Linux and SQL

Date: 03-23-2017

Abstract: This document provides information for interview purpose so the content is selective. Linux knowledge is not for system administration or basic usage of the windowing system or programming toolsor software. It is about system build-in commands and scripting capability.

Chapter 1 Linux Knowledge

1. Basic command checklist

Directory: ～，cd, rm, mkdir,mv, ls, cp, cat, chown, chgrp,chmod, ln

Process: **ps, kill**, at, bg,fg, crontab, nice, time, **id, whoami, who, what,** which,df, du,nohup

Text: touch, vi, wc, tr, split, cut, sort, join, file, head, tail, tee, uname

Permission: ulimit, umask,

Scripting: **alias, unalias,** sh, **sed, awk**, pr, sleep, test, read, diff, dirname, uniq, basename, nl, less, more, echo, printf

Searching: **find, grep, exec, xargs**

Network: netcat

Documentation: man

Software Development: ar, make, nm, yacc, lex, strip, ctags

System administration: mount, umount , lsof...etc

1. Disk and partition (24-30)

IDE:

SCSI:

Logical partition:

Extended partion:

Install linux to logical partition? (21)

1. Runlevel

3: multiuser without GUI

5: multiuser with GUI

(TODO: how to switch runlevel when login , where to set runlevel when boot in)

4/ Firewall (less important)

/etc/iptables

(TODO:)

1. Startup scripts

/etc/init.d/rc1.d, rc2.d...

/etc/rc.d/init.d

Redhat (fedora)

Canonical LTD (ubuntu)

iso, sun virtual box

1. Networking

34, 35, ifconfig

7. How DNS works

8. Profile files

/etc/profile

~/.bash\_profile

~/.bash\_rc

9/ /etc/passwd

/etc/shadow

/etc/fstab

(Need to write a little bit)

10.Server config

Ssh: /etc/ssh/sshd\_config

/etc/hosts.allow, /etc/hosts.deny

(TODO: write a little bit about it)

11. Security

12. Text process

How to kill one process with a name called java in one line?

A: kill -9 `ps axuw|grep java| grep -v grep| awk ‘{print $2}’`

How to kill all the processes with a name in one line? For example, kernels processes.

A:

*for pid in $(ps -ef | grep "some search" | awk '{print $2}'); do kill -9 $pid; done*

There are ways to make that more efficient,

*for pid in $(ps -ef | awk '/some search/ {print $2}'); do kill -9 $pid; done*

How to find out a file named “hello\*.txt”in all the directory underneath?

A: *find ~ -name “hello\*.txt”*

For example,

cd ~,

touch hello.txt; touch hello1.txt; touch hello2.txt

find ~ -name “hello\*.txt”

How to delete a file in all places with a name hello\*.txt in above example?

How to find out all the file names that contains “hello world”?

How to delete all the files that contain “hello world”?

grep “hello world” hello1.txt

(will show it contains or not, grep needs to specify a pattenr and files, note files can be empty or directory)

Find . -type f -exec grep “hello world” {} \;

(will show it in all the files found, but it is not these files, it just displays)

In order to show files, do this,

grep -rnw '/path/to/somewhere/' -e "pattern"

* -r or -R is recursive,
* -n is line number, and
* -w stands for match the whole word.
* -l (lower-case L) can be added to just give the file name of matching files.

Along with these, --exclude, --include, --exclude-dir or --include-dir flags could be used for efficient searching:

* This will only search through those files which have .c or .h extensions:

grep --include=\\*.{c,h} -rnw '/path/to/somewhere/' -e "pattern"

* This will exclude searching all the files ending with .o extension:

grep --exclude=\*.o -rnw '/path/to/somewhere/' -e "pattern"

Just like exclude files, it's possible to exclude/include directories through --exclude-dir and --include-dir parameter. For example, this will exclude the dirs dir1/, dir2/ and all of them matching \*.dst/:

grep --exclude-dir={dir1,dir2,\*.dst} -rnw '/path/to/somewhere/' -e "pattern"

* This works very well for me, to achieve almost the same purpose like yours.

How to show a pid and ppid of a process?

A: To see every process in standard way, ps -ef or ps -ely,

To see every process in bsd way, ps aux

12 VIM

How to replace a text once?

