РЕЛАЦИОННИ И НЕРЕЛАЦИОННИ БАЗИ ОТ ДАННИ

Даниела Орозова
Бургаски свободен университет

RELATIONAL AND NON-RELATIONAL MODEL OF DATABASES

Daniela Orozova
Burgas Free University

Abstract: *Разглеждат се основните понятия и характеристики на релационните бази от данни като набор от обекти (таблици, ограничения, индекси, изгледи, съхранени процедури, дефинирани от потребителя функции, тригери и други), които представляват логически компоненти, видими за потребителите. Представени са алтернативни подходи и особености на нерелационните бази от данни, примерни справки и съответните заявки. Засяга се парадигмата Big Data с нейни основни характеристики и предизвикателства.*

Key words: *Databases, NoSQL, Big Data, Education Space, Data Science.*

Днес почти всяко бизнес приложение поддържа своя база от данни. Тяхното изучаване е важно не само за информатици, именно защото знанията за базите от данни станаха неразделна част от компютърната грамотност в съвременния свят.

1. Релационен модел на данните

Голяма част от съвременните системи за управление на бази от данни (СУБД) съхраняват и обработват информацията, използвайки релационен модел за управление на бази от данни, доказал своите предимства. Неслучайно списание Forbes през 2002 година посочи релационния модел на данни, разработен от Едгар Код, за една от най-важните иновации от последните 85 години.

В системите за управление на релационни бази от данни (Relational Database Management Systems – RDBMS) системата управлява всички данни в таблици (релации). Теорията на релационното проектиране се състои от следните основни понятия: релации (представяни чрез таблици), уникалност, външни ключове и домейни, релационни връзки, нормализация на данните, правила за запазване на целостността на данните.

В основата на релационния модел стои математическото понятие n-членна релация. Всяка релация е множество от елементи, които се състоят от n-компонента и се наричат n-токи.

Дефиниция 1: Нека D1, D2,…Dn са n≥1, именувани множества от допустими стойности на съответни величини, (не непременно различни). Релация (отношение) се нарича произволно подмножество на декартовото произведение на тези множества:

$D_1 \times D_2 \times \ldots \times Dn = \{<d_1, d_2, \ldots, dn> | \; di \; Di, i=1,2,\ldots n\}$

Чрез една релация се моделира даден клас от обекти, а всяка n-торка от релацията представя конкретен обект от този клас. Съществено предимство на релационния модел пред другите модели е еднотипното представяне на класовете от обекти и връзките между тях чрез релации. Така релационният модел представя данните в базата като множество от релации или отношения.

Като се използва самият начин на дефиниране на понятието релация тя може да се представи чрез структурата таблица. Таблицата е структура за съхраняване на данни, изградена от редове, които тук са n-торките: <d_1, d_2, …,dn>| di ϵ Di, i=1,2,…n и стълбове, които съдържат елементи само от една и съща област. Тук се изисква спазване на наредбата на стълбовете в таблицата. Всеки именуван стълб при това представяне на релацията се нарича атрибут, а всеки ред – запис или кортеж (typle). На всяко име на атрибут Aij отговаря множество D(Aij), (област от допустими стойности) на атрибута Aij, което съдържа неделими (атомарни, неразложими) стойности. Техните елементи образуват най-малката значима единица от данни. Множеството от стойности, към което принадлежи даден атрибут се нарича домейн.

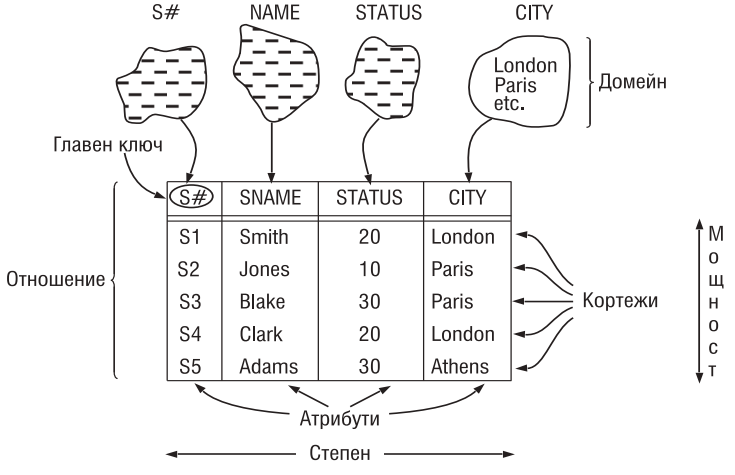
Дефиниция 2: Домейн е произволно множество от неделими (атомарни, неразложими) стойности. Домейнът може да се представи като произволен тип на данните. Математическото множеството, от което е съставен е непразно, крайно и елементите, над които е дефинирано са произволни.

Дефиниция 3: Запис t се нарича наредена n-торка от стойности <d1, d2, …,dn>.

Дефиниция 4: Броят m на записите се нарича мощност на релацията r.

Дефиниция 5: Борят на атрибутите n се нарича степен на релацията r.

Така всеки ред в таблицата представлява съвкупност от свързани помежду си данни, които описват свойствата на някакъв обект или връзка. За да могат отделните стойности да се различават по-лесно, колоните както и самата таблица са именувани. Така таблицата може да се приеме като множество от редове (записи) и колони. Редът е съвкупност от факти относно обект и в терминологията на релационните бази от данни се нарича запис или кортеж. Колоната в таблица се нарича атрибут и представя някое от свойствата към обектите. Броят на всички атрибути представлява степента, а броя на кортежите мощността на таблицата, както е показано на фиг.1 [9].



