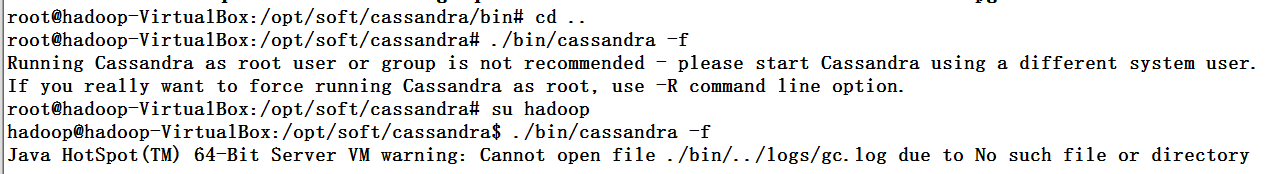
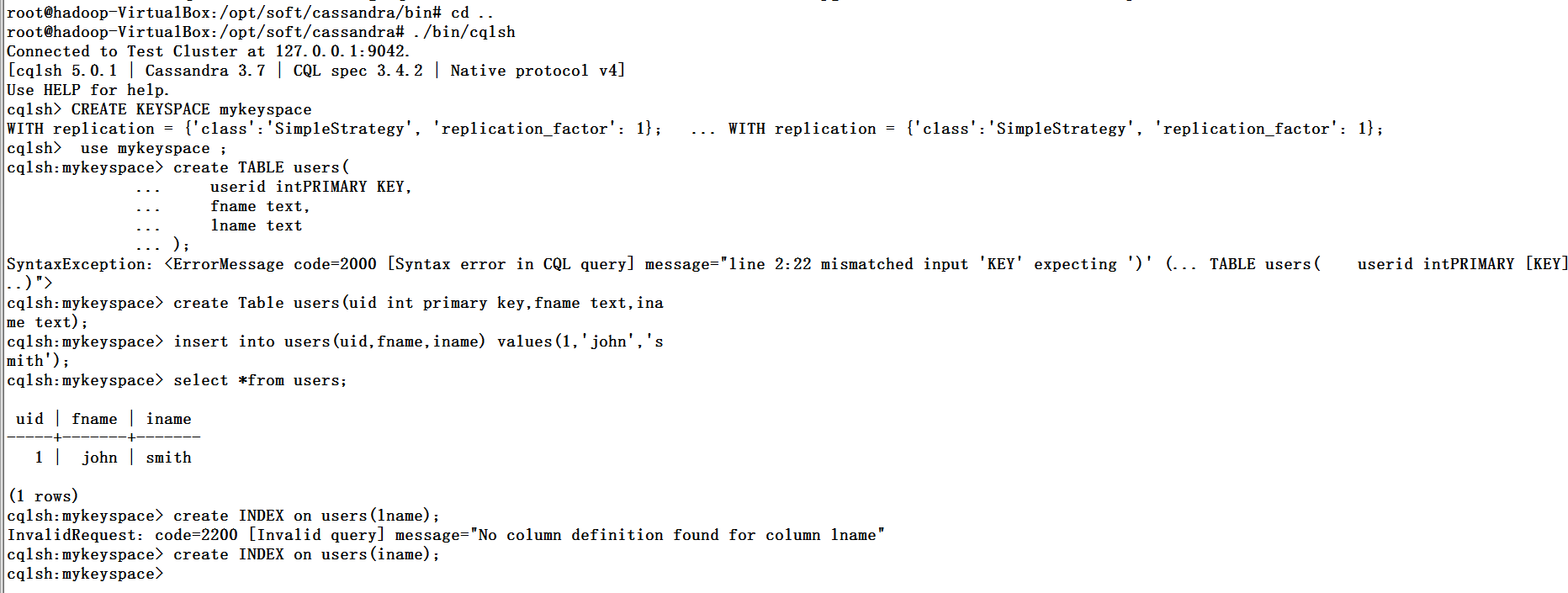
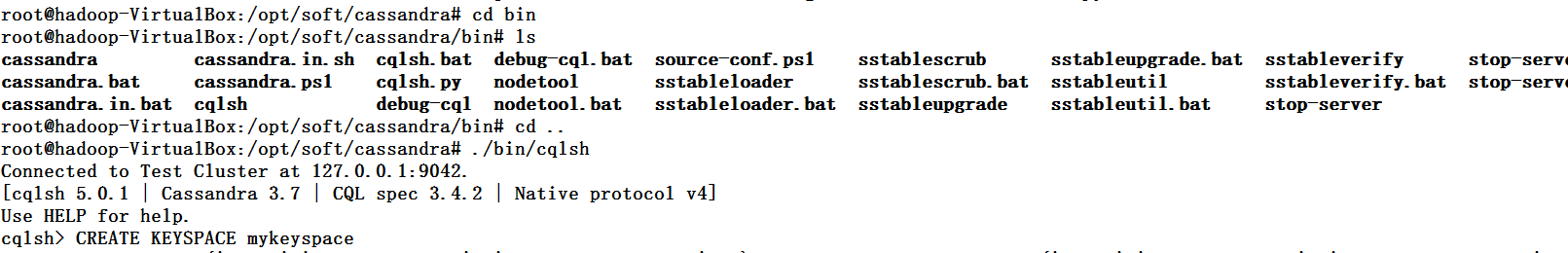
Cass启动命令：

