mongoDB gyakorlat

2013. április.

Áttekintés

- nosql.ppt
  http://www.wrox.com/WileyCDA/WroxTitle/Professional-NoSQL.productCd-047094224X.html

A mongoDB előnyei

- beépített replikáció: replica sets, több (páratlan számú) mongodb példánnyal. Automatikus failover recovery.
- a kollekciók automatikus, horizontális adat-particionálása: sharding. A belépő új szervereken automatikus terhelés- és adatelosztás (load balancing). A shard key olyan mező, ami a kollekció összes dokumentumában előfordul.
- ilyen funkciók a RDMS-ekben (oracle, stb.) csak a **drága**, high-end változatokban vannak

MongoDB telepítése

- letöltés: http://downloads.mongodb.org/win32/mongodb-win32-i386-2.4.2.zip
- dokumentáció: http://docs.mongodb.org/master/MongoDB-Manual-master.pdf
  http://docs.mongodb.org/manual/single/
- összehasonlítás az SQL-lel: http://docs.mongodb.org/manual/reference/sql-comparison/
- kicsomagolás c:\mongo
- c:\data\db könyvtár létrehozása
- deamon indítása c:\mongo\mongod
- kapcsolódás c:\mongo\mongo

Alapműveletek

tutorial: http://docs.mongodb.org/manual/tutorial/getting-started/

- egyszerű parancsok
  - **help**
  - **db**: az aktív adatbázis neve
  - **show dbs**: az adatbázisok listája
  - **use mydb**: új adatbázis (üres)
  - primitív mezőtípusok: datetime, string, double
  - új dokumentum-kollekció felvétele, melynek a neve things:
    - **j = { name : "mongo" }**
    - **k = { x : 3 }**
    - **db.things.insert(j)**

        - **db.things.insert(k)**
    - **show collections** (a rendszer hozza létre és tartja karban a system.indexes kollekciót is) a mydb ekkor 32 MB-ot foglal el
    - ha nincs a dokumentumban _id mező, akkor a rendszer létrehozza
    - **db.things.remove()**: a kollekció törlése
    - a dokumentum felépítése: kulcs-érték párok, az értékek primitív mezőtípusai: datetime, string, double, bool, …
    - az érték lehet önálló dokumentum vagy primitív értékek/dokumentumok tömbje
    - példa kicsit bonyolultabb dokumentumra JSON (JavaScript Object Notation) formátumban (a mongo BSON, azaz binary JSON formátumban tárolja a dokumentumokat):

```
{
    _id: 1,
    name: { first: 'John', last: 'Backus' },
    birth: new Date('Dec 03, 1924'),
    death: new Date('Mar 17, 2007'),
    contribs: [ 'Fortran', 'ALGOL', 'Backus-Naur Form', 'FP' ],
    awards: [
        {
          award: 'W.W. McDowell Award',
          year: 1967,
          by: 'IEEE Computer Society'
        },
        {
          award: 'National Medal of Science',
          year: 1975,
          by: 'National Science Foundation'
        }
      ]
  }
```

- listázás és lekérdezések
    - **db.things.find()**: teljes listázás
    - **for (var i = 1; i <= 20; i++) db.things.insert({ x : 4 , j : i })** :20 dokumentum beszúrása
    - **it**: iterate, a listázás folytatása
    - kurzor létrehozása és iterálása ciklusban
        - **var c = db.things.find()**
        - **while (c.hasNext()) printjson(c.next())** //json formátum = asszociatív tömb
        - **printjson(c [4])** //a kurzor elérhető tömbként is, ami a 0-s indexszel kezdődik. A rendszer tömbbé alakítja az egész kurzort, és a memóriába tölti.
    - **db.things.find({ name : "mongo" })** : kurzor szűrése
    - **db.things.find({ x : 4 } , { j : 1 })** : a visszaadott mezők korlátozása, 1 megjelenítendő mezőt jelent (az _id, ha nincs 0-val tiltva, magától jön)

- **db.things.findOne()** : a kurzor aktuális dokumentuma
- **db.things.find().limit(3)** : az első 3 dokumentum