Фиг. 1. Основни понятия в релационния модел

Понеже информационното съдържание на релацията не зависи от наредбата на атрибутите й, естествено е да се търси алтернативен начин на представяне на релацията, различен от таблица. Използвайки, че стълбовете са именувани, спазването на наредбата им не е необходимо. Така всяка релация може да бъде представена като

множество от функции, дефинирани върху атрибутите, като всяка функция определя някаква n-торка (елемент) на релацията. Например, ако R е релация, с атрибути A={Име, Оценка}, тя може да се определи с функции: f_1, f_2, …, от вида:

f: A → Di, за i=1, 2.

Не е трудно да се установи, че всъщност двата начина на представяне са еквивалентни и това ни позволява да използваме, който и да е от тях.

1.1. Релационна схема

Релациите са динамични, променливи във времето структури от данни. Във всеки момент те представят текущото състояние на класа от обекти. Съществено е, че през цялото време на съществуването си, релациите запазват своята структура. Структурата на всяка релация се нарича още тип на релацията и се задава с нейната релационна схема.

Дефиниция 6: Релационна схема на релация R с атрибути A_1, A_2, …An се задава чрез името на релацията и нареден списък от съответните й атрибути R (A_1, A_2, …An). Всеки един от атрибутите може да получава стойност само в определен домейн, затова може да се използва записът: R (A_1:D_1, A_2:D_2 …An : Dn).

Нека R (A_1, A_2, …An) е релационна схема, а r е релация над тази релационна схема. Този факт се означава със записа r : R или r : R(A_1, A_2, …An).

Дефиниция 7: Нека R_1 , R_2 ,..., Rn са релационни схеми, а IC е множество от ограничения за цялостност на данните. Тогава множеството {R_1, R_2 ,..., Rn} от релационни схеми, заедно със множеството от ограничения за цялостност IC се наричат *схема на релационната база от данни*.

Нека r_i за i=1, 2,...,n са релации (отношения) дефинирани върху схемата на релационната база от данни S, които удовлетворяват ограниченията за цялостност зададени в IC. Тогава множеството S = { r_1 , r_2 ,...r_n} от релации се нарича състояние на релационната база от данни със схема S. Понеже на всеки атрибут Ai, съответства домен Di за i = 1,2,…n, този факт може да се отбележи по следния начин: Di=dom (Ai), i = 1,2,…n.

1.2. Ключове

Тъй като в релационния модел не се допуска наличието на два абсолютно еднакви записа, налага се въвеждането на логическите понятия ключ и суперключ.

Дефиниция 8: Суперключ SK на релационната схема R (A_1, A_2, …An) се нарича произволно подмножество {B_1, B_2, …Bm} на множеството от атрибути {A_1, A_2, …An}, такова, че за всяка двойка различни записи ti ≠ tj от отношението r : R (A_1, A_2, …An), стойностите на атрибутите от множеството SK са различни.

Дефиниция 9: Минималният суперключ на релационната схема R (A_1, A_2, …An) се нарича ключ.

Една релационна схема може да има няколко възможни ключа, но един от тях се избира за ключ на релацията и се нарича първичен (главен) ключ на релацията, а останалите са вторични (възможни, алтернативни) ключове. Така първичният ключ се състои от един или повече атрибути, които еднозначно идентифицират записите. Този ключ трябва да бъде уникален и не може да съдържа null стойности. Една релация има само един първичен ключ.

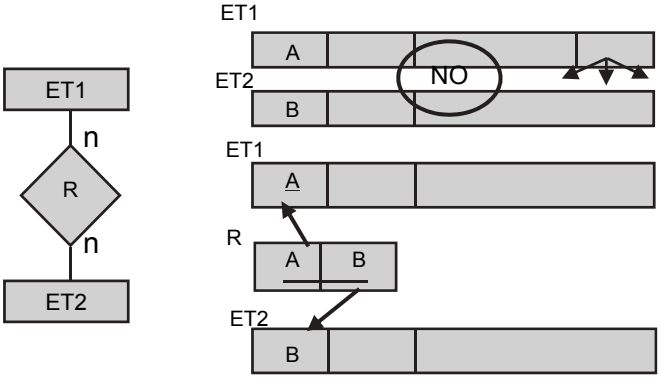
Дефиниция 10: Външен ключ K* на релационната схема R (A_1, A_2, …An) се нарича произволно подмножество {B_1, B_2, …Bm} на множеството от атрибути {A_1,

A_2, …An}, който не е първичен ключ за тази релационна схема, но съществува друга релационна схема, за която K* е първичен ключ.

Първичните (главните) и външните ключове са основно средство за установяване на връзките между елементите на релациите. Първичният и външният ключ могат да се състоят от единствен атрибут (прост ключ) или от повече от един атрибути – нарича се съставен ключ.

Релационна връзка (relationship) се нарича връзка между таблици, която е базирана на първичен ключ от едната таблица и външен ключ от другата таблица. Съществуват три типа релационни връзки между таблиците: релационни връзки „едно към едно"; релационни връзки „едно към много"; релационни връзки „много към много".