How to replace a text in all places?

How to go to the end or front of a large file.

How to go forward a page and backward a page?

How to copy and paste?

How to delete a line?

Chapter 2 SQL Knowledge

What is primary key?

A column or multiple columns that can be used to identify an unique record/row.

Primary key can not be null, and can only have one primary key set for a table.

Primary key is always indexed by default.

What is a unique key?

The value of this column is unique. The value can be null, once null it is not unique.

A table can have many unique keys.

What is foreign key?

A column in this table but it is a primary key in another table.

Foreign key can be use in combination with other key as a primary key.

What is a key?

Key is a column used in concept with primary key and foreign key.

What is a index?

Index is some extra bit in the data structure of b/b+ tree of a column. This is to make the column searched faster.

\* Index makes where and groupby faster but make insert and udpate slower.

\* Index makes joins faster.

\* Index makes variable length column slower.

What is the difference between index and key?

Key is a column. Index is an extra bit in B/B+ tree.

What is group by?

What is having?

How to optimiaze code with database as storage?

1. Less DB call: Code optimaztion, more oo design, save queried result in object and reduce dao.db call everywhere.
2. Faster query: Add index for frequently searched column, make query faster.

3. Less join: To many join, relation complex, use big flat table, reduce join.

4. Network, bandwidth, switch...etc.

5. Architecture, distributed database.

6. Load balancing: DB shading, keep certain amount of records in one db, and move other records to different db.

Chapter 3 DBMS

1. MySQL and MariaDB (free open source version MySQL since MySQL has commercial version.)

2. HBase

Distributed database used by hadoop and hdfs. Please check other documentation to fill out this part.

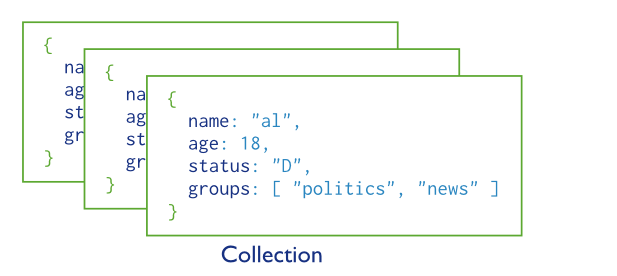
Chapter 4 NoSQL

4.1 MongoDB

MongoDB stores all documents in [collections](https://docs.mongodb.org/manual/reference/glossary/" \l "term-collection). A collection is a group of related documents that have a set of shared common indexes. Collections are analogous to a table in relational databases and each record is implemented in a way called BSON, binary json.

Collection: table

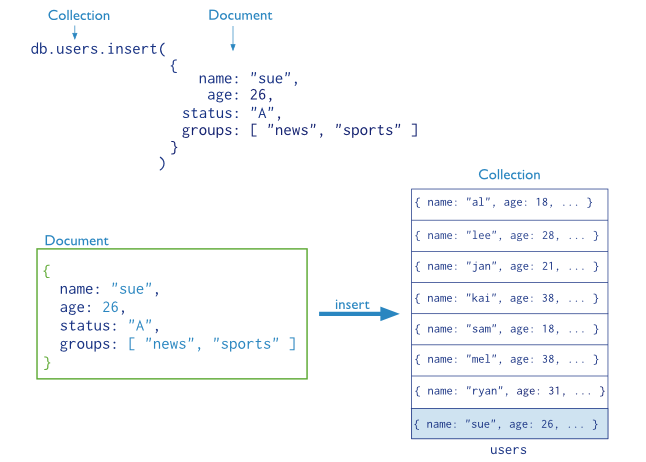
Document: record (BSON)



