данни, то съответната им физическа реализация в конкретна система за програмиране представлява тяхната *структура от данни*. Може да се каже, *че структурата* на една данна се определя от нейния тип, и по-конкретно от:

– начина, по който се съхраняват съставните ѝ компоненти и от начина, по който са установени връзките между тях;
– реализираните операции за създаване, достъп до компонентите ѝ, търсене (извличане) и обновяване (актуализиране) на стойността на данната, като цяло, и на компонентите ѝ поотделно.

Данни, за които има изградена структура, е прието да се наричат *структурирани данни*.

Структурирани и неструктурирани данни

Да разгледаме два варианта на текстове, с които се съобщават данни за предстоящ полет (фиг. 3).

Пример 1		Пример 2
Номер на полет	GS 9032	*Излитане на* 22.10.2020 *в* 15:25
Авиокомпания	Бързи криле	часа *с полет номер* GS9032 *на*
Дата на излитане	22.10.2020	*авиокомпания „Бързи криле“*.
Време на излитане	15 ч. 25 мин.	*Продължителността на полета*
Време на кацане	19 ч. 30 мин.	4 *часа и* 5 *минути*.

Фигура 3. Данните в пример 2 са неструктурирани

Двата текста по същество имат едно и също съдържание. Но не може да се отрече, че първият вариант е по-ясен, възприема се по-лесно и по-бързо, отколкото вторият. Причината за това е в структурата на данните. Вторият вариант на текста е пример за *неструктурирани данни*. Като допълнение към тези примери може да се отбележи, че първият вариант е много добър за „разбиране“ и от компютърна програма, защото данните са записани като двойки от вида <*име на данна*, *стойност*>. За разлика от него програмната обработка на текста във втория вариант изисква анализ на текста, т.е. необходимо е данните да се *търсят* измежду останалите думи и след *разпознаване* на техния смисъл да се *извлекат* от текстовото съобщение. Ако текстът от втория пример подлежи на програмна обработка, то резултатът би трябвало да е близък до текста от първия пример. Но тъй като във втория пример е дадена само продължителността

на полета, за пресмятането на времето на кацане са необходими допълнителен анализ и пресмятане, за да се определи времето на пристигане. А това невинаги е тривиален проблем. В този и в подобни случаи, най-общо казано, се достига до необходимостта от разработване на програми с изкуствен интелект (ИИ), предназначени да решават проблеми от конкретна предметна област.

Данните могат да се класифицират по различни начини, като един от тях се отнася до тяхната структура. В тази класификация данните се разглеждат в три категории: *структурирани*, *полуструктурирани* и *неструктурирани*. Данните от пример 1 представляват полуструктурирани данни и те са разгледани в секция 5, а неструктурираните данни – в секция 6.

2. Структурирани данни

Приемаме, че данните са *структурирани*, ако те могат да бъдат описани със *система от правила*, които не са част от самите данни. Правилата включват описание на отделните компоненти, от които данните се състоят, както и начина, по който те са свързани помежду си. Всъщност със системата от правила се описва *структурата на данните*. Правилата трябва да позволяват конструиране на *алгоритми*, с които да се осъществява достъп до отделните компоненти на данните и да се извършват необходимите операции за обработката им. На практика множеството от алгоритми ще бъде *множеството от операции,* допустими за този тип данни.

2.1. Поле, запис, файл

В информатиката много често се използват три понятия, които пряко се отнасят до структурирането на данните. Това са понятията *поле*, *запис* и *файл*.

Пример. Разглежда се таблица, която съдържа следните данни за студенти: име, факултетен номер и среден успех (фиг. 4). Всеки ред от тази таблица е пример за данни от тип *запис*. Компонентите, от които е изграден записът, са три и се наричат *полета* на записа. Всяко поле има съответен тип данни (в случая те са примитивни) и съответно те са: име (знаков низ), факултетен номер (цяло число) и среден успех (реално число).

Име	Фак#	Успех
Иван Христов	902134	5.34
Мария Тонева	902346	5.22
Петър Иванов	902221	5.73

Фигура 4. Илюстративен пример на структурирани данни

Описаната таблица, съхранявана на външна памет, е пример за *файл* от най-прост вид. Казаното от по-горе може да се резюмира накратко така: файлът е множество от записи, а всеки запис е списък от полета.

Данните, структурирани като таблици, са лесно разбираеми за хората и компютърните програми, за които достъпът до тези данни е лесен и бърз. Това е

типичен пример за структурирани данни. Пример, за който всеки се сеща, са и електронните таблици. Следващото ниво в тази йерархия от понятия за данните е *базата от данни* (БД).

2.2. Бази от данни

В началото на 60-те години на миналия век малък екип от инженери и програмисти с ръководител Charles Bachman работи върху проект за автоматизация на бизнес процесите в компанията General Electric. Като резултат е създадена системата *The Integrated Data Store* (IDS), която е първата система за управление на бази от данни (Azalov, 1991). С това се поставя началото на една област в компютърните науки, която и до днес е актуална и продължава да се развива в различни направления.

Страници от историята: Charles W. Bachman

Charles Bachman (1924 – 2017) е инженер с бакалавърска степен от Michigan State University и с магистърска степен от University of Pennsylvania. За изключителния му принос към технологиите на базите от данни през 1973 той е удостоен с престижната награда ACM A.M. Turing Award[14]*. ACM* (*Association for Computing Machinery*) *е една от най-авторитетните световни асоциации по компютърни науки. C. Bachman е почетен член на Британското общество по компютърни науки, а в САЩ той е носител на Националния медал за технологии и иновации.*

С развитието на компютърните технологии понятието БД също еволюира. Представата от най-първите години за БД като понятие е, че тя е съвкупност от отделни файлове. Записите в тях съдържат полета, от които едно или няколко от тях представлява *ключ на записа*, с който еднозначно се идентифицира целият запис в рамките на файла. На този етап файловете са независими помежду си, като за всеки файл се разработват една или няколко приложни програми. При следващия етап от развитието на БД между файловете могат да се установяват връзки. Това улеснява създаването на нови приложения, защото в някои случаи необходимите данните могат да се извличат от един или от няколко вече съществуващи файла в БД.

Модели на данни

Най-общо моделът на данни (МД) е описание, което характеризира (определя) структурирането на данните в БД и включва:

– описанието на допустимите структури от данни;
– множеството от операции, с които могат да се обработват данните;
– множеството от правила, които гарантират непротиворечивост на данните и на връзките между тях.

През първите години са разработени и са използвани предимно два модела на данни: йерархичен и мрежов, а в края на 60-те години се появява и т.нар. *релацио-*

нен МД (РМД). Той е актуален и до днес и често допълван с нови възможности, отразяващи постиженията на информационните технологии и нуждите на бизнеса.

2.3. Релационен модел

Със създаването на РМД настъпва съществено нов етап от развитието на БД. Описанието му е изцяло извършено с математическия апарат и това позволява да се изгради стройна математическа теория. Основни обекти в РМД са таблиците, разглеждани като *релации*. На таблицата от фиг. 5 е дадено съответствието между термините, които се употребяват ежедневно за таблиците, и съответните термини от математиката, използвани в РМД. Според конкретния случай в настоящата статия се използват и двете терминологии.

Таблица	Релация
таблица с *n*-колонки	*n*-членна релация
име на колонка на таблица	атрибут на релация
ред от таблица с *n* колонки	*n*-торка (елемент на релация)

Фигура 5. Терминологични съответствия

При релациите данните във всяка отделна колонка са от един и същ тип, който се приема за тип на съответния атрибут. Операциите с релации се описват на език, известен като *релационна алгебра*. Така през 1970 г. се появява РМД. Негов автор е Edgar Codd. Още от първата му публикация *A Relational Model of Data for Large Shared Data Banks* (Codd, 1970) специалистите осъзнават големите възможности на идеята в РМД. Така започва един продължителен период на научни изследвания по цял свят.

Страници от историята: Edgar F. Codd

Edgar Codd (1923 – 2003) е англичанин, завършил е математика, а след това продължава с докторат по компютърни науки в *University of Michigan, 1965 г. Той започва изследванията си върху РМД като сътрудник на изследователската лаборатория на IBM в San Jose. E. Codd създава релационния език Alpha, който оказва силно влияние на следващия език QUEL. През 1981 г. е удостоен с ACM A.M. Turing Award. Носител е и на много други награди – на British Computer Society, на ACM SIGMOD, IEEE и др. Счита се, че Codd въвежда и термина On Line Analytical Processing* (*OLAP*).

Към релациите в една релационна база от данни (РБД) се предявяват определени изисквания. Част от тях са следните:

– всяка релация има уникално име;
– имената на атрибутите на всяка отделна релация са различни;
– всяка „клетка“ на релацията съдържа точно една данна и тя е от примитивен тип;

– елементите на всеки ред са свързани по някакъв начин помежду си, а типът им е определен от типа на съответните атрибути;
– всички елементи (редове) на релацията са различни.

Структурата на отделната релация се описва с т.нар. *релационна схема*. Тя се състои от името на релацията и списък на нейните атрибути, придружени от съответния тип данни. Съвкупността от схемите на всички релации се приема за *схема* на цялата РБД.

Релационни езици

Релационните езици се разделят на две основни групи: *език на релационната алгебра* (РА) и *език на релационното смятане* (РС). РА е процедурен език. Чрез него стъпка по стъпка се описва изчислителен процес с операции от РА. Този постъпков процес продължава до получаването на необходимия резултат. РС е език от по-високо ниво. При него крайният резултат, който трябва да се получи, е релация, която се записва като израз от предикатното смятане.