- Релационни връзки „едно към едно". Две таблици A и B са свързани с релационна връзка „едно към едно", ако за всеки ред от таблицата A има най-много един съответстващ ред от таблицата B, но всеки ред от таблицата B може да има точно един съответстващ ред от таблица A. Релационна връзка „едно към едно" се създава, ако и двете свързани колони са първични ключове или имат ограничения за уникалност. Този тип релационна връзка е най-рядко срещан, тъй като информация, свързана по този начин, обикновено се намира в една таблица. От съображения за сигурност или удобство при обновяване на данните информацията може да се раздели в две таблици.
- Релационни връзки „едно към много". Две таблици A и B са свързани с релационна връзка „едно към много", ако за всеки ред от таблицата A има нула или повече съответстващи редове от таблицата B, но всеки ред от таблицата B може да има точно един съответстващ ред от таблицата A. Този тип релационна връзка е най-често срещан и може да се създаде, ако само една от свързаните колони е първичен ключ или има ограничение за уникалност.
- Релационни връзки „много към много". Две таблици A и B са свързани с релационна връзка „много към много", ако за всеки ред от таблица A има много съответстващи редове от таблицата B и обратно. Този тип релационна връзка се създава, като се дефинира трета таблица, наречена свързваща таблица (junction table), която съдържа първичните ключове от двете таблици като външни ключове; първичният ключ в свързващата таблица е съставен от двата външни ключа. Следователно една релационна връзка „много към много" се представя чрез две релационни връзки „едно към много" (фиг.2).



Фиг. 2. Представяне на връзка от тип M:N в релационен модел

1.3. Ограничения за цялостност на данните

Ограниченията за цялостност на данните са логически условия, отнасящи се до стойностите на различните атрибути, дефинирани в схемата на базата от данни. Съществуват няколко вида ограничения върху данните, които могат да се дефинират върху релационната схема и са валидни за всяка релация над тази релационна схема.

- Ограничение за цялостност на обекта. Могат да се дефинират ограничения върху атрибутите, които не трябва да приемат *null* стойности. Такива стойности не се допускат за главните ключове, т.к. ако два различни записа могат да имат *null* стойности за главния ключ, то те могат да се укажат неразличими.
- Ограниченията върху домейните – налагат неделимост на техните стойности.
- Ограниченията за цялост на връзките представляват система от правила, която осигурява връзките между записите в свързаните релации да са винаги валидни и да не се позволява случайно изтриване или промяна на свързани данни. Разделяме ги на структурни и поведенчески логически ограничения. Референтен интегритет е процесът, чрез който връзките се поддържат валидни в данните. *Референтният интегритет* предотвратява съществуването на запис от подчинена таблица (наследен запис) без съответен запис от главната таблица (родителски запис). Освен тези ограничения на всяко приложение са присъщи и собствени ограничения, произтичащи от семантиката на атрибутите.

1.4. Нормализация на данните

Оптимизацията на проекта на базата от данни включва процеса на нормализация. Решенията, взети за структурата на таблиците, могат да повлияят по-късно върху производителността и върху написването на програмите по време на процеса на разработване. Нормализацията на данните осигурява организиране на данните по такъв начин, че:

- освобождава от излишество като осигурява данните да се въвеждат и съхраняват само веднъж;
- осигурява непротиворечивост на данните – данните в базата да не съдържат противоречива информация, което води до аномалии.

Целта на нормализацията е да се декомпозират таблиците в по-малки таблици, дефинирани така, че да се избегне излишно дублиране на данни. След разделянето на данните в отделни по-малки и добре структурирани таблици, се дефинират релационни връзки между тях. За да се получи нормализация на данните, първо се достига до първа нормална форма (First Normal Form – 1NF), след това до втора нормална форма (Second Normal Form – 2NF) и накрая до трета нормална форма (Third Normal Form – 3NF). Без да навлизаме в теорията на релационната алгебра и функцуоналните зависимости при релационния модел, можем кратко и неформално да определим смисъла на тези три нормални форми, които са практически важни при проектиране на една релационна база. По същество нормалните форми определят правила, които отделните таблици трябва да притежават.

Първа нормална форма. Една таблица е в първа нормална форма, ако всяка колона има точно една стойност за всеки ред, а не списък или множество от стойности. Първата нормална форма изисква всички стойности на данните да бъдат атомарни (принцип за неделимост).

Втора нормална форма Една таблица е във втора нормална форма, ако тя е в първа нормална форма и всяка колона, която не е част от ключа, зависи от първичния ключ, (който може да е съставен) и не зависи от някое подмножество на първичния ключ.

Трета нормална форма. Една таблица е в трета нормална форма, ако тя е във втора нормална форма и всяка колона, която не е част от ключа, не зависи от нищо друго, а само от ключа. Така се елиминират зависимости между колоните, които не са част от първичния ключ.

Допълнителните нормални форми Boyce/Codd и четвърта нормална форма се прилагат в специални и рядко срещани случаи.

Основните предимства от нормализацията на данните са:

- данните заемат по-малко място за съхраняване;
- по-бързо се изпълняват актуализациите на данните;
- по-лесно се въвеждат нови данни;
- намалява се наличието на несъгласуваност и противоречивост на данните;
- ясно се дефинират релационните връзки между данните в таблиците.

При работа със системите за управление на бази от данни отделните потребители работят с базата по едно и също време. Ако действията им са неконтролируеми, една операция за достъп може да влияе върху останалите, като в резултат базата от данни може да стане противоречива. Системите реализират методи за регулиране на едновременния достъп и възстановяването на базата от данни при неуспешно приключване на транзакция.

Транзакцията е логически неделима единица за обработка на базата от данни, която включва една или повече операции за достъп до нея и променя базата от данни от едно състояние в друго. Традиционните релационни бази от данни са ориентирани към работа с транзакции, които удовлетворяват така наречените ACID правила:

- Atomicity (атомарност) – една транзакция или се изпълнява изцяло, или не се изпълнява въобще и базата от данни остава непроменена.
- Consistency (консистентност) – транзакцията променя базата от данни от едно непротиворечиво състояние в друго непротиворечиво състояние.
- Isolation (изолация) – не може да се достъпва информация, която се обработва от незавършила транзакция.
- Durability (дълготрайност) – след като една транзакция приключи успешно, направените от нея промени стават постоянни, даже и при последващ срив в системата.