BSON



Architecture,



4.1.1 Database Binary:

Mongod: server

Mongo: client

4.1.2 Start server in windows:

1. Automatic creation of windows service

"C:\mongodb\bin\mongod.exe" --config "C:\mongodb\mongod.cfg" --install

"C:\mongodb\bin\mongod.exe" --remove

Net start MongoDB

Net stop MongoDB

4.2 Usage and syntax

1/ create collection

*C:\Users\linfeng>mongo*

*2017-03-24T15:47:53.437-0400 I CONTROL [main] Hotfix KB2731284 or later update*

*is not installed, will zero-out data files*

*MongoDB shell version: 3.2.4*

*connecting to: test*

*>*

The following creates table and insert a record,

**CREATE** **TABLE** users (

id MEDIUMINT **NOT** **NULL**

AUTO\_INCREMENT,

user\_id Varchar(30),

age Number,

status char(1),

**PRIMARY** **KEY** (id))

**db**.**users**.**insert**( { **user\_id**: "abc123", **age**: 55, **status**: "A" } )

Output:

*connecting to: test*

*>*

*> CREATE TABLE users (*

*... id MEDIUMINT NOT NULL*

*... AUTO\_INCREMENT,*

*... user\_id Varchar(30),*

*... age Number,*

*... status char(1),*

*... PRIMARY KEY (id))*

*2017-03-24T15:49:53.726-0400 E QUERY [thread1] SyntaxError: missing ; before*

*statement @(shell):1:7*

*> db.users.insert( { user\_id: "abc123", age: 55, status: "A" } )*

*WriteResult({ "nInserted" : 1 })*

*>*

*When you do this, it warns you that the collection already existed.*

*> db.createCollection("users")*

*{ "ok" : 0, "errmsg" : "collection already exists", "code" : 48 }*

*The above sql creates not only a table similar to the collection.*

*If you don’t specify a primary, an auto id will be set as primary key.*

2/ ALTER TABLE users ADD join\_date DATETIME

Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level.

However, at the document level, [update()](https://docs.mongodb.org/manual/reference/method/db.collection.update/" \l "db.collection.update" \o "db.collection.update()) operations can add fields to existing documents using the [$set](https://docs.mongodb.org/manual/reference/operator/update/set/" \l "up._S_set" \o "$set) operator.

db.users.update(

{ },

{ $set: { join\_date: new Date() } },

{ multi: true })

3/ ALTER TABLE users DROP COLUMN join\_date

Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level.

However, at the document level, [update()](https://docs.mongodb.org/manual/reference/method/db.collection.update/" \l "db.collection.update" \o "db.collection.update()) operations can remove fields from documents using the [$unset](https://docs.mongodb.org/manual/reference/operator/update/unset/" \l "up._S_unset" \o "$unset) operator.

db.users.update(

{ },

{ $unset: { join\_date: "" } },

{ multi: true }

)

4/ DROP TABLE users

db.users.drop()

5/ **CREATE** **INDEX** idx\_user\_id\_asc**ON** users(user\_id)

**db**.**users**.**createIndex**( { **user\_id**: 1 } )

**6/ CREATE** **INDEX**

idx\_user\_id\_asc\_age\_desc**ON** users(user\_id, age **DESC**)

**db**.**users**.**createIndex**( { **user\_id**: 1, **age**: -1 } )

7/ delete records

A. **DELETE** **FROM** users**WHERE** status = "D"

**db**.**users**.**remove**( { **status**: "D" } )

B. **DELETE** **FROM** users

**db**.**users**.**remove**({})

8/ update records

A. **UPDATE** users **SET** status = "C" **WHERE** age > 25

**db**.**users**.**update**( { **age**: { **$gt**: 25 } }, { **$set**: { **status**: "C" } }, { **multi**: **true** })

B. **UPDATE** users **SET** age = age + 3 **WHERE** status = "A"

**db**.**users**.**update**( { **status**: "A" } , { **$inc**: { **age**: 3 } }, { **multi**: **true** })

9/ Select

<https://docs.mongodb.org/manual/reference/sql-comparison/>

4.3 Import data

C:\Program Files\MongoDB\dataset.jason

Ref: <https://docs.mongodb.org/getting-started/java/import-data/>

import sample database to Mongodb

Eg:

C:\Users\linfeng>mongoimport --db test --collection restaurants --drop --file "c

:\Program Files\MongoDB\dataset.json"

2016-03-25T00:11:24.643-0500 connected to: localhost

2016-03-25T00:11:24.646-0500 dropping: test.restaurants

2016-03-25T00:11:25.884-0500 imported 25359 documents

4.4 java driver and bson jar

mongo-java-driver-3.0.4.jar and bson-3.0.4.jar in C:\Program Files\MongoDB.