Страници от историята: Езикът SQL

Езикът SQL (Structured Query Language) е част от „златната“ история на РБД. Негови автори са Donald Chamberlin и Raymond Boyce. И двамата са сътрудници на IBM.

> D. Chamberlin (1944 г.) има докторска степен по инженерни науки от Stanford University. През 1997 г. получава престижната награда ACM SIGMOD Edgar F. Codd Innovation Award. R. (1947 – 1974) има докторска степен по компютърни науки от Purdue University. Съвместно с D. Chamberlin разработват езика SEQUEL, по-късно преименуван в SQL. Заедно с E. Codd формулират поредната нормална форма в процеса за нормализация, известна като Boyce-Codd Normal Form (BCNF).

Езикът SQL се развива много интензивно през годините, като се разширява и преминава през редица варианти. В тях се отразяват новите идеи в базите от данни. Така се създават множество стандарти (ANSI 1982, 1989, 1992,1999, 2003, 2008, 2011). Едно важно разширение се отнася до OLAP (On Line Analytical Processing). SQL има ядро (Core SQL), за което се предполага, че се поддържа от всички РБД.

3. Складове от данни

Технологията на складовете от данни (*Data Warehouse*) се появява в началото на 80-те години и за неин идеолог се приема William H. Inmon. Първата подобна система е създадена от IBM, в основата на която е релационната система DB2.

Идея за склад на данни

Пример. Ръководството на една голяма компания е в процедура за усъвършенстване на дейността, най-вече в областта на продажбите на продукцията си. Тя се нуждае от отговори на въпроси, като: (1) какво е количест-

вото на продадените стоки през текущата година по отделните клонове на компанията; (2) защо продажбите от миналата година са били по-добри от тези на текущата година; (3) с колко процента биха се увеличили продажбите през следващата година и т.н. Компанията има множество клонове в страната и по света. До настоящия момент всеки клон на компанията поддържа своя БД, регулярно предава отчетите си на по-горното ниво на компанията. За да се автоматизира цялата тази дейност, от която да е възможно получаването на отговори на въпроси като тези от по-горните примери, е необходима нова система. Решението на проблема в общия му вид води до разработването на складове от данни в различни области на бизнеса и държавните агенции.

Складът от данни (СД) е „хранилище“, в което организацията съхранява всичките си данни. Те могат да се събират от различни източници и да бъдат с различни релационни схеми. В някакъв смисъл данните в склада имат исторически характер, т.е. съхранените вече данни не се променят, а в склада постъпват само нови данни. Данните са класифицирани по определени признаци и са предназначени за извършване на анализ. Типични операции с данните от склада са *извличане*, *трансформиране* и *зареждане*, известни като ETL (Extract, Transform, Load) (Connolly et al, 2015). По-долу тези операции са описани по-подробно.

а) *Почистване на данните*. Данните, които обикновено постъпват за съхраняване в склада, често са непълни, могат да са в различни мерни единици, а понякога могат и да са неточни. Това изисква внимателното им „оглеждане“ преди да постъпят за съхранение в склада.

б) *Интегриране на данните*. Това е трудна процедура, първата стъпка на която се състои в *идентификация* на атрибутите на данните в отделните БД. Не може да се изключи случаят на редундантност на данните, при който стойностите на един атрибут могат да се получат от стойностите на един или няколко други атрибута.

в) *Редуциране на данните*. Най-вече от съображения за намаляване на общия обем на данните в склада данни, които няма да се анализират, се отстраняват в процеса на обединението.

г) *Трансформиране на данните*. Тази стъпка включва изглаждане, обобщаване, нормализация и дискретизация на данните (Thomas et al, 2016).

д) *Изследване на данните*. Цел при съхраняването на данните в склада е тяхното изследване, от което се очаква да се дават отговори на важни за компанията въпроси. То включва дейности, които най-общо могат да се обединят в три групи:

– планиране и подготовка на данните, подлежащи на изследване;

– анализи на данните (има няколко вида);

– обобщаване и визуализиране на резултатите.

4. Обектни модели на БД

Наред с многобройните си предимства и реални успехи в практиката РМД няма претенции да е универсален, и съвсем естествено това води до разработването на нови МД. Офис системите, управлението на бази от документи, системите за съхранение и обработване на изображения, географските системи, CAD/CAM системите, мултимедийните системи и други са все примери, при които са необходими данни със сложна структура. Но такива данни са недопустими в РМД. Хомогенността на данните във всяка колонка, също и на редовете в цялата релация е основно изискване при релациите. Допълнително ограничаващо изискване е данните във всяка клетка на таблицата да са атомарни. Броят на атрибутите в релациите е фиксиран във всяка релационна схема, а промяната му е твърде „скъпа“ операция (Connolly et al, 2015). Тези ограничения и неудобства създават затруднения при използването на РМД в много от новите приложения. Така през 1985 г. се появяват и първите идеи и статии за обектно ориентираните БД (ООБД).

Страници от историята: В света на данните и обектите

Терминът „обектно ориентиран“ се появява в края на 50-те и началото на 60-години на миналия век. За автор се приема Alan Kay (1940 г). През 1969 г. той получава докторска степен по компютърни науки. A. Kay има изключително много постижения в науката и през 1983 г. става носител на ACM A.M. Turing Award:

> …За идеите му в съвременните обектно ориентирани езици за програмиране, ръководене на екипа, разработил Smalltalk, и за фундаментален принос към персоналните компютри.

Идеята на ООБД е заимствана от обектно ориентирания подход, който през 80-те години вече е широко навлязъл в езиците за програмиране. Необходимостта от нов подход в БД събира представители на заинтересовани компании, работещи в областта на БД, като измежду тях са: American Airlines, Canon, Data General, Hewlett-Packard, Philips Telecommunications, Sun Microsystems и други. Така през 1991 г. се сформира работна група с ръководител Christopher Stone, наречена Object Data Management Group (ODMG). В един от първите си материали групата е записала:

> Ние дефинираме ODBMS като СУБД, която интегрира възможностите на базата от данни с обектно ориентирани възможности на езика за програмиране.

През 1998 г. ODMG променя името си в Object Data base Management Group (ODBMG), за да отрази по този начин и подхода на обектно релационните БД. Между 1993 г. и 2001 г. работната група създава няколко документа. Последният от тях е със заглавие *The Object Data Standard:ODMG3.0* (Cattell et al, 2000), след което тя се разпуска.

Обекти и литерали

Обектите и литералите са основните градивни елементи на обектните модели.

Обектът се характеризира с идентификатор, състояние и поведение. Всеки обект има идентификатор, който е уникален в рамките на цялата БД. Състоянието на обекта се определя от текущите стойности на съставящите го величини, а поведението му – от операциите, които могат да се прилагат над тях.

Литералът има стойност, но няма идентификатор. Стойността му може да бъде проста (атомарна: int, double, string и други), структурирана и колекция от литерали или обекти (сложни литерали).

Сложни обекти

Сложните обекти и *литерали* са изградени от по-прости, като към тях се прилагат *конструктори* (Balcilhon et al, 1989), (Azalov, 1991). С тях могат да се „конструират“ сложни категории от данни като множества, масиви, *n*-торки, списъци и други. Чрез множествата се изграждат СД, представляващи съвкупности от еднотипни данни (обекти, литерали). Масивите позволяват да се въведе някаква наредба в една съвкупност от еднотипни данни. Чрез *n*-торките добре се описват разнообразните (от различни типове данни) свойства на моделираните обекти. Списъкът, като структура, позволява да се представят еднотипни данни, без да се фиксира техният брой. Съществено е да се отбележи, че се допуска влагане на конструктори. Например тази идея позволява да се дефинира релация, чиито елементи са релации. На фиг. 6 е даден пример на сложен обект (ненормализирана релация), който представлява множество от две *n*-торки, в които вторият атрибут е множество.

Учебник	Автор (множество)	Издателство
Data on the Web: From Relations to Semistructured Data and XML	{S .Abiteboul, P. Buneman, D. Suciu}	Morgan Kaufmann
Big C++	{C. Horstmann , T. Budd}	John Wiley & Sons, Inc

Фигура 6. Множество от *n*-торки, вторият атрибут на които е множество

Във фиг. 7 релацията съдържа същите данни като тези от фиг. 6, но тази таблица има само прости атрибути, т.е. тя е *нормализирана релация* и е приведена към изискванията на РМД.

Име на учебник	Автор	Издателство
Data on the Web: From Relations to Semistructured Data and XML	S . Abiteboul	Morgan Kaufmann
Data on the Web: From Relations to Semistructured Data and XML	P. Buneman	Morgan Kaufmann
Data on the Web: From Relations to Semistructured Data and XML	D. Suciu	Morgan Kaufmann
Big C++	C. Horstmann	John Wiley & Sons, Inc
Big C++	T. Budd	John Wiley & Sons, Inc

Фигура 7. Релация, в която всички атрибути са прости

Обектно ориентираната концепция в езиците за програмиране е вградена в два нови подхода при моделите на БД (Connolly et al, 2015):

– *Обектно ориентиран МД* (ООМД) – характеристиките на този модел се определят основно от съответния обектно ориентиран език за програмиране (Cattell et al, 2000);
– *Обектно релационен МД* (ОРМД) – основната идея на този модел е разширяването на РМ с възможности за оперирane и с обекти.

Страници от историята: Michael Stonebraker

До голяма степен развитието на ОРМД се основава на резултатите на изследователската група по бази от данни в Калифорнийския университет в Бъркли с ръководител Michael Stonebraker.