Счита се, че по критерия за трудност на програмиране, а следователно и на използване, най-добър е релационният модел на данните. Той оперира само с едно понятие – релация (отношение) за представяне както на данните, така и на връзките между тях. Мрежовият модел е по-сложен за използване, защото работи с типове записи, връзки и техните взаимоотношения. В този модел особено трудно се реализира връзка от типа „много към много”. Аналогична е ситуацията и с йерархичния модел, който работи с указатели (типове виртуални записи).

Релационните системи за управление на бази от данни са разработени за приложения, при които след проектирането структурата на данните се променя рядко и веднъж уточнена, схемата остава постоянна във времето. Освен това данните са хомогенни по тип.

Съществуват приложения, при които данните представляват нехомогенни множества от обекти, изисква се управление на сложни, вложени и взаимосвързани обекти като: пространствени данни (карти, графи); данни за инженерен дизайн (архитектура на сграда, интегрални схеми); хипертекстови и мултимедийни данни (текст, изображения, аудио, видео данни); времеви редове; world-wide-web информация и др. Ограничения на релационния модел го правят трудно използваем при редица приложения, управляващи процеси. Основни недостатъци на релационния модел са:

липса на възможности за дефиниране на собствени типове данни, не се поддържат различни видове абстракции, ограничени операции за обработка на данните, съхраняваните данни са хомогенни (вертикално и хоризонтално), не могат да се формулират рекурсивни заявки, не се работи с параметъра „време“, проблем с езиковото несъответствие при използване от език за програмиране.

Релационният модел на данните, въпреки че притежава добра теоретична основа и редица важни преимущества като: простота при работата, пригодността му за интерактивна обработка на транзакции, осигуряване на независимост на данните, процесът на нормализацията, който намалява излишеството в съхраняваните данни, този модел има редица алтернативи, които са подходящи при изграждане на специализирани приложения.

2. Нерелационни бази от данни

Бази от данни, наричани **NoSQL** – Not only Structured Query Language**,** възникват поради необходимостта от обработка на големи масиви от данни, които се разширяват хоризонтално. Тези хранилища предоставят механизъм за съхранение и възстановяване на данни със свободен модел. Важна цел е оптимизиране на производителността при обработка на големи обеми от данни, затова обикновено те имат разпределена архитектура – данните се разпределят и съхраняват на няколко сървъра.

В този случай важи теоремата CAP, известна още като теорема на Брюър, която гласи, че в една разпределена система трябва да се изберат две от следните правила:

- Консистентност (Consistency - C) – всички клиенти на базата от данни виждат една и съща информация, даже при конкурентно обновяване.
- Наличност (Availability - A) – всички клиенти на базата от данни могат да достъпват данните и да извличат информация.
- Възможност за разделяне (Partition tolerance - A) – базата от данни може да се разделя и локализира върху множество сървъри.

Едновременното осигуряване на трите не е възможно.

В повечето случаи системите за работа с нерелационни бази от данни избират наличност и възможност за разделяне; гарантирането на консистентност при разпределена база от данни изисква кворум между отделните сървъри, което води до забавяне във времето.

Така при нерелационни бази от данни се спазват следните BASE правила:

- Basically Available (базова наличност): системата не гарантира наличието на данните (както е в CAP теоремата). Всяка заявка гарантирано получава отговор. Но отговора може да бъде „неуспех“ за получаване на исканите данни или данните могат да бъдат непълни или в процес на промяна.

- Soft state – състоянието на системата може да се променя с течение на времето (дори и по време, в което няма обработване на данни, може да има промени в състоянието на данните).

- Eventual consistency (евентуална консистентност) – системата евентуално ще стане непротиворечива, след като спре да получава данни (т.е. ако няма обновяване, всички възли на базата от данни ще връщат актуални резултати).

Различните системи за работа с нерелационни бази от данни използват различни технологии за съхранение на данните. Следва кратък преглед на най-често използваните технологии.

2.1. Технология за съхранение на данни: *Key-Value*

Това е бързо, мащабируемо и високо достъпно хранилище за произволен достъп (четене/запис) до всякакви данни, асоциирани с ключ.

При този подход базата от данни е съвкупност от наредени двойки *<ключ, стойност>*. В програмните езици тази структура е позната като: associative array, map, dictionary и др.

Така с ключовете са свързани стойности от различни типове: низ, число, дата, списък от низове, масив и т.н. Използват се основно за:

- търсене на стойност по ключ или на ключ по дадена стойност;
- съхранение на записи като добавяме ключ и стойност;
- промяна на съхранявана стойност, когато ключа съществува;
- изприване на записи (на ключ и стойност);
- услуги за архивиране, разпространение на съдържаниеи др.

Популярни продукти от този тип са системите:

Cassandra – система за управление на данни с отворен код, първоначално разработена 2008 г. от Facebook. За заявките на Cassandra се използва езика CQL (Cassandra Query Language), който много прилича на SQL.

BerkelryDB е създадена 1991 г. като част от BSD (Berkeley Software Distribution) операционната система с отворен код, разработена в университета в Калифорния. Скоро след това Oracle добавя SQL интерфейс.

2.2. Документни хранилища (Document stores)

Документното хралилище е хранилище от тип ключ-стойност, където стойността е документ, представен отново във вида ключ-стойност и служи за работа с документи в структуриран или неструхтуриран формат. Те се разделят на два типа, като програмния код е оптимизиран за работа с определени външни формати.