MapReduce

- kiindulási kollekció: egy-sok kapcsolat rendelések és tételek, vásárlók és rendelések között

```
db.orders.insert( {
    _id: ObjectId(),
    cust_id: "abc123",
    ord_date: new Date("Oct 04, 2012"),
    status: 'A',
    price: 250,
    items: [ { product: "chair", qty: 10, price: 2 },
           { product: "sofa", qty: 20, price: 3} ]  })

db.orders.insert( {
    _id: ObjectId(),
    cust_id: "abc123",
    ord_date: new Date("Oct 04, 2012"),
    status: 'A',
    price: 200,
    items:  [{ product: "chair", qty: 11, price: 2 }]  })

db.orders.insert( {
    _id: ObjectId(),
    cust_id: "abc456",
    ord_date: new Date("Oct 04, 2012"),
    status: 'A',
    price: 300,
    items: [ { product: "bed", qty: 2, price: 20 },
           { product: "sofa", qty: 20, price: 3} ] })
```

- definiáljuk a map és reduce függvényeket. A map egy kulcsot és egy értéket tartalmazó párosokból álló listát készít, de mind a kulcs, mind az érték lehet teljes dokumentum is. A reduce függvény az első paraméter (a példában keyCustId) szerint csoportosítást végez a sum művelet előtt:

```
var mapFunction1 = function() {
                    emit(this.cust_id, this.price);
                  };
var reduceFunction1 = function(keyCustId, valuesPrices) {
                      return Array.sum(valuesPrices);
                    };
```

- a definiált függvények összekapcsolása és futtatása, az eredmény az "example" kollekcióban

```
db.orders.mapReduce( mapFunction1, reduceFunction1, { out: "example" }    )
```

- bonyolultabb példa az esetek számolásával

```
var mapFunction2 = function() {
                for (var idx = 0; idx < this.items.length; idx++) {
                   var key = this.items[idx].product;
                   var value = {
                              count: 1,
                              qty: this.items[idx].qty
                             };
                   emit(key, value);
                 }
               };
var reduceFunction2 = function(key_product, valuesCountObjects) {
                  reducedValue = { count: 0, qty: 0 };
                  for (var idx = 0; idx < valuesCountObjects.length; idx++) {
                     reducedValue.count += valuesCountObjects[idx].count;
                     reducedValue.qty += valuesCountObjects[idx].qty;
                  }
                  return reducedValue;
                };
var finalizeFunction2 = function (key, reducedValue) {
                   reducedValue.average = reducedValue.qty/reducedValue.count;
                   return reducedValue;
                 };
db.orders.mapReduce( mapFunction2, reduceFunction2,
               {
                 out: { merge: "map_reduce_example" },
                 query: { ord_date: { $gt: new Date('01/01/2012') } },
                 finalize: finalizeFunction2
               }
              )
```

- próbáljuk a lekérdezést futtatni a mongoVUE-ben: http://www.mongovue.com/downloads/
- teszt-adatbázist másoljuk be a mongo/bin könyvtárba: http://media.mongodb.org/zips.json
- mongoimport --db demo --collection zips --file zips.json
- ÖNÁLLÓ FELADAT: számoljuk ki az egyes államok össznépességét a map-reduce alkalmazásával! csoportosítsunk több mező (status, cust_id) szerint az orders kollekcióban!

CRUD: create, read, update, delete

- create: insert, save
- **update** szintaxisa: db.collection.update(<query>, <update>, <options>)
  - a query a dokumentumokat szűri. Alapértelmezésben csak az első illeszkedő dokumentumon végez az update változtatást.
  - az update part operátorai: $set, $unset, $push, $inc, $pull

- a $set módosítja a mezők értékét, ha nincs adott nevű mező, akkor hozzáadja a dokumentumhoz. Pl. az alábbi update az 1 id-jű dokumentum name aldokumentumának middle mezőjét állítja be, vagy hozza létre:

```
db.bios.update(
  { _id: 1 },
  {
    $set: { 'name.middle': 'Warner' },
  }
)
```

- $set hiányában *az összes be nem állított mező törlődik!*
- tömb valamelyik elemének módosítása: **$set: { 'contribs.1': 'ALGOL 58' }**
- ha a tömbpozíció nem ismert, a $ tetszőleges helyet jelent. A példában az első olyan dokumentumot vesszük, melynek id-je 6 és az awards aldokumentum-tömb valamelyik dokumentumában a by mező értéke ACM, majd az első ilyen aldokumetum by mezőjét átírjuk az új értékre

```
db.bios.update(
  { _id: 6, 'awards.by': 'ACM'  } ,
  { $set: { 'awards.$.by': 'Association for Computing Machinery' } }
)
```