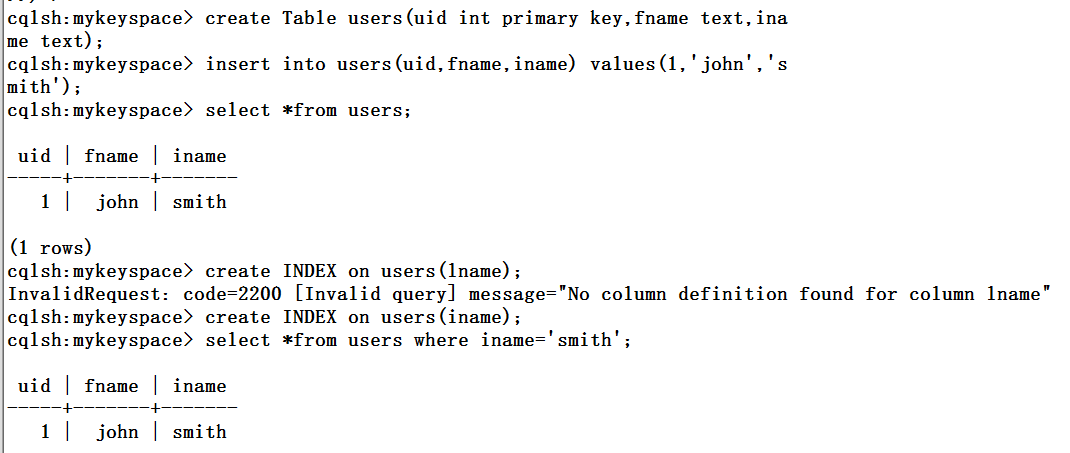


打开执行命令窗口：

Cql：







CREATE TABLE t1 (

id text PRIMARY KEY,

name text,

favs map<text, text>);

INSERT INTO t1 (id, name, favs)

VALUES ('jsmith', 'John Smith', { 'fruit' : 'Apple', 'band' : 'Beatles' });

// Replace the existing map entirely.

UPDATE t1 SET favs = { 'fruit' : 'Banana' } WHERE id = 'jsmith';

UPDATE t1 SET favs['author'] = 'Ed Poe' WHERE id = 'jsmith';

UPDATE t1 SET favs = favs + { 'movie' : 'Cassablanca', 'band' : 'ZZ Top' } WHERE id = 'jsmith';

emoving one or more element (if an element doesn’t exist, removing it is a no-op but no error is thrown):

DELETE favs['author'] FROM t1 WHERE id = 'jsmith';

UPDATE t1 SET favs = favs - { 'movie', 'band'} WHERE id = 'jsmith';

Lastly, TTLs are allowed for both INSERT and UPDATE, but in both case the TTL set only apply to the newly inserted/updated elements. In other words:

UPDATE t1 USING TTL 10 SET favs['color'] = 'green' WHERE id = 'jsmith';

**Sets**

A set is a (sorted) collection of unique values. You can define and insert a map with:

CREATE TABLE images (

name text PRIMARY KEY,

owner text,

tags set<text> // A set of text values

);

INSERT INTO images (name, owner, tags)

VALUES ('cat.jpg', 'jsmith', { 'pet', 'cute' });

// Replace the existing set entirely

UPDATE images SET tags = { 'kitten', 'cat', 'lol' } WHERE name = 'cat.jpg';

Further, sets support:

* Adding one or multiple elements (as this is a set, inserting an already existing element is a no-op):
* UPDATE images SET tags = tags + { 'gray', 'cuddly' } WHERE name = 'cat.jpg';
* Removing one or multiple elements (if an element doesn’t exist, removing it is a no-op but no error is thrown):
* UPDATE images SET tags = tags - { 'cat' } WHERE name = 'cat.jpg';

Lastly, as for [maps](http://cassandra.apache.org/doc/latest/cql/types.html#maps), TTLs if used only apply to the newly inserted values.

**Lists**

Note

As mentioned above and further discussed at the end of this section, lists have limitations and specific performance considerations that you should take into account before using them. In general, if you can use a [set](http://cassandra.apache.org/doc/latest/cql/types.html#sets) instead of list, always prefer a set.

A list is a (sorted) collection of non-unique values where elements are ordered by there position in the list. You can define and insert a list with:

CREATE TABLE plays (

id text PRIMARY KEY,

game text,

players int,

scores list<int> // A list of integers

)

INSERT INTO plays (id, game, players, scores)

VALUES ('123-afde', 'quake', 3, [17, 4, 2]);

// Replace the existing list entirely

UPDATE plays SET scores = [ 3, 9, 4] WHERE id = '123-afde';

Further, lists support:

* Appending and prepending values to a list:
* UPDATE plays SET players = 5, scores = scores + [ 14, 21 ] WHERE id = '123-afde';
* UPDATE plays SET players = 6, scores = [ 3 ] + scores WHERE id = '123-afde';
* Setting the value at a particular position in the list. This imply that the list has a pre-existing element for that position or an error will be thrown that the list is too small:
* UPDATE plays SET scores[1] = 7 WHERE id = '123-afde';
* Removing an element by its position in the list. This imply that the list has a pre-existing element for that position or an error will be thrown that the list is too small. Further, as the operation removes an element from the list, the list size will be diminished by 1, shifting the position of all the elements following the one deleted:
* DELETE scores[1] FROM plays WHERE id = '123-afde';
* Deleting *all* the occurrences of particular values in the list (if a particular element doesn’t occur at all in the list, it is simply ignored and no error is thrown):
* UPDATE plays SET scores = scores - [ 12, 21 ] WHERE id = '123-afde';