Michael Stonebraker (1943) завършва бакалавърска степен по електроинженерство в Princeton University, а магистърска и докторска степен (1971 г.) по компютърни науки – в University of Michigan. Голяма част от научната му кариера протича като професор в University of California at Berkeley. Носител е на награди от най-висок ранг, измежду които са: ACM System Software Award, ACM SIGMOD Innovation Award, National Academy of Engineering Award, IEEE John von Neumann Medal. За фундаментални приноси в концепциите и практиките на съвременните системи за БД през 2014 г. получава ACM A.M. Turing Award.

5. Полуструктурирани данни

През последните 30 години силно нарасна разнообразието от нови категории от данни. Както вече беше посочено и по-горе, релационните схеми, с които данните до този момент са данни, „стегнати в таблични рамки“, се оказват твърде „тесни“ за новите категории от данни. Тези данни вече „очакват“ своите модели – подходящи структури на представянето им и съответни алгоритми за обработката им. Така вниманието на специалистите се пренасочва към „новите данни“, наречени *полуструктурирани* и *неструктурирани* данни.

Какво представляват тези данни и защо са важни? Ще ги представим поотделно, тъй като те представляват съществени жалони по пътя към големите данни.

Идея за полуструктурирани данни

Основна характеристика на структурираните данни е възможността за пълното описание на структурата (схемата) им. Това ги отличава съществено от полуструктурираните данни (*semistructured data*).

Полуструктурираните данни (ПСД) се определят като (Abiteboul et al, 1999):

– данни, върху които няма наложена структура (схема) и по тази причина те не спадат към структурираните данни или като

– данни, чиято структура се съдържа в самите тях, т.е. те са *самоописващи се данни*.

Интересът към полуструктурираните данни нараства през последните две десетилетия. Причините са разнообразни и някои от тях са споменати по-горе, но има и други.

Много от уебстраниците на глобалната мрежа днес се използват като източници на данни за различни СУБД. Естествено данните от тези източници, които са твърде разнообразни, трудно могат да бъдат сложени „под шапката“ на една и съща релационна схема. Като допълнение може да се отбележи, че форматът на ПСД се оказва твърде удобен за такъв обмен на данни между съществуващи БД и днес това е вече добре установена практика. Ето защо измежду основните функции на съществуващите релационни системи са и тези, с които се експортират и импортират данни, например в XML формат, които са ПСД.

Пример. Разглеждаме данните на Петър от град Рила, а телефонът му е с номер 8834891. Тези данни можем да „кодираме“ по следния начин:

{*име*: „Петър“, *тел*: 8834891, *град*: „Рила“}

От примера се вижда, че данните са записани като множество от елементи, всеки от които е двойка от вида *n*: *v*, в която с *v* е означена стойността на данната (знаков низ или число), а с *n* – името ѝ (етикет).

Това е един конкретен пример, но в някои случаи данните за други лица могат да бъдат твърде разнообразни, по-сложни и практически да имат различна структура. Да отбележим, че идеята за представяне на данните като двойка „*име*: *стойност*“ предоставя големи възможности. Това е универсален подход за структуриране на множество от данни, без да се изисква да бъдат с една и съща схема. Съществено е да се отбележи, че самата структура е част от самите данни. На фиг. 8 е даден пример за данните от адресната регистрация на Христо и Иван.

Регистрация	
адрес:	{ *име*: {*собствено*: „Христо“, *фамилно*: „Тонов“}, *град*: „Рила“, *улица*: „Хр. Ботев 123“, *тел*: 8976152, *тел*: 8971098, *пощенски_код*: 2098}
адрес:	{ *име*: {*собствено*: „Иван“, *фамилно*: „Петров“}, *град*: „София“, *комплекс*: „Слатина“, *блок*: „22, вход А“ *пощенски_код*: 1572, *ел._поща*: „IPetrov@xyz.pqr“ }

Фигура 8. Пример за множество от ПСД

Данните от тези примери не могат да се опишат с една и съща релационна схема, защото са с различни атрибути, а има и повторение на някои от атрибутите. Въпреки тези особености, които са недопустими за релационните схеми, възприетият начин е напълно приемлив за представянето на подобни ПСД. Това е част от силата на техните възможности, което позволява с лекота да се представят данни с „по-свободен“ формат.

Идеята за ПСД има разнообразни реализации. XML технологията е една тях.

XML

XML е съкращение на английското му именование *eXtensible Markup Language*. С популярното мнение, че XML е *език за данни*, а C++ е *език за програмиране*, напълно се разкрива същността на езика XML. Данните, представени в XML формат, са текстови файлове, които обикновено се наричат XML документи. Всеки XML документ съдържа текстови данни, като едновременно с тях е „вградена“ и структурата им, а това напълно оправдава смисъла на „уравнението“: XML = Data + Structure.

Всеки XML документ се състои от отделни елементи, между които има установени връзки. *Елементът* в XML е логическа единица от данни, чиято структура се състои от три части, които са:

– начален таг (отваряща скоба). Пример <name>;
– краен таг (затваряща скоба). Пример </name>;
– стойността на елемента е това, което се съдържа между двата по-горни тага.

Пример за елемент със стойност „Христо Тонев“:
<name> Христо Тонев </name>

Елементите могат да имат поделементи, като например елемента <студент> в примера от фиг. 9. Неговите поделементи са: *name*, *addr*, *tel*, *fax*, *email*. Трябва да се отбележи, че списъците от поделементите на отделните студенти са различни.

```
<CSstudents>
<student>
<name>Христо Тонев</name>
<addr>Рила 2098</addr>
<addr>Хр. Ботев 12</addr>
<tel>879761525</tel>
<fax>878976541</fax>
<email>ht123@fmi.edu</email>
<email>alo99@xyz.com</email>
</student >
<!--Продължение -->
<student>
<name>Марин Анев</name>
<addr>Сливен 4502</addr>
<addr>Сините скали 36</addr>
<tel>888325719</tel>
<tel>887454164</tel>
<email>MH98@fmi.edu</
email>
</student >
</CSstudents>
```

Фигура 9. Пример за XML текст

На пръв поглед, примерният XML документ има твърде неясна структура и това е вярно. Няма го строгия формат на структурираните данни. Но неяснотата в структурата на данните на двамата студенти е привидна, защото тази структура се определя от правилата, заложени в граматиката на този XML текст. В случая тя представлява следният регулярен израз:

name, addr+, (tel | fax)*, email+

Този израз определя данни с твърде разнообразна структура, която е последователност от елементите: точно едно име (*name*), поне един адрес (*addr*), нула или няколко повторения на телефон (*tel*) и/или факс (*fax*) и накрая поне един електронен адрес (*email*).

В общия случай XML данните не могат да се описват с релационни схеми. За описанието им се използват т.нар. DTD файлове (Document Type Definition). Това са текстови файлове, които съдържат граматиката на конкретния XML език. Друга, по-добра възможност е езикът XSD (XML Schema Definition). Интересно е да се отбележи, че самият XSD език е също XML език. Всъщност трябва да се каже, че XML е *метаезик*, т.е. той е език за описание на синтаксиса на други езици. С него са създадени стотици езици, като например: MathML (*Mathematical Markup Language*), VML (*Vector Markup Language*), MML (*Music Markup Language*), GPX (*Transferring GPS data between software applica*tions), Biml (*Business Intelligence Markup Language*) и др.

JSON

XML е масово използван език. Но има и други езици, които оперират с ПСД. При това те са не по-малко използвани и в някакъв смисъл са по-прости. Един такъв език е JSON (Tom, 2017). Наименованието му е съкращение от *Java Script Object Notation*. И той като XML е „самоописващ“ се език, в който елементите се представят с двойка от име и стойност.

Пример: „name“: „Христо“

Елементите допускат подлементи. Файловете на JSON са текстови файлове, които са лесни за разчитане от хората и са удобни за програмна обработка. Всичко, посочено дотук за JSON, е твърде сходно с това за езика XML. В JSON има разнообразни типове данни, като могат да се използват и масиви.

Пример: „names“: [„Христо“, „Мария“, „Петър“, „Марин“]

На фиг. 10 е даден пример, в който едни и същи данни са записани във форматите на XML и JSON.

XML формат	JSON формат
<Books>	{
<book>	"book": {
<title>Data on the Web</title>	"title": [
<title>From Relational	"Data on the Web",
to Semistructured Data andXML	"From Relational to Semistructured Data and XML"
</title>	],

```
<author>Serge Abiteboul</author>
<author>Peter Buneman</author>
<author>Dan Suciu</author>
<year>2000</year>
<publisher>MorgnKaufmann</publisher>
</book>
</Books>
```

```
"author": [
 "Serge Abiteboul",
"Peter Buneman",
"Dan Suciu"
],
"year": "2000",
"publisher": "Morgn Kaufmann"
}
}
```

Фигура 10. Представяне на един и същ текст чрез XML и JOSN

6. Неструктурирани данни

Ако от всички съществуващи данни се изключат структурираните данни и ПСД, тези, които остават, са *неструктурирани данни*.

А кои са те? По-горната дефиниция е твърде кратка и не може да се каже, че не е разбираема и точна. Но не може и да се отрече, че тя не е особено полезна и не изяснява добре какво са неструктурираните данни. Всъщност дефинирането на това понятие е трудно. Затова е добре да се започне с примери.

Примери за неструктурирани данни

Най-общо неструктурираните данни се класифицират в две групи.