- Хранилища за съхранение и анализиране на неструктурирани данни (новини, документи, ...) в формат (PDF, MS Word, Excel и др.) Неструктурираните данни са данни, които се различават по състав и организация, както и по съдържание – например текстови документи с общ файлов формат, но различно съдържание (доклади, анализи, заповеди и т.н.). Организацията на неструктурираните данни допуска динамични и често непредсказуеми промени. Например една преписка може да се превърне от прост линеен списък от документи „въпрос/отговор“ в сложна структура, поддържаща данни от технически, юридически и друг характер. Друга важна характеристика е високото ниво на свързаност между тях, например документ - едни преписка може да е свързана с непредвидим брой други документи, събития или лица.
- Хранилища за съхранение и анализиране на структурирани данни (JSON, XML, YAML и други формати).

Популярен продукт от този тип е софтуерът *MongoDB* - NoSQL система за работа с бази от данни, която съхранява структурираната информация в JSON формат.

JSON или JavaScript Object Notation е текстово базиран отворен стандарт за обмен на данни. Данните се описват като свързани двойки „ключ-стойност“:

```
{
        поле1: стойност1,
        поле2: стойност2,
        ...
        полеN: стойностN
}
```

Създаден първоначално за JavaScript, в последствие JSON се използва като алтернатива на XML, за пренос на данни между приложения, написани на различни езици. Следва пример, относно съхранение на данни в JSON формат:

```
{
        "firstName": "John",
                "lastName": "Smith",
                "isAlive": true,
                "age": 25,
                "height_cm": 167.6 ,
                "address": {
                        "streetAddress": "21 2nd Street",
                        "city": "New York",
                        "state": "NY",
                        "postalCode": "10021-3100“
                },
                "phoneNumbers": [// масив
                        {"type": "home","number": "212 555-1234"},
                        {"type": "office","number": "646 555-4567" }
                        ],
                "children": [],
                "spouse": null
}
```

2.2.1. Основни понятия в MongoDB

• База данни – съвкупност от колекции;

• Колекция – съвкупност от документи (съответства на понятието таблица в релационните бази от данни);

• Документ – съвкупност от записи (съответства на понятието ред/запис в релационните бази от данни, но отделните документи не е задължително да имат еднаква структура на данните);

• Запис – наредена двойка „ключ-стойност“ или „поле-стойност“.

Структурните елементи „база данни“, „колекция“, „документ“ не се създават с команди, а те се създават просто с добавяне на запис.

Какво е характерно за работа с MongoDB?

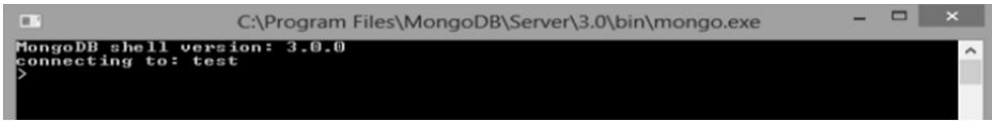
За стартиране на сървъра, трябва да извикате mongod.exe, разположен на съответното място на диска:

C:\Program Files\MongoDB\Server\3.0\bin\mongod.exe

След това стартирате shell чрез обръщение към mongo.exe:

C:\Program Files\MongoDB\Server\3.0\bin\mongo.exe

При това действие се извършва свързване с временна тестова база от данни, наречена “test”, както е показано на фигура 3.



Фиг. 3. Стартиране на MongoDB

Достъп до сървъра може да се прави и чрез приложения написани на различни езици за програмиране. За целта трябва да има съответен драйвер на езика за работа с MongoDB. Драйвери: http://docs.mongodb.org/manual/applications/drivers/

Примерни команди в MongoDB, достъпни чрез обекта db са:

- db.help() – метод показващ помощна информация за достъпните операции и методи за работа с базата данни;
- db.<колекция>.help() – помощна информация за операциите достъпни за колекциите, където <колекция> е име на колекция (съществуваща или несъществуваща);
- db.<колекция>.find().help() – помощна информация за модификаторите и операциите достъпни за курсор. Метода *find*, връща курсор с намерени данни от дадена колекция. Курсорът представлява извлечени чрез заявка документи от базата от данни, които могат да бъдат обработвани допълнително (в RAM паметта).

Следва пример за добавяне на потребители в колекцията *students* в базата, наречена BSU:

- use BSU – избор на база от данни;
- db.users.insert({"name":"Иван", age:20}) – добавяне на документ в колекцията students;
- db.users.insert({"name":"Мария", age:19});
- db.users.find() – търсене и извеждане на всички документи от колекцията users;

Резултат:

```
{ "_id" : ObjectId("5513d0e47f0cb347bf2a4708"), "name" : "Иван", "age" : 20}
{ "_id" : ObjectId("5513d0f47f0cb347bf2a4709"), "name" : "Мария", "age" : 19 }
```

Всеки документ има уникален идентификатор с *име _id*, който се записва като първи елемент. Ако не сме указали идентификатор, то такъв се създава автоматично.