- **$push**: új tömbelem beszúrása, **$pull** tömbelem törlése, pl.

```
db.bios.update(
  { _id: 1 },
  {
    $push: { awards: { award: 'IBM Fellow', year: 1963, by: 'IBM' } }
  }
)
```

- az **$unset** törli a mezőt
- **$inc** növeli a mező értékét valamennyivel
- <**options**>: upsert és multi. upsert = update ha van találat, különben insert. multi = nem az első, hanem az összes illeszkedő dokumentumra lefut
  pl. **{ upsert: true }**
-

- ÖNÁLLÓ FELADAT: írjuk át az AL állam nevét ALABAMA-ra a zips kollekcióban
  vigyünk be egy új várost, Veszprém, state:”HU” az upsert alkalmazásával! Ellenőrizzük az új dokumentum meglétét!
- **delete**: a korább remove függvénnyel. Megadható, hogy az első vagy az összes illeszkedő legyen törölve.

db.bios.remove({ 'name.first' : /^G/ }) //mindet törli, ahol az első betű G

db.bios.remove({ turing: true }, 1) //csak az elsőt törli

Adatmodellezés, tervezés

- az RDBMS-hez 3NF-ig normalizáltunk, itt a normalizált és denormalizált formák közti átmenet szükséges (beágyazás vs hivatkozás?). Esetenként más és más megoldás. Példa:

BEÁGYAZVA:

```
{
  title: "MongoDB: The Definitive Guide",
  author: [ "Kristina Chodorow", "Mike Dirolf" ],
  published_date: ISODate("2010-09-24"),
  pages: 216,
  language: "English",
  publisher: {
        name: "O'Reilly Media",
        founded: 1980,
        location: "CA"
       }
}
{
  title: "50 Tips and Tricks for MongoDB Developer",
  author: "Kristina Chodorow",
  published_date: ISODate("2011-05-06"),
  pages: 68,
  language: "English",
  publisher: {
        name: "O'Reilly Media",
        founded: 1980,
        location: "CA"
       }
}
```

HIVATKOZVA:

```
{
  _id: "oreilly",
  name: "O'Reilly Media",
  founded: 1980,
  location: "CA"
}
{
  _id: 123456789,
```

```
  title: "MongoDB: The Definitive Guide",
  author: [ "Kristina Chodorow", "Mike Dirolf" ],
  published_date: ISODate("2010-09-24"),
  pages: 216,
  language: "English",
  publisher_id: "oreilly"
}
{
  _id: 234567890,
  title: "50 Tips and Tricks for MongoDB Developer",
  author: "Kristina Chodorow",
  published_date: ISODate("2011-05-06"),
  pages: 68,
  language: "English",
  publisher_id: "oreilly"
}
```

- szempontok:
  - bár egy kollekció dokumentumai más szerkezetűek lehetnek, általában a legtöbbjük hasonló, és a mezőik hasonló értelműek
  - a várható lekérdezések szerkezetét támogassa a dokumentumok beágyazottsági szintje
  - csak az egy dokumentumon belüli művelet atomicitása biztosított—az egyszerre változó mezők egy dokumentumon belül legyenek (test-and-set jellegű műveletek)
  - a kollekció sharding mezője minden dokumentumban legyen meg

Minta-alkalmazás: webshop

- klasszikus OLTP alkalmazás, amit tipikusan RDBMS szolgál ki. Christian Kvalheim példája a mongodb rugalmasságát mutatja: http://www.infoq.com/articles/data-model-mongodb
- a termék sémája (kellőképpen denormalizált):

```
db.products.insert({
 sku: "111445GB3",
 title: "Simsong One mobile phone",
 description: "The greatest Onedroid phone on the market .....",
 manufacture_details: {
  model_number: "A123X",
  release_date: new ISODate("2012-05-17T08:14:15.656Z")
 },
 shipping_details: {
  weight: 350,
  width: 10,
  height: 10,
  depth: 1
```

```
  },
  quantity: 99,
  pricing: {
    price: 1000
  }
})
```

- utólag hozzáadunk még egy mezőt, a termékkategóriát:

```
db.products.update({sku: "111445GB3"}, {$set: { categories: ['mobile/15G', 'mobile/fm'] }});
```

- indexet készítünk a categories mezőre:

```
db.products.ensureIndex({categories:1 })
```