– Данни, създадени от човек:
  – *офис документи*: текстови файлове, PDF файлове, презентации и други;
  – *електронна поща*: само част от данните се състои от структурирани данни. Текстовото поле е неструктурирано и се поддава на анализ трудно;
  – *мултимедийни файлове*: MP3, аудио и видео файлове, цифрови снимки (JPEG, GIF, PNG) и др.;
  – *данни от социални медии*: съдържащи се във Facebook, Twitter и др.;
  – *комуникации*: телефонни разговори и записи на телефонния секретар и др.
– Данни, генерирани от електронни устройства:
  – *цифрово наблюдение*: снимки, видеа от наблюдения и др.;
  – *сателитни изображения*: чрез Google Earth, Geo SCAN (целогодишно заснемане на полето с култури), данни за времето и др.;
  – *научни данни*: сеизмични изображения, данни от проучване на Космоса, от мигрирането на диви животни и др.;
  – *сензорни данни*: за наблюдение на трафика, океанографски и др.

Няма съмнение, че спрямо „обикновените“ данни посочените по-горе видове са от съвсем друго естество. В общия случай не се познават универсални

алгоритми за обработване на неструктурирани данни, въпреки че има немалко успешни „пробиви“ в някои конкретни области.

Обработване на неструктурирани данни

Вниквайки по-задълбочено в примерите за неструктурирани данни, могат да се забележат някои от характерните им черти.

В общия случай неструктурираните данни трудно могат да бъдат декомпозирани на компоненти, които да са „разпознаваеми“ за обикновените СУБД. Например данните за един студент могат лесно да се структурират в няколко полета, като: име, факултетен номер, среден успех и т.н., но да се декомпозира една снимка, чийто компютърен формат е „дълъг низ“ от нули и единици, изобщо не е тривиална процедура. Ето защо операциите актуализиране, изтриване и търсене в общия случай са трудни, а някои от тях са частично или напълно невъзможни.

Възможностите за съхраняване и обработване на неструктурирани данни значително нараснаха през последните години. Един подход, който става все по-ценен като начин за получаване на информация с по-висока бизнес стойност от неструктурирани данни, е *аналитиката на текста*. Това е процедура за анализ на неструктуриран текст с цел извличане на подходящи фрагменти от текста и трансформирането им в структуриран текст, който след това може да се използва по различни начини. Например, когато Google прочете нова страница, той започва да я опознава, като анализира не само текстовото ѝ съдържание, но и графичните изображения и видеа, които са част от нея. Той може да „разбере“ някои от тях, но не така добре, както би разбирал текста[4)].

7. Ерата на големите данни

Преди да започнем с представянето на основните сведения за големите данни, е добре да се ориентираме в таблицата на големи числа, които се използват като мерни единици за обем на компютърна памет (фиг. 11).

Име	Размер в байтове	
Yottabyte (YB)	1, 208, 925, 819, 614, 629, 174, 706, 176	$\approx 10^{24}$
Zettabyte (ZB)	1, 180, 591, 620, 717, 411, 303, 424	$\approx 10^{21}$
Exabyte (EB)	1, 152, 921, 504, 606, 846, 976	$\approx 10^{18}$
Petabyte (PB)	1, 125, 899, 906, 842, 624	$\approx 10^{15}$
Terabyte (TB)	1, 099, 511, 627, 776	$\approx 10^{12}$
Gigabyte (GB)	1, 073, 741, 824	$\approx 10^{09}$
Megabyte (MB)	1, 048, 576	$\approx 10^{06}$
Kilobyte (KB)	1,024	$\approx 10^{03}$

Фигури 11. Таблица на големи числа

7.1. Терминът големи данни

Големите данни са огромни масиви от данни, които могат да бъдат структурирани, полуструктурирани и неструктурирани данни или комбинации от

тях. В общия случай те се създават и се генерират от разнообразни източници, твърде бързо нарастват в големи обеми и не могат да бъдат обработвани с традиционните СУБД. Накратко техните характеристики често се определят като данните, които са характерни с трите *V*: *обем* (Volume), *скорост* (Velocity) и *разнообразие* (Variety). За тези три характеристики се споменава за първи път в началото на 2001 г. от Doug Lancy (Lancy, 2001).

> … управлението на данните в електронната търговия експлодира в три направления: обем, скорост и разнообразие.

Обем на данните

В повечето от случаите неструктурирани данни изискват голям обем от памет за съхранение. Да разгледаме електронните писма, изпратени през 2019 г. Техният брой за *една минута* е бил 188 милиона[15]. Да пресметнем и количеството от данни, което е било необходимо за съхранението им за цялата година. Общият брой на всички писма за цялата година е бил 106945000000000. Средният размер на едно писмо е 75K (в Google има твърде много данни!). Оттук следва, че общият обем в байтове на всички писма през 2019 г. е бил около 8×10^{18}. Това е обем в диапазона на екзабайтовете (фиг. 11). Да отбележим още няколко впечатляващи числа, които отразяват обема на данните, генерирани за *една минута* през 2019 г.[15]:

– 3.8 милиона обръщения към Google за търсене на информация;
– 4.5 милиона обръщения за видеа в YouTube;
– 41.6 милиона обръщения, изпратени чрез Whats App и Facebook Messenger.

Данните в тези примери са неструктурирани. Всъщност анализите на редица информационни агенции по света показват, че неструктурираните данни представляват не по-малко от 80% от всички данни, които се съхраняват сега.

Скорост на генерирането на данните

Със скоростта се измерва времето, за което се създават данните. Като се има предвид огромният обем от данни, който се генерира във всеки момент, фактически това е сравнимо с „потоп“ от данни, който всяка секунда „залива“ сървърите. Ето защо важно изискване към технологията на големите данни е този голям обем от данни да може да бъде поет и съхранен за приемливо време.

Разнообразие на данните

Разнообразието в големите данни се отнася до възможността за съхранение и пълно или частично обработване на данни от всички категории. Това означава, че данни от различни типове и формати, които се срещат в структурираните данни (като например финансови трансакции), в полуструктурираните данни (като например XML документи) и в неструктурирани данни (като например графични изображения) трябва да могат да бъдат обработвани по съответен начин.

С времето към трите *V* се добавиха още много други. Две от тях са *достоверност* (Veracity) и *стойност* (Value) на данните.

Достоверност на данните

Изискването за достоверност се отнася до верността на данните, т.е. до тяхното качество. То може да се изразява в непълнота, неяснота, несигурност или приближения. Интересни са въпросите: „*Колко точни са данните при оценка на стойността на даден бизнес*? *Дали резултатите от анализа на големите данни всъщност имат смисъл, ако има неточност в данните*?“. За разлика от данните, които пристигат по „вътрешните канали“ на системата и предварително са проверени, повечето от големите данни идват от източници, които трудно могат да бъдат контролирани и следователно е възможно да има проблем с точността им. Това изискване е важно за повишаване на доверието към новопостъпващите данни и изисква прилагане на съответни технологии и алгоритми.

Стойност на данните

Стойността да данните се определя от тяхната полза. Тази характеристика е в зависимост от истинността им и от необходимото време за обработката им. Данните с по-висока достоверност са по-ценни за бизнеса. Но ако обработката за достоверност изисква 3 секунди, то това време може да се окаже твърде дълго. Само като пример да се мисли за фондовия пазар, при който времето се измерва в части от секундата.

Големите данни са „надарени“ с много дефиниции, които често се допълват и уточняват, но повечето от тях гравитират около *V*. А каква е етимологията на този термин? Едно изследване по този въпрос е публикувано в *The Origins of ‚Big Data‘: An Etymological Detective Story*. В статията се казва, че бащата на термина *big data* вероятно е John Mashey[8)], който през 90-те години е бил главен сътрудник в компанията Silicon Graphics[9)]. John Mashey (1946 г.) е с богата професионална кариера и е с докторска степен по компютърни науки от Pennsylvania State University[10)].

7.2. Срещи с големите данни от ежедневието

„Обикновените данни“ от десетки години „съжителстват“ с нас. Това са нашите данни за кредитните ни карти, за банковите ни сметки, за автомобилите ни, данните от трансакциите при пазаруването в търговските вериги, личните и геномните данни на хората, данните за здравното и пенсионното ни осигуряване, за застраховките на имуществото ни и още много др. Всички тези данни, въведени в компютърната памет, са структурирани. Това са данни от историята на ежедневието ни. Те се създават бавно, но с годините се натрупват огромни обеми от тях и представляват източници на огромен обем от информация.

Бизнес с вкус на кафе

Световна компания, която притежава хиляди кафенета за сервиране на кафе, е в подготовка за предлагане на нов продукт. Естествено, предварител-

ното проучване на пазара е извършено, но истинският резултат се разбира едва с пускането на продукта в продажба. Следва сбит преразказ в свободна форма, заимстван от (Watson et al, 2014).

Още от ранната сутрин в деня, когато кафенетата на компанията вече приятно ухаят на новото кафе, нейните аналитици активно натрупват данни за новия продукт. Те наблюдават и проучват блогове, Twitter и дискусиите на групи във форум за кафе и получават първите реакции и оценки от клиентите. Ръководството на компанията е доволно, че кафето се харесва, но разбира, че цената е висока. Следва бърза реакция за понижаването на цената и до края на деня отрицателните коментари са преустановени.

Доколко бизнесът държи на бързината на услугите, които предлага, може да се почувства и от следващата бележка. Докато пътувате с колата към кафенето, може да направите поръчката си онлайн. Приложението, инсталирано на вашия iPhone, ще намери най-близкото кафене в района, ще предаде поръчката и ще извърши плащането с кредитната ви карта. Когато пристигнете, чашата с кафето, на която е залепен етикет с вашето име, вече ви очаква. Вашите данни и предпочитанията ви са съхранени в „облака“ на компанията и тя е в пълна готовност да ви обслужва отново занапред само с натискането на 2 – 3 бутона.