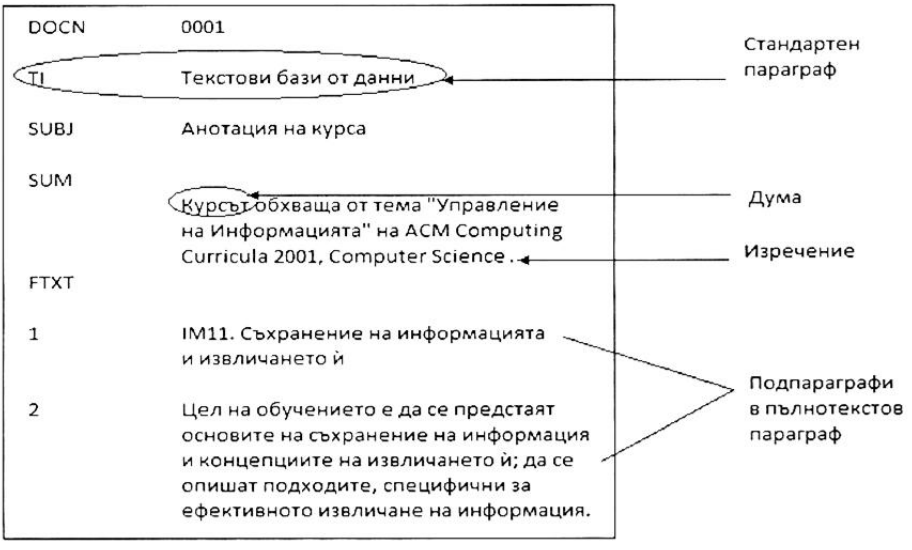
Пълно описание на опциите на MongoBD за работа с вложени документи, елементи на масиви и др., можете да намерите в [14].

2.2.2. Пълнотекстова извличаща система (Full Text Retrieval System)

Пълнотекстовите извличащи системи са документни хранилища, които предоставят възможности за свободно търсене по дума или поредица от думи в голям набор от документи; промяна на отделни полета в документ или на цял документ. Освен съдържанието на документите обикновено се съхраняват и метаданни – описание на документа, връзки с предишни версии и т.н.

- *Документът* е основна единица в документното хранилище, еквивалент на запис в релационната база от данни.

- *Параграфът* е отделна част на документа, с цел по-добро структуриране на информацията, еквивалент на поле при релационните бази от данни. Параграфите се проектират. Например даден параграф може да съдържа името на производителя, друг – дата на производство, трети името на автора на документа и т.н.



Фиг. 2. Елементи на документна бааза от данни

- Изречението е част от параграф и се разглежда като низ от символи, включващ разделителите на думи като: интервал, скоби и др., завършващ със знак за край на изречението. Търсенето може да се извършва на ниво изречение.

- Дума. Изреченията се разделят на думи. Думата е символ или низ от символи, оградена от не индексируеми символи (интервал, знак за пунктуация, нов ред, край на изречение). Може да се търси по всяка дума с изключение на стоп-думите (предлози или съюзи, които се срещат много често).

2.2.3. Файлова организация на пълнотекстова извличаща система

При изпълнение на заявки, обикновено системата за търсене не извършва търсенето директно върху текстовете от документната база от данни. Търсенето се извършва чрез група индекси – инвертирана файлова система.

Текстов файл съхранява текстовете на документите, участващи в базата от данни. В текстовия индексен файл има запис за всеки документ в базата. Записът за документа съдържа указател към документа в текстовия файл, параграфите му и правата за достъп до документа. Текстовите и инвертираните файлове могат да са повече от един при големи текстови бази от данни и физически отделните файловете могат да бъдат разположени върху различни устройства.

Така, когато се зареждат документите в базата от данни, всеки един от тях се индексира в съответствие със зададените спецификации – такива като максимален размер на документите, тип и максимален размер на параграфите и др.

Генерират се два основни файла – речников файл и инвертиран файл.

- В речниковия файл се съхраняват всички отделни думи, намиращи се в заредените документи на текстовата бази от данни (броя появи на думата и броя документи, съдържащи думата). С помощта на речниковият файл бързо се търси по отделна дума и по маска на дума. За всяка дума речниковият файл съхранява указател към инвертирания файл.

- Инвертирания файл съхранява точна информация за всяка дума, по която може да се търси в базата от данни. Това е запис за всяка дума, съдържащ номер на документ, номер на параграф, номер на изречение и номер на дума в изречението. Тази информация се използва при позиционното търсене, например търсенето в параграф или при използване на операторите AND, SAME (думите да са в един параграф), NEAR (думите да са в едно изречение) и т.н.

- Обратния речник е речников файл, който съхранява думите, написани в обратен ред и подобрява бързодействието при някои видове търсене, като търсене по префикс (напр. „$ция“) и др.

2.3. Графови хранилища (Graph stores)

Графовите хранилища с данни съхраняват и анализират графови структури. Този тип хранилища се използват за записване на семантика на данни; записване на връзки и отношения между обекти; обработка на графови структури с цел търсене и вземане на решения и др.

Широко разпространен продукт от този тип е Neo4j – система с отворен код, създадена през 2007 г., написана на Java. При работа със системата се използва език Neo4j CQL (CypherQuery Language). Например чрез средствата на този език можем да подадем към системата заявка като:

```
CREATE ( book : BOOK
        {id : 55, title: “Neo4j Tutorial“, pages : 340, price : 90})
```

Този израз ще предизвика създаване на структурата BOOK, ако тя не съществува и въвеждане на един запис за конкретна книга.

Пример за употреба във Facebook – поддържане на графова структура за всеки потребител, като всеки профил е свързан с неговите приятели, харесвания на някой статуси, изпращани съобщения, коментирани други постове и други много и разнородни данни. При това за работата на системата се поддържат милиони профили.

Системи от този вид се използват за доставка на съдържание за уеб, управление и търсене на документи в предприятия, поддръжка на платформи за игри, медия и др.

2.4. Други системи за управление на данни

Съвременните приложения с бази от данни трябва да обработват не само релационни, а и множество от други данни – чертежи, карти, мултимедия, WWW и др. Те изискват обработка на различни структури. Тези изисквания се удовлетворяват до определена степен от обектно-ориентираните, обектно-релационните, пространствените и дтуги типове системи за управление на данни.