- a kategóriáknak egy másik, fa szerkezetű kollekciót készítünk:

```
db.categories.insert({title: "Mobiles containing a FM radio", parent: "mobile",  path:
"mobile/fm"})
db.categories.insert({title: "Mobiles with 15G support", parent: "mobile",  path: "mobile/15G"})
db.categories.ensureIndex({parent: 1, path: 1})
 db.categories.ensureIndex({path: 1})
```

- ebben reguláris kifejezéssel keresve pl. megtalálhatjuk a mobile kategória al-kategóriáit (csak a path mezőt jelenítjük meg)

```
db.categories.find({parent: /^mobile/}, {_id: 0, path: 1})
```

- a kosarak kollekciója. Az üzleti logika lényege, hogy a kosarakban lévő mennyiségek soha ne haladják meg a termék mennyiségét, és az otthagyott kosarakból a mennyiségek visszakerüljenek a termékhez.

```
db.carts.insert({
  _id: "the_users_session_id",
  status:'active',
  quantity: 2,
  total: 2000,
  products: []});
```

- a termék bekerül a kosárba

```
 db.carts.update({
    _id: "the_users_session_id", status:'active'
  }, {
    $set: { modified_on: ISODate() },
    $push: {
      products: {
        sku: "111445GB3", quantity: 1, title: "Simsong One mobile phone", price:1000
      }
```

```
    }
  });
```

- csökkentjük a termék mennyiségét és létrehozzuk az in_cart tömböt

```
db.products.update({
    sku: "111445GB3", quantity: {$gte: 1}
  }, {
    $inc: {quantity: -1},
    $push: {
      in_carts: {
        quantity:1, id: "the_users_session_id", timestamp: new ISODate()
      }
    }
  })
```

- ellenőrizzük, hogy sikeres volt-e a művelet (volt-e találat), s ha nem, töröljük a kosárból is:

```
 if(!db.runCommand({getLastError:1}).updatedExisting) {
  db.carts.update({
     _id: "the_users_session_id"
    }, {
     $pull: {products: {sku:"111445GB3"}}
  })
```

- ha a kosárban lévő mennyiség változik, pl. még egyet a kosárba tesz (new_quantity):

```
db.carts.update({
   _id: "the_users_session_id", "products.sku": "111445GB3", status: "active"
 }, {
   $set: {
     modified_on: new ISODate(),
     "products.$.qty": new_quantity
 }
})
```

illetve:

```
 db.products.update({
    sku: "111445GB3",
    "in_carts.id": "the_users_session_id",
    quantity: {
      $gte: 1
    }
  }, {
    $inc: { quantity: (-1)*quantity_delta },
    $set: {
```

```
    "in_carts.$.quantity": new_quantity, timestamp: new ISODate()
   }
  })
```

- az előző művelet hibaellenőrzése:

```
if(!db.runCommand({getLastError:1}).updatedExisting) {
  db.carts.update({
     _id: "the_users_session_id", "products.sku": "111445GB3"
   }, {
     $set : { "in_carts.$.quantity": old_quantity}
   })
 }
```

- ha a kapcsolat megszakad, töröljük a kosarat:

```
var carts = db.carts.find({status:"expiring"})
 for(var i = 0; i < carts.length; i++) {
  var cart = carts[i]

  for(var j = 0; j < cart.products.length; j++) {
   var product = cart.products[i]

   db.products.update({
      sku: product.sku,
      "in_carts.id": cart._id,
      "in_carts.quantity": product.quantity
     }, {
      $inc: {quantity: item.quantity},
      $pull: {in_carts: {id: cart._id}}
     })
  }

  db.carts.update({
     _id: cart._id,
     $set: {status: 'expired'}
   })
 }
```

- a rendelés véglegesítése az orders kollekcióban:

```
db.orders.insert({
  created_on: new ISODate("2012-05-17T08:14:15.656Z"),

  shipping: {
```

```
    customer: "Peter P Peterson",
    address: "Longroad 1343",
    city: "Peterburg",
    region: "",
    state: "PE",
    country: "Peteonia",
    delivery_notes: "Leave at the gate",

    tracking: {
     company: "ups",
     tracking_number: "22122X211SD",
     status: "ontruck",
     estimated_delivery: new ISODate("2012-05-17T08:14:15.656Z")
    },
   },

   payment: {
    method: "visa",
    transaction_id: "2312213312XXXTD"
   }

   products: {
    {quantity: 2, sku:"111445GB3", title: "Simsong mobile phone", unit_cost:1000,
  currency:"USDA"}
   }
  })
```

ennyi!