Онлайн магазини

Много хора прекарват с часове пред екраните, разглеждайки интернет сайтове. За момент да си представим, че искаме да пазаруваме от някакъв виртуален магазин. Такива магазини има хиляди по света, но като пример да си мислим за Amazon. За да се ориентираме към определен вид стока, обикновено се налага да се разгледат няколко страници. Но докато ние гледаме, четем и кликваме върху страниците на сайта, едни „невидими и интелигентни“ алгоритми разглеждат нас. Нашето „търговско досие“ е отворено още в момента, в който сме „прекрачили“ входа на магазина. Това е само началото, защото тези алгоритми продължават да се интересуват от нас през цялото време: колко време разглеждаме отделните категории продукти, какво купуваме накрая, на каква цена, в какъв брой и цвят, какви размери, колко харчим на месец и т.н.

Какво харесваме и колко харчим, са много важни данни за магазина, защото това е неговият бизнес и той иска ние да купуваме от него повече неща. В един момент не може да не забележим, че интелигентните алгоритми стават все по-видими и започват да се „държат“ твърде любезно с нас. „Усещайки“ към какво проявяваме интерес в момента, а и от предишните „разходки“ из магазина, те ни подсещат с предложения за още подобни стоки. Например при търсене на книги по темата Databases & Big Data се появяват десетки заглавия на книги със съответните снимки на кориците и много данни за тях. Ако се спрете и кликнете върху някои от тях, ще получите нови предложения,

придружени с текста *Inspired by your browsing history*. Вече е ясно, че вие сте под прожекторите на „интелигентните алгоритми“, които ще продължават да ви „ухажват“ с нови предложения от вида: *Customers who viewed this item also viewed* или *Frequently bought together* и т.н.

А сега да обобщим, че в случая за интелигентните алгоритми от особена важност е потребителското поведение. Събраните данни за потребителите са източниците, от които те „черпят знания“ и ги превръщат в пари. Тук е мястото да се постави въпросът „*Откъде магазинът знае какви книги да ни предложи?*“. Като отговор, да продължим с още един кратък пример. Решили сте да си купите градински маркуч за поливане. Без съмнение, ще ви бъдат предложени и пръскачки с различен брой опции за поливане. Алгоритмите няма откъде да знаят дали вие имате нужда от пръскачка за маркуча, но те знаят добре, че 80% от купувачите на маркучи купуват и пръскачки. Нека и сега да се опитаме да изкажем по-общо тази теза. Алгоритмите стават все по-интелигентни, защото те постоянно се *самообучават* от поведението и решенията на клиентите. А обучението на алгоритмите чрез данни (*Data Learning*) е една от важните дисциплини на науката за данни.

Ето защо много компании се интересуват от нашите данни. Компаниите са много, трансакциите са още по-много и така данните се трупат във всеки един момент, като постепенно нарастват до размерите на големите данни.

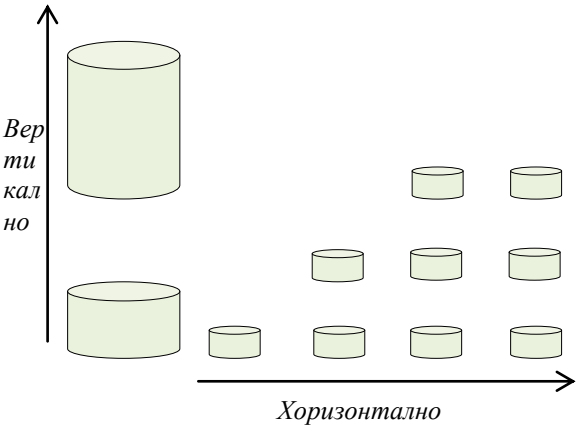
Кредитни карти

В представения сценарий за онлайн магазини има и „скрит“ участник. Заплащането се извършва обикновено с кредитна карта. Трансакцията, която се обработва, изисква редица операции и като резултат кредитната ви сметка ще бъде обновена, а с това се актуализират и данните, отнасящи се до историята на вашето кредитиране. Кредитирането е още един източник на данни с голям обем. Съответните данни се използват не само за отделните хора, но и за получаване на обобщени резултати относно кредитирането в различни сектори, включително и на държавно ниво.

При плащането с кредитна карта е възможно системата да откаже да извърши трансакцията. В подобни случаи е ясно, че моделът на обслужване, вграден от „интелигентните алгоритми“ на компанията, възприема плащането с картата за ненадеждна операция. Защо? Причините могат да бъдат различни. Алгоритмите „знаят“ твърде много за притежателя на кредитната карта и всяко отклонение от неговото поведение (например сумата е твърде голяма, плащането е твърде далеч от адреса на местоживеенето и др.) е сигнал за повишаване на вниманието. Случаят на отказ е такъв, който първоначално не се вписва в модела, реализиран в алгоритмите, базирани на вашите данни. Но по-важното в случая е това, че алгоритмите ще „запомнят“ данните от този случай, което е част от процеса на „самообучаване“ и усъвършенстване на модела за вземане на решение.

7. 3. Софтуерни и хардуерни технологии за големи данни

Големите обеми от данни изискват нови методи за съхранение и обработване. Тук трябва да се отбележи, че характеристиката „обем“ е в силна зависимост от наличните в момента технологии – хардуер и софтуер. Днешните големи данни навярно след години ще бъдат вече „средни“ по размер, а след още години ще станат „малки“. Преди десетилетия обемите от данни от порядъка на ТВ бяха немислими. Сега много компании поддържат БД с такива обеми, а голямата борба е вече в зоната на екзабайтовете (EB).



Фигура 12. Илюстрация на вертикално и хоризонтално мащабиране на разпределена система

Пътят към големите данни не беше възможен без съответния „голям“ хардуер и софтуер. Те се появиха от необходимостта за решаване на задачи, които не бяха по силите на традиционните системи в края на 90-те години. Основна роля изиграва развитието на технологиите, от които хардуерът поевтинява. Това позволява изграждането на големи разпределени системи. Използвани са два подхода за увеличаване на мощта на компютърните системи при разширяване – вертикално и хоризонтално мащабиране (фиг.12). *Мащабирането* е способност на система да запазва производителността и мощността си при нарастване на обема на данните, което е от решаващо значение за големите данни.

При *вертикалното мащабиране* нарастването на ресурсите става на един възел от системата – увеличават се броят на процесорите, разширява се дисковата памет и т.н. В този вариант разходите са големи, а и увеличаването не е без граници.

Идеята при *хоризонталното мащабиране* е да се добавят повече, но евтини компоненти (commodity hardware) към системата. Освен приемливите финансови разходи хоризонталното разширение позволява по-добро разпределение на натоварването върху множество от сървъри, вместо да се разчита на един високоскоростен сървър с голям капацитет. Този подход е по-гъвкав за внедряване, по-надежден и по-лесен за поддръжка.

Разпределени файлови системи

Нека за момент се замислим за система, която обслужва милиони клиенти в една социална мрежа. Данните, които системата поддържа, са структурирани в ориентиран граф, включващ стотици милиони възли и много милиарди дъги. Това е един типичен пример, в който алгоритмите в такава система оперират с големи данни. И сега се появява естественият въпрос „*Как системата се справя с огромния обем от данни и как генерира толкова бързи отговори на потребителските заявки*?

Отговорът на този въпрос идва от новите хардуерни и софтуерни решения. Новите системи работят върху т.нар. *компютърни клъстери*. Всеки клъстер представлява група от компютри (*възли*), свързани помежду си в мрежа. Възлите са независими и всеки от тях работи под уравнението на своя операционна система, без да споделя ресурсите си с други възли. Така се осигурява по-голяма изчислителна мощ и по-лесно се преодоляват сривове в някои от компютрите. Да отбележим, че по този начин натоварването на компютрите е балансирано и най-важното – обработването на данните може да се извършва паралелно от много компютри. Това би било възможно само ако е инсталиран съответен софтуер. На първо място, това е *разпределената файлова система* (РФС), чиято цел е да поддържа много на брой и големи файлове. Файловете в РФС са разделени на блокове, които също са с голям размер – от порядъка на МВ. За да се избегне загубата на данни, например при срив на един компютър, блоковете от файлове се дублират върху два или три други компютъра, но в рамките на други клъстери.

Информацията за разположението на блоковете на всеки файл и на техните копия се съдържа в специален възел, наречен *главен възел*, на който също се поддържа съответно копие.

Файловете с данни са най-ценният елемент във всяка файлова система. Ето защо средствата, с които се осигурява тяхната наличност, т.е. да няма загуба на данни, са с много висок приоритет. Разбира се, това има своята цена, а тя е наистина голяма.

Да приемем, че мрежата на една компания разполага с около един милион възела. Средната издръжливост на една машина е около 3 години, т.е. около 1000 дни. Това означава, че всеки ден ще се налага една от машините да бъде подменена.

Това е един примерен сценарий, но той е твърде реалистичен и от него могат да се направят поне два извода, които ще формулираме във вид на следните въпроси.

– Може ли компанията да поддържа данните, описани в примера, само с един мощен сървър? Това е трудно. Ето защо е необходима мрежа от много хиляди компютри и съответен софтуер за паралелна работа в разпределена среда.