Мултимедийна бааза от данни – следващата стъпка в развитието на текстовите бази от данни. Теорията на „Мултимедийните бази от данни“ се занимава със съхраняване, категоризиране и извличане на мултимедийна информация и търсенето на техни оптимални решения. Мултимедийната информация е съвкупност от различни типове разнородни данни, които са неразривна част от нашия живот.

Например: Британската енциклопедия в първата си електронна версия е организирана като текстова бази от данни с отделни илюстрации, т.е. копие на печатното издание, но в последствие се развива като мултимедийна бази от данни, в която са включени и много други средства за представяне на информация като анимация, компютърна графика, филмови клипове и т.н.

Работата с изображения предполага извличане от тях на текстова информация, която най-често се интегрира във формата на самите изображения.

Мултимедийни системи за управление на бази от данни – това е софтуерът, който осигурява съхраняването, управлението и извличането на мултимедийни данни от мултимедийната база от данни.

От своя страна мултимедийна база от данни е съвкупност от управлявани мултимедийни данни, които могат да бъдат структурирани данни, както и полуструктурирани и неструктурирани данни, като аудио, видео, текст и изображения.

Ако мултимедийните данни се представят в суров вид, те са с големи обеми и съответно по-бавно търсене. Затова тези данни се представят така, че обработването им да е възможно с най-малко операции. За всеки запис системата трябва да може да извлеча максимално семантични знания, както за формата на медийния файл, така и за неговото съдържание.

Обектно-ориентираните бази от данни се основават на основните понятия при обектно-ориентираният подход: класове, обекти, капсулиране, наследяване и полиморфизъм. Основен елемент е обекта. Обектът е свързан с множество от стойности, които го описват (атрибутите в релационния модел); множество съобщения, чрез които обекта кореспондира с другите обекти (relationships); множество методи, които осигуряват изпълнението и обработката на съобщенията. Обектите, поделящи общо множество от характеристики се групират в класове, които се групират в йерархии.

От друга страна обектно-релационните системи за управление на данни са създадени на базата на релационния модел. Но елементите, с които потребителя борави са обекти. Освен това притежават допълнителни възможности за създаване на йерархични системи. Този тип системи стават все по-популярни при създаването на бизнес-ориентирани приложения.

Пространствени бази от данни се използват за обработка на графична информация. Графичната информация може да се представи в растерен формат (като схема от пиксели) или векторен формат (чрез скалиране). Този тип данни може да се опише и чрез многоразмерен куб чрез OLAP операции.

Темпоралните бази от данни са системи за управление на данни, включващи атрибути, свързани с отчитане на времето, които имат различна семантика. От друга страна при времевите редове (Time-series) се съхранява последователността от стойности, които се променят динамично във времето.

3. Парадигмата Big Data

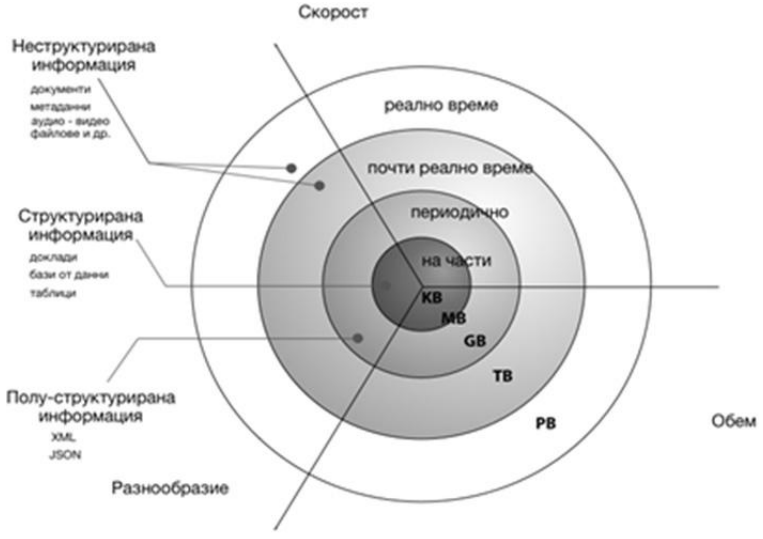
Big data е ново и развиващо се понятие, което описва голям обем структурирани, полу-структурирани и неструктурирани данни. С термина Big Data се означава събирането на големи количества информация, които са толкова обемни и сложни, че е невъзможна тяхната обработка с досегашните СУБД или традиционните приложения за работа с данни. Предизвикателствата, пред които са изправени IT специалистите, включват събиране, съхраняване, търсене, споделяне, прехвърляне, анализиране и визуализиране на данните.

Въпреки че Big Data не се отнася за определено количество данни, терминът се използва за големи информационни масиви от порядъка на петабайти и ексабайти данни.

Например Google search обработва 1 PB данни за 1 час. Потребителите на YouTube ъплодват 100 часа видео за 1 минута и т.н [8].

Предизвикателства пред Big Data

Източниците на данни обикновено са от различен тип, генерираните данни са подчинени на различни стандарти. Често данните постъпват в системата във вид неподходящ за директна обработка и интеграция в платформа, което налага допълнителни стъпки за привеждане на данните във формат, отговарящ на дефинираните правила в платформата [1].



Фигура: 3. Измерения на Big Data

Данните се разделят в три класа според степента си на структурираност: структурирани, неструктурирани и полуструктурирани данни. Пример за структурирани данни са таблици, бази от данни, доклади генерирани от бази от данни и др. Неструктурираните данни се генерират от: интернет на събитията; интернет на хората (социалните мрежи – Facebook, Tweeter, LinkedIN и др.); интернет на нещата; интернет на локацията (мобилни телефони, смартфони, таблети и др.) [11]. Понятието полуструкторирани данни се използва, когато данните се съхраняват във формат XML, JSON или друг.