4.5 connect to mongodb and insert data

Import com.mongodb.MongoClient;

Import com.mongodb.client.MongoDatabase;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Locale;

import static java.util.Arrays.asList;

MongoClient mongoClient = new MongoClient();

MongoDatabase db = mongoClient.getDatabase("test");

DateFormat format = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'", Locale.ENGLISH);db.getCollection("restaurants").insertOne(

new Document("address",

new Document()

.append("street", "2 Avenue")

.append("zipcode", "10075")

.append("building", "1480")

.append("coord", asList(-73.9557413, 40.7720266)))

.append("borough", "Manhattan")

.append("cuisine", "Italian")

.append("grades", asList(

new Document()

.append("date", format.parse("2014-10-01T00:00:00Z"))

.append("grade", "A")

.append("score", 11),

new Document()

.append("date", format.parse("2014-01-16T00:00:00Z"))

.append("grade", "B")

.append("score", 17)))

.append("name", "Vella")

.append("restaurant\_id", "41704620"));

4.6 find and query

import org.bson.Document;

import com.mongodb.Block;

import com.mongodb.client.FindIterable;

import static com.mongodb.client.model.Filters.\*;

import static com.mongodb.client.model.Sorts.ascending;

import static java.util.Arrays.asList;

Query for All Documents in a Collection

FindIterable<Document> iterable = db.getCollection("restaurants").find();

iterable.forEach(new Block<Document>() {

@Override

public void apply(final Document document) {

System.out.println(document);

}});

Specify Equality Conditions

If the <field> is in an embedded document or an array, use [dot notation](https://docs.mongodb.org/manual/reference/glossary/" \l "term-dot-notation" \o "(in mongodb-manual v3.2)) to access the field.

To help specify the query condition, the Java driver also provides the [Filters](http://api.mongodb.org/java/3.0/com/mongodb/client/model/Filters.html) class. The class contains various static methods to simplify building the query predicates, including the [eq](http://api.mongodb.org/java/3.0/com/mongodb/client/model/Filters.html" \l "eq-java.lang.String-TItem-) method:

eq(<field>, <value>)

**Query by a Top Level Field**

FindIterable<Document> iterable = db.getCollection("restaurants").find(

new Document("borough", "Manhattan"));

Iterate the results and apply a block to each resulting document.

iterable.forEach(new Block<Document>() {

@Override

public void apply(final Document document) {

System.out.println(document);

}});

Using the static [Filters](http://api.mongodb.org/java/3.0/com/mongodb/client/model/Filters.html) helper(s), you can also specify the query as follows:

db.getCollection("restaurants").find(eq("borough", "Manhattan"));

More:

<https://docs.mongodb.org/getting-started/java/query/>

4.7 update data

4.8 remove data

4.9 data aggregation

4.10 FAQ

1. MongoDB does not support SQL
2. MongoDB does not support transaction
3. MongoDb handles cache
4. To list a collection’s indexes, use the [db.collection.getIndexes()](https://docs.mongodb.org/manual/reference/method/db.collection.getIndexes/" \l "db.collection.getIndexes" \o "db.collection.getIndexes()) method.
5. > db.users.getIndexes()

[

{

"v" : 1,

"key" : {

"\_id" : 1

},

"name" : "\_id\_",

"ns" : "test.users"

}

]

6. How can I see the size of an index?

The [db.collection.stats()](https://docs.mongodb.org/manual/reference/method/db.collection.stats/" \l "db.collection.stats" \o "db.collection.stats()) includes an [indexSizes](https://docs.mongodb.org/manual/reference/command/collStats/" \l "collStats.indexSizes" \o "collStats.indexSizes) document which provides size information for each index on the collection.

Depending on its size, an index may not fit into RAM. An index fits into RAM when your server has enough RAM available for both the index and the rest of the [working set](https://docs.mongodb.org/manual/reference/glossary/" \l "term-working-set). When an index is too large to fit into RAM, MongoDB must read the index from disk, which is a much slower operation than reading from RAM.

Cassendra

Chapter 5 New DBMS

OrientDB

InfluxDB (graph db)