– Може ли компанията да си позволи да губи данни при ежедневен срив на компютър от мрежата? Загубата на данни е абсолютно недопустима и затова дублирането на файловете на няколко места в мрежата е наложително.

И накрая да си припомним закона на Паркинсон, адаптиран към данните, който гласи, че: „*Данните се увеличават, докато запълнят наличната дискова памет*“. Точно това се случва в края на 90-те години, когато компании и правителства в много страни закупуват в големи обеми поевтинялата дискова памет и данните са започнали да нарастват неограничено. Създала се е необходимост от нови технологии, които да се „преборят“ с огромното количество данни. Водещата идея е проста, но гениална – данните са обемни и не могат да бъдат едновременно на един компютър заедно с програмата, която ги обработва. Но програмата може да „отиде“ при данните, които са разположени на различни компютри.

Страници от историята: Началото на големите данни

Около началото на 2000-ата година редица компании и организации започват да развиват интензивна дейност, свързана с големите данни. На първо място, това е технологичната компания Google и софтуерната фондация Apache (ASF). По-долу следва кратка хронология на първите основни резултати, свързани с големите данни.

Година	Резултат
1989	Tim Berners-Lee изобретява световната мрежа World Wide Web (WWW).
1998	През април 1998 г. Lary Page и Sergey Brin публикуват статията „The Anatomy of a Large-Scale Hypertextual Web Search Engine“ (Brin, S. and L. Page, 1998).
	На 4 септември 1998 г. Lary Page и Sergey Brin основават компанията Google.
1999	На 25 март 1999 г. се основава и софтуерната фондация Apache (ASF).
2000	През януари 2000 г. Page и Brin публикуват алгоритъма Page Rank (Page, L. et al. (1998). Това е алгоритъм за оценка на уебстраници, който Google прилага при търсене на данни. Оценката се основава на важността на страниците, които цитират дадената страница. През януари 2000 г. Doug Cutting обявява първата версия на проекта Apache Lucene. Това е библиотека за пълно търсене и индексиране на текст.
2001	Doug Laney публикува статията за трите *V* на големите данни „3D *Data Management*: *Controlling Data Volume*, *Velocity and Variety*“ (Lancy, 2001).
2003	Сътрудници на Google публикуват статията „*The Google File System*“. GFS (Ghemawat et al, 2003) е мащабируема разпределена файлова система за големи данни.
2004	Google публикува статии относно системите за работа с големи данни: *BigTable* (Chang et al. 2008) – разпределена система за съхранение и управление на структурирани данни, проектирана да работи с PB обеми, съхранени на хиляди сървъри. *MapReduce* (Dean & Ghemawat, 2004) – среда за разпределени пресмятания с големи данни.
2006	На 1 април 2006 г. Doug Cutting и Michael Cafarel обявяват първата версия на Apache Hadoop – платформа за работа с големи данни (White, 2015).
2009	Посредством HDFS and Map Reduce Hadoop преодолява терабайтовата зона и сортира един TB файл, разположен на 1460 възела, само за 62 секунди.

Страници от историята: Двама докторанти и един гараж в California

Lary Page

Lary е компютърен учен и интернет предприемач. Роден е през 1973 г. в Lansing, Michigan, САЩ. Баща му е бил професор по компютърни науки в University of Michigan, а майката му – преподавател по програмиране. Когато е бил едва на 6 години, измежду играчките му е и микрокомпютър от първото поколение. Lary се интересува от технологии и бизнес. Самият той казва: „*От много ранна възраст разбрах, че искам да измислям неща. Вероятно на* 12*-годишна възраст знаех, че в крайна сметка ще създам компания*“. Lary има бакалавърска степен по компютърна науки от University of Michigan и магистърска от Stanford University. В същия университет продължава и с докторска програма[7)].

Sergey Brin

Sergey е компютърен учен и интернет предприемач. Роден е през 1973 г. в Москва, Русия. Когато е на 6 години, семейството му имигрира в САЩ. Получава бакалавърска степен по математика и компютърни науки от University of Maryland, College Park, където баща му е професор по математика. Продължава с докторска програма в Stanford University[13)].

Тандемът Lary – Sergey

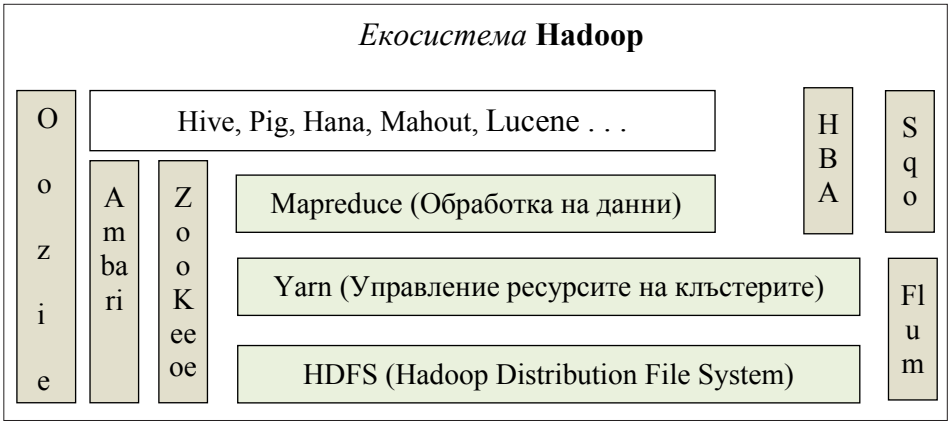
През 1995 г. в Stanford University двамата докторанти по компютърни науки Lary Page и Sergey Brin се срещат за първи път, без да осъзнават, че това е началото на един съюз и на блестяща кариера в технологичния бизнес.

L. Page започва да работи по проекта BackRub. Към него се присъединява и S. Brin. Това всъщност е началото на работата върху търсачката на Google. Следват години на упорита изследователска работа, през които написват две основни статии (Page, L. And S. Brin,1998) и (Brin, S. and L. Page, 1998). През 1996 г. на сървъра на Университета се намира първоначалната версия на Google и двамата студенти са вече пред прага да основат своя компания. Това става в гаража на тяхната приятелка, Susan Wojcicki, който те са наели под наем и използват за офис. На 4 септември 1998 г. Lary Page и Sergey Brin основават Google.

Интересно е да се отбележи, че в САЩ освен Google още шест други компании са стартирали своя бизнес от гаражи[5)]: Hewlett-Packard (1939), Mattel (1945), Microsoft (1975), Apple (1976), Maglite (1979) и Amazon (1994).

Екосистемата Hadoop

Apache Hadoop е среда с отворен код за работата с компютърни клъстери. Тя предоставя цялостна технология и инструменти, предназначени за съхранение и анализ на големи данни. Началото е поставено от Doug Cutting и Mike Cafarella през 2005 г.



Фигура 13. Част от компонентите на екосистема Hadoop

Идея за структурата на Hadoop е посочена на (фиг. 13). Hadoop е една от най-използваните платформи за разработване и изграждане на решения за големи данни. Например тя се използва от компании, управляващи социални мрежи (Facebook, Twitter, LinkedIn), компании за електронна търговия (Amazon, eBay, Alibaba), онлайн портали (Yahoo, AOL), финансови институти (Royal Bank of Scotland), транспортни компании и хотелски вериги (British Airways, Expedia) и много други.

Екосистемата Hadoop включва над 20 системи, обединени в няколко групи: базови, обработващи, NoSQL, аналитика, трансфер, сериализация, управление и наблюдение. Списъкът с разработените системи е дълъг (HDFS, MapReduce, Pig, Hbase, Mahout, Lucene, Sqoop, HCatalog, ZooKeeper,…), но имената им не подсказват тяхното предназначение (White, 2015). През следващите години е разработена цялостна система за работа с големи данни, известна днес като „*Екосистема Hadoop*“[16].

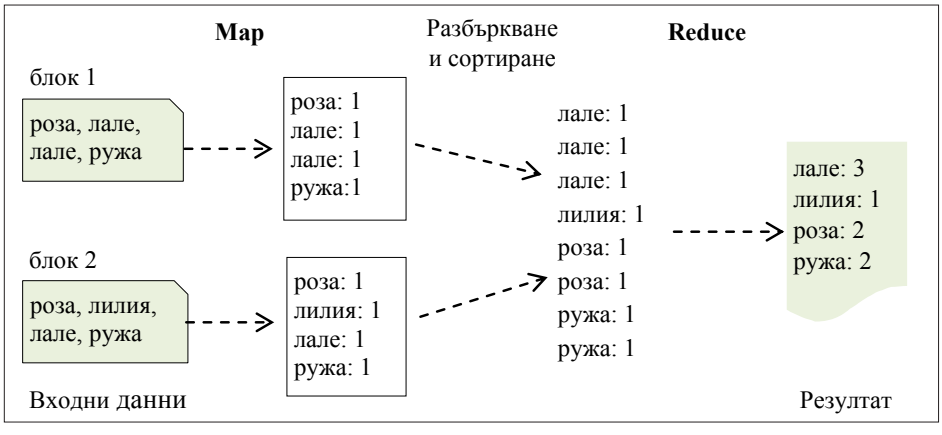
HDFS – разпределена файлова система на Hadoop

Doug Cutting и Michael Cafarel адаптират идеята на GFS и създават свои вариант, наречен по-късно *Hadoop Distributed File System* (HDFS).