Характеристиките на Big Data, известни като четерите V се определят като:

- Volume – голям обем (количество) данни;
- Variety – разнообразие на данните (различни форми);
- Velocity – висока скорост на генериране на данни (анализи на потоци от данни);
- Veracity – истинност на данните (сигурността на качеството на данните варира).

Поради обема на големите данни обикновено те не могат да бъдат съхранявани на една машина, защото ще е необходимо много време за тяхната обработка. Стандартно решение е да се използват много машини и данните да се обработват паралелно. В резултат на това могат да се решават мащабни изчислителни задачи като се обработват големи масиви входно/изходни данни (терабайти) и да се извършват много изчисления (стотици процесорни изчисления).

MapReduce е основна парадигма за разпределени изчисления на големи масиви от данни. Това е фреймуърк, който позволява паралелно обработване на големи обеми неструктурирани данни. Осигурява висока устойчивост на откази (сривове). Разработен от Google през 2004 г за индексиране на уеб страници. Днес програмите, написани в MapReduce – стил, са с автоматична паралелизация и се изпълняват на големи клъстери от компютри, като един клъстер може да се състои от стотици или хиляди машини.

Основната идея на тази парадигма е входните файлове да се разделят на M входни части. Отделните входни части се разпределят автоматично от управляваща програмата (master) между една или повече машини (или други части на програмата – workers) и преминават през Map процедурите. Резултатите от Map се записват в междинни файлове и се обработват чрез Reduce - процедурите.

Apache Hadoop е безплатен фреймуърк с отворен код, написан на Java [15]. Поддържа множество от алгоритми за разпределено съхранение и разпределена обработка на много големи масиви от данни на компютърни клъстери. Основни модули:

- Hadoop Distributed File System (HDFS) – разпределена файлова система, която съхранява данни на компютърни системи, предоставя високо агрегиран трафик в рамките на клъстера;
- Hadoop YARN е платформа за управление на ресурсите, отговорна за управление на изчислителните ресурси в клъстера и използването им за потребителските приложения;
- Hadoop Map Reduce – програмен модел за паралелно обработване на големи масиви от данни.

4. Заключение

Съвременните тенденции в компютърните системи са свързани с развитието на високопроизводителни изчисления (HPC) [2] и работа с феномена големи данни (BigData). Обработката и съхранението на големите данни изисква нов поглед и съвместно прилагане на редица утвърдени технологии. Отворени проблеми в направлението са свързани с оптимизиране на достъпа до големи обеми от данни, чрез прилагане на агенти за извличане на знания от данните [4] и съвременни Data mining техники за анализ на данните [6]. Появява се понятието Data Science [14], което се свързва с извличането на предварително неизвестни знания директно от данни, чрез процес на откриване и аналитичен анализ на хипотези във виртуалните образователни пространства.

Благодарности

Изследването е свързано с работата по проект „Иновативни фундаментални и приложни научни изследвания по компютърни науки”, фонд Научни изследвания на Бургаския свободен университет.

Литература

1. Boyd D., Crawford K., Critical questions for Big Data, 2012 уеб ресурс: http://www.tandfonline.com/doi/pdf/10.1080/1369118X.2012.678878
2. Geoffrey Fox, Big Data HPC Convergence and a bunch of other things, 02/04/2016, http://www.slideshare.net/Foxsden/big-data-hpc-convergence-and-a-bunch-of-other-things.Date, C.J. An Introduction to Database Systems, 7th edn. Reading, MA: Addison-Wesley, 2000.
3. Date. C.J., Database in Depth: Relational Theory for Practitioners, Published 2005 by O'Reilly Media.
4. Shen, W., Hao, Q., Yoon, H. J., Norrie, D. H. Applications of agent-based systems in intelligent manufacturing: An updated review, Advanced Engineering INFORMATICS, 20(4), (2006), pp.415–431.
5. Han, Jiawei and Micheline Kamber. 2007. Data Mining: Concepts and Techniques. San Francisco, CA: Morgan Kaufmann publishers.
6. Orozova D, S. Stoyanov, I. Popchev, „Virtual education space“, International Conference „Knowledge – tradition, innovation, perspectives”, Burgas, Bulgaria, ISBN: 978-954-9370-99-7, vol. 4, pp.153–159, 2013.
7. Попчев И., Милена Георгиева, Наукометричен анализ в областта на Big Ddata, БСУ Научна конференция с международно участие, 2015, стр. 469-476.Пенева, Ю., Бази от данни, Изд. „Регалия 6“ София, 2004.
8. Стоянов С., Теоретичен модел на виртуално образователно пространство, Международна конференция „From DeLC to VelSpace”, 26-28 март, Пловдив, ISBN: 0-9545660-2-5, 285-297
9. Орозова Д, М. Георгиева, Приложения на „Големите данни“, Списание „Компютърни науки и комуникации“, Том 3, № 4 (2014), БСУ, Бургас, стр.118-121.
10. Стоянов С., И. Попчев, DeLC – минало, настояще, бъдеще, пленарен доклад, Международна конференция „From DeLC to VelSpace“, 26-28 март, Пловдив, ISBN: 0-9545660-2-5.
11. С.Стоянов, Д.Орозова, И.Попчев, Виртуално образователно пространство – настояще и бъдеще, Юбилейна конференция на БСУ, том. II, стр. 410-418.
12. http://docs.mongodb.org/manual/reference/operator/query/
13. http://hadoop.apache.org
14. Journal of Data Science an international journal devoted to applications of statistical methods at large: http://www.jds-online.com/