Поради големия размер на файловете HDFS разделя данните на отделни блокове. Обикновено те са с размер от 128 MB и се съхраняват в отделни възли. За да се добие по-добра представа за обема на данните, който може да се съхрани и управлява от HDFS, можем да си мислим за 100 милиона файла, разположени върху 10 хиляди възела, с общ обем от много десетки PB. Системата поддържа политика на хоризонтално мащабиране и дублиране на данните. Управлението на отделните клъстери се извършва от главен възел, в който се съхраняват метаданните (данни, описващи основните данни: имена на файлове, директории, разположение на блокове и мн. др.) на цялата файлова система.

Map Reduce – среда за разпределени пресмятания на големи данни

Map Reduce е програмен модел за разпределени пресмятания, които се извършват върху данните, съхранени в блокове на клъстери от компютри. Продуктът е патентован от Google (Dean et al, 2004).



Фигура 14. Примерен вариант на Map Reduce за преброяване на цветята

Когато става дума за големи данни, възникват два основни проблема – съхранението и обработването им. За съхранението на големи данни в Hadoop се използва файловата система HDFS, а за обработката им – Hadoop Map Reduce.

Map Reduce е софтуерна среда за обработка на големи данни, разпределени върху много компютри. В общия случай всяка програма на Map Reduce изисква да се напишат две специални функции – *Map* и *Reduce*. Основната идея в цялата процедура е в изобразяването на данните в множество от двойки <*ключ*, *стойност*>, а след това се извършва редуциране на двойките с един и същ ключ. Един често използван прост пример за представяне на Map Reduce е свързан с текст, който представлява входни данни. Като резултат трябва да се пресметне колко пъти всяка дума се среща в текста. Тук също е използван този пример (фиг. 14).

Когато Map Reduce се стартира, входният текст ще бъде разбит на отделни блокове, които се прехвърлят на различни машини. След това всяка машина изпълнява Map функцията над съответния блок. Като резултат текстът се разбива на отделни думи и така се формират двойки от вида: <*w*, 1> , където *w* представлява отделна дума от текста. Единицата в случая означава, че думата *w* е намерена в текста един път. Но такива двойки в общия случай ще има много и те ще формират входните данни за втората фаза – разбъркването и сортирането на всички двойки. При третата фаза двойките се редуцират само до двойки с различни думи, а съответният им брояч ще бъде число, показващо колко пъти думата е намерена в текста.

NoSQL бази от данни

NoSQL се приема като съкращение на Not Only SQL (Не само SQL). На практика това е нов подход в моделите на БД, който е алтернатива на РМД. На NoSQL може да се гледа като на движение и като на технология. NoSQL *движението* стартира напълно независимо от големите компании с идеята да се търсят модели на данни, които допълват, изменят или изцяло заместват РМД. NoSQL *технологията* се изразява в проектирането и реализацията на БД от следващото поколение, към които се предявяват изисквания като: да бъдат нерелационни, разпределени и с отворен код, да са хоризонтално мащабируеми, да имат прост интерфейс и да позволяват огромен обем от данни и др.

Страници от историята: Първите NoSQL БД

Първата идея за NoSQL БД се заражда в началото на 1998 г., но тя все още е „размита“ и няма ясна концепция. През следващите няколко години идеята започва да си прокарва път чрез малки софтуерни групи, които стартират проекти за решаване на конкретни проблеми. Софтуерните проекти, които оказват най-голямо влияние на NoSQL системите за големи данни, са GFS и Bigtable (Chang, 2008). През 2007 г. Amazon публикува статия, в която представя своята база от данни Dynamo (Hastorun et al, 2007).

Тези смели първи проекти вдъхновяват други компании и те създават своите NoSQL БД. Така само за две години от 2007 до 2009 г. се създават NoSQL БД с отворен код: Riak, MongoDB, HBase, Accumulo, Hypertable, Redis, Cassandra и Neo4j (Fowler, 2015).

Обнадеждаващите резултати от първите реализации на NoSQL БД събира през 2009 г. водещите специалисти на симпозиум на тема „*Отворен код и нерелационни бази от данни*“. Доклади изнасят специалисти от софтуерните департаменти на добре познатите компании като Facebook, LinkedIn и Powerset. Те представят своите МБД и това съответно са системите Cassandra, Voldemort и Dynamite. Днес има вече редица БД, разработени „под флага“ на NoSQL движението.

Следва кратко представяне на основните четири NoSQL модела на данни.

Моделът на данни „Ключ – стойност“ (*Key – value*)

Данните в модела „Ключ – стойност“ са двойки от вида <*ключ – стойност*>, а структурата на данните представлява разпределена хеш таблица. За типа на ключовете и стойностите няма ограничение. За стойностите на всяка двойка могат да се използват разнообразни структури за съхранение, но обикновено това са JSON или BLOB (binary large object). Езикът на данните в системите, използващи този модел, се свежда до търсене, добавяне и изтриване, като позволяват да се обслужват едновременно заявките на милиони потребители. Например Amazon заявява, че Dynamo DB може да обработва повече от 10 трилиона заявки на ден и повече от 20 милиона заявки в секунда[1)].

Моделът на данни „Представяне по колони“ (*Wide Column Store***)**

Структурите от данни при този модел са сходни на таблиците от релационните системи. Концепциите за редове и колони са запазени, но вместо данните да се съхраняват по редове, те се съхраняват по колони. Този модел на данни позволява оптимизирането на операциите, характерни за цели колони. Концепцията в модела не е нова. Използвана е още през 1970 г. Представители на този клас NoSQL системи са Hbase (Apache), Cassandra (Facebook), Hypertable и др.

Графов модел на данни (*Graph Stores*)

Структурата от данни в графовия модел на данните е ориентиран граф. Всеки *връх* на графа има уникален идентификатор и се използва за съхранение на данни. *Дъгите* в графа не само показват отношенията между отделни върхове, но и ги допълват с още данни. БД изградени по този модел, отлично се справят с управлението на силно свързани данни и генерират отговорите на сложни заявки в рамките на милисекунди. Типични представители на този клас NoSQL системи са Neo4j и Amazon Neptune[2),12)].

Документен модел на данни (*Document*)

Документът е основно понятие в този модел на данни, което е близо до понятието обект в езиците за програмиране. Фокусът в този модел е в ПСД. В една документна NoSQL БД документите се съхраняват в JSON, XML или друг полуструктуриран формат и в общия случай са с различна структура. Една структура от по-високо ниво е т.нар. *колекция*. Тя се използва за групиране на множество от подобни по характеристиките си документи. Например един документ, в който се съхраняват данни за една книга, се представя като JSON обект. Данните за книгите от областта на БД в една библиотека биха могли да се групират в колекция „Бази от данни“. Операциите в документния модел са обикновено върху един документ и се свеждат до промяната на неговото съдържание, до пресмятания на изрази и/или търсене на определена стойност. MongoDB е една от водещите NoSQL БД[11)], изградена по този модел на данни.

На идейно ниво NoSQL моделите от данни са в противовес на РМД, но в някои проекти те се допълват и се използват съвместно. Това специално е отбелязано в (Mitreva & Kaloyanova, 2013).

> NoSQL базите данни споделят общи принципи, но решават различни потребности. Поради това често в практиката на NoSQL базата от данни се възлага изпълнението на една или няколко специфични задачи от проект, а не на целия проект и основната функционалност се реализира с RDBMS или друга NoSQL база от данни.

NoSQL СУБД намират редица приложения при работа с големи данни, но класическите SQL системи, обогатени с редица допълнения и разширения, продължават да бъдат също масово използвани.

7. 4. Трансформиране на данните в продукти

Заглавието на тази секция е повлияно от публикацията „*What is Data Science*? *The future belongs to the companies and people that turn data into products*“, която завършва с текста (Loukides, 2010):

> Способността да възприемаме данните – да ги разбираме, обработваме, да извличаме стойности от тях, да ги визуализираме и представяме на други – това ще бъде изключително важно умение през следващите десетилетия.

Това са думи на Hal Varian, който е главен икономист на Google и Emeritus Professor at the University of California, Berkeley (School of Information). Всъщност в този цитат се визира важността на основните дейности с данни, обединени в термина *аналитика на данните*.

Аналитика и анализ на данните

Бизнес компаниите, както и държавните институции и организации, документират своята текуща дейност с данни. След време тези данни стават ценен архив, в който е запаметен човешкият опит, натрупан през годините. Възможно ли е този „океан от данни“ да се трансформира в знания, които да бъдат използвани? Напредналите компютърни и информационни технологии вече са създали условията, при които „скритата мощ“ на данните може директно да се прилага при вземането на решения, при потвърждаването на научни хипотези и при формулирането на обобщени изводи, важни за цялото общество. Големите данни са днешното богатство на компаниите. Те са неговият „двигател“, който успешно ги трансформира в управленски решения.

Натрупването и съхраняването на огромно количество от данни, най-често имащи нестандартна, произволна структура, е само едната страна на проблемите на големите данни. Извличането само на подходящите данни измежду тях, техният анализ и интерпретация е втората, при това по-важната част от същността на големите данни.

Аналитика на данните (*data analytics*) е последователност от дейности, която обхваща целия жизнен цикъл на данните: събиране, идентифициране, филтриране, извличане на данни, проверка и почистване, обобщаване, анализ, визуализиране и използване на резултатите. Чрез методите на аналитиката на данните вземането на решения, базирани на интуиция и предишен опит от практиката, отпада. Решенията вече се основават на факти, извлечени от данните. Те успешно се прилагат не само в компаниите, които произвеждат стоки, но и в редица други области на живота, като сферата на услугите, търговията, научните изследвания, обществени организации, правителството и др. Съществена част от аналитика на данните е техният анализ, при който се използват разнообразни научни методи и техники. Най-общо *анализът на данни* (*data analysis*) е процедура, чиито акценти са върху извличане на полезна информация от данните, като се търсят отделни факти и шаблони, показващи или потвърждавайки определени тенденции (Thomas and al. 2016).

След серията от разнообразни гледни точки за големите данни завършваме с едно прагматичното мнение на института SAS[3)]:

> Важно е това, което организациите правят с данните. Големите данни трябва да бъдат анализирани за идеи и модели, които да доведат до по-добри решения и стратегии в бизнеса.

Естествено е, че сме съгласни с тази теза, защото данните наистина са големи, когато могат да се трансформират в знания, водещи към по-добри бизнес решения и стратегии. Големите данни са вече реалност и те стават все по-големи. Важно е всяка организация да разбере какво означават те за нея и което е по-важно – с какво големите данни могат да ѝ помогнат. Възможностите наистина са много.

Заключителни бележки

Темата за големите данни е обширна и по нея вероятно са написани много милиони страници, които вече също са част от големите данни. В тази статия акцентът е предимно върху основните исторически факти и събития, които са се случили до първите години от ерата на данните. Пътят, извървян от човечеството за последните 70 – 80 години, е белязан да бъде път на неговия технологичен прогрес. Всичко вече е променено – от личния до обществения живот на хората, от малкото селище до големите мегаполиси и държави. Хората не само използват резултатите от анализите на данните. Те самите в много случаи неосъзнато и/или неволно участват в генерирането на данни.

Технологичният прогрес в ерата на данните променя света, но създава и немалко сериозни проблеми. Тази тема, която не е представена в този текст, е също обширна и не е маловажна. Тук ще спомена само за две области, които обществото предстои да решава. На първо място, с използването на големите данни се засягат личното пространство на хората и начинът им на общуване. Личното общуване, характерно най-вече за подрастващото поколение, вече е предимно дистанционно и в момента голяма част от това поколение живее в един свой виртуален свят. Не по-малко важно е и публикуването на неверни и/или подвеждащи данни. Те се „размиват“ измежду останалите, трудно е да бъдат разпознати, а много често, при нарастващата глобализация на света, са с голяма потенциална мощ за влияние върху цели общества.

От проучванията на информационните агенции се вижда, че тенденцията на големите данни е да стават по-големи, но и допълнени с повече изкуствен интелект. Колко и как? Засега не е ясно, дори няма въведен термин за тях. Но докато това се случи и те станат новата реалност, можем да се „потопим“ по-надълбоко в света на днешните големи данни. *Науката за данните* предоставя тази възможност, а големият бизнес очаква нейните специалисти (Azalov, 2020).

БЕЛЕЖКИ

1. Amazon DynamoDB https://aws.amazon.com/dynamodb/
2. Amazon Neptune Fast, reliable graph database built for the cloud. https://aws.amazon.com/neptune/
3. Big Data. What it is and why it matters. https://www.sas.com/en_us/insights/big-data/what-is-big-data.html.
4. How Google Search Works https://support.google.com/webmasters/answer/70897?hl=en
5. Burgess, B. (2014). 7 Startups That Started in a Garage. https://pakwired.com/7-startups-that-started-in-garage/
6. Han, J, M. Kamber, J. PeiData (2012). Mining. Concepts and Techniques, Third Edition, *Morgan Kaufmann.*
7. Lary Page https://en.wikipedia.org/wiki/Larry_Page
8. Mashey, J. (2013) https://bits.blogs.nytimes.com/2013/02/01/the-origins-of-big-data-an-etymological-detective-story/
9. Mashey J. (1998). Big Data ... and the Next Wave of InfraStress. https://static.usenix.org/event/usenix99/invited_talks/mashey.pdf
10. Mashey J. https://en.wikipedia.org/wiki/John_Mashey
11. Mongodb – document-based, distributed database https://www.mongodb.com/
12. Neo4j – The Leader in Graph Databasesn https://neo4j.com/
13. Sergey Brin https://en.wikipedia.org/wiki/Sergey_Brin
14. The A.M. Turing Award, sometimes referred to as the „Nobel Prize of Computing“, was named in honor of Alan Mathison Turing (1912 – 1954), a British mathematician and computer scientist. He made fundamental advances in computer architecture, algorithms, formalization of computing, and artificial intelligence. https://amturing.acm.org/
15. Walker-Ford, M. (2019). This is What Happens in an Internet Minute in 2019 [Infographic] https://www.socialmediatoday.com/news/this-is-what-happens-in-an-internet-minute-in-2019-infographic/551391/
16. Екосистема Hadoop. Съществуват разнообразни *екосистеми на данни* (ЕСД). Hadoop е ЕСД за големите данни. Всяка екосистема е среда от технологии, която се развива във времето и включва: *инфраструктура*, *анализи* и *приложения*. Инфраструктурата предоставя хардуерните и софтуерните услуги от най-ниско ниво. Аналитичните платформи се използват за по-задълбочен анализ с цел разкриване на нова информация, визуализиране на данни и други. Приложенията включват разнообразни области като здравеопазване, търговия на дребно, финансови институции и много други.
17. Източник на снимките в текста: интернет.
18. Всички интернет страници, цитирани в статията, са посетени и отворени за последен път на 20.02.2020 г.

ЛИТЕРАТУРА

Азълов, П. (1981). *Бази от данни. Релационен и обектен подход*. Техника.

Азълов, П. (2020). Данните и пътят към Науката за данни. *Математика и информатика* (Под печат).

REFERENCES

Abiteboul, S., P. Buneman & D. Suciu (1999). *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann.

Azalov, P. (1981). *Databases. Relational and Object Approaches*. Tehnika.

Azalov, P. (2020). Data and the Road towards the Data Sciene. *Mathematics and Informatics* (To appear).

Codd, E. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*. 13 (6), pp. 377 – 387.

Balcilhon, F. & S. Khoshafian (1989). A Calculus for Complex Objects. *Journal of Computer and System Sciences*. no.38, pp. 326 – 340.

Brin, S. & L. Page (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems.* 30, pp. 107 – 117. https://snap.stanford.edu/class/cs224w-readings/Brin98Anatomy.pdf

Cattell, R. & D. Barry (Ed) (2000). *The Object Data Standard: ODMG 3.0*. Morgan Kaufmann Publishers.

Chang, F., et al. (2008). Bigtable: A Distributed Storage System for Structured Data. *ACM Transactions on Computer Systems* (TOCS), no. 4. https://doi.org/10.1145/1365815.1365816

Connolly, T. & C. Begg. (2015). *Database Systems. A Practical Approach to Design, Implementation, and Management.* Pearson.

Dean, J. & S. Ghemawat (2004). MapReduce: Simplified Data Processing on Large Clusters. *OSDI'04*: *Sixth Symposium on Operating System Design and Implementation*, pp. 137 – 150

Fowler, A. (2015). *NoSQL For Dummies*. John Wiley & Sons.

Ghemawat, S,. H. Gobioff & S. Leung. (2003). The Google File System. *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pp. 20 – 43.

Hastorun, D. et al. (2007*). Dynamo: Amazon's Highly Available Key-value Store Giuseppe DeCandia*. SOSP'07: ACM 978-1-59593-591-5.

Lancy, D. (2001). 3D Data Management: Controlling Volume, Velocity, and Variety. Applicvation Delivery Strategies. *Meta Group*, https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf

Loukides, M. (2010). What is Data Science? The future belongs to the companies and people that turn data into product. *O'Reilly Radar*.

https://cdn.oreillystatic.com/en/assets/1/event/55/strata2011_what-is-data-science_pdf.pdf.
Mitreva, E. & K. Kaloyanova (2013). *NoSQL Solutions to Handle Big Data* https://www.academia.edu/29583159/NoSQL_Solutions_to_Handle_Big_Data
Page, L, et al. (1998). *The PageRank Citation Ranking: Bringing Order to the Web.* http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf
Thomas, E., W. Khattak & P. Buhler (2016). *Big Data Fundamentals. Concepts, Drivers &Techniques*. Prentice Hall.
Tom, M. (2017). *JSON at Work: Practical Data Integration for the Web*. O'Reilly Media, Inc.
Watson, H. (2014). Big Data Analytics: Concepts, Technologies, and Applications. *Communications of the Association for Information Systems*: vol. 34, article 65. DOI: 10.17705/1CAIS.03462
White T.(2015). *Hadoop: The Definitive Guide*. O'Reilly Media.

ABOUT THE DATA AND THE ROAD TOWARDS THE BIG DATA

Abstract. Over the past two decades, data turned into *big data*. The term *big data* encompasses the exponential growth of all data but most notably, the semi-structured and unstructured types. The speed with which data is generated and amassed necessitates the introduction of new technologies for their collection, transfer, storage and processing.

In a chronological fashion, this article begins with the foundational work of database development and progresses through the emergence of big data. Concurrently, the scientists, whose theoretical and practical work underpinned the development of data processing systems, are also discussed. The article provides examples to show the pervasiveness of *big data* in our daily lives. The key technologies utilized to process big data are described, including the *Hadoop ecosystem* and the NoSQL type of databases. A strong emphasis is given to the role of data analytics, used to transform the raw data into information, decision-making models and knowledge.

Keywords: big data; history of databases; NoSQL databases; tructureddata; unstructured data; semistructured data; ecosystem Hadoop

✉ **Dr. Pavel Azalov**
ORCID iD: 0000-0003-4313-3383
Professor Emeritus of Computer Science
Penn State, Hazleton, U.S.A.
E-mail: pka10@psu.edu