

 무료 전자 책

배우기

hive

Free unaffiliated eBook created from
Stack Overflow contributors.

#hive

.....	1
1:	2
.....	2
Examples.....	2
.....	2
(linux)	3
Linux Metastore	4
2: HIVE	7
Examples.....	7
SEQUENCEFILE.....	7
ORC.....	7
.....	7
AVRO.....	8
.....	8
3: SELECT	9
.....	9
Examples.....	9
.....	9
.....	9
:	10
4: Sqoop	11
.....	11
.....	11
Examples.....	11
.....	11
5:	12
.....	12
.....	12
Examples.....	13
.....	13
.....	14

ACID	14
HIVE_HBASE	14
.....	14
6:	16
.....	16
.....	16
Examples.....	16
.....	16
.....	16
7: (UDAF)	18
Examples.....	18
UDAF	18
8: (UDTF)	20
Examples.....	20
UDTF	20
9:	23
Examples.....	23
.....	23
10:	24
Examples.....	24
.....	24
.....	24
.....	24
.....	24
.....	24
.....	25
:	25
.....	25
.....	25
.....	25
.....	25
.....	26
.....	26

11: (UDF 's)	27
Examples.....	27
UDF	27
UDF.....	27
12:	29
Examples.....	29
.....	29
.....	30

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [hive](#)

It is an unofficial and free hive ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official hive.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1:

- Hive [Hadoop](#) .
- SQL .
- Hadoop SQL ., Hadoop Cluster map-reduce . Hive .

Examples

()

.
. .
.

```
CREATE TABLE FILES (line STRING);

LOAD DATA INPATH 'docs' OVERWRITE INTO TABLE FILES;

CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
(SELECT explode(split(line, ' ')) AS word FROM FILES) w
GROUP BY word
ORDER BY word;
```

word_counts

, 2
, 1
a, 1
, 1
, 2
, 1
, 1
, 2
, 1
as, 1
, 1

1
, 1
, 1
, 1
, 1
, 1
~, 1
, 1

(linux)

<https://hive.apache.org/downloads.html> .

-> .

\$ tar -xvf hive-2.xy-bin.tar.gz

-> /usr/local/ .

\$ sudo mkdir /usr/local/hive

-> .

\$ mv ~/Downloads/hive-2.xy/usr/local/hive

-> .bashrc hadoop

\$ gedit ~/ .bashrc

export HIVE_HOME = /usr/local/hive/apache-hive-2.0.1-bin/

export PATH = \$ PATH : \$ HIVE_HOME / bin

export CLASSPATH = \$ CLASSPATH : /usr/local/Hadoop/lib/* :.

export CLASSPATH = \$ CLASSPATH : /usr/local/hive/apache-hive-2.0.1-bin/lib/* :.

-> , hadoop . .

\$ hadoop fs -mkdir /user/hive/warehouse

" " .

\$ hadoop fs -mkdir /tmp

```
"tmp" .
```

```
-> / .
```

```
$ hadoop fs -chmod g + / /
```

```
$ hadoop fs -chmod g + / tmp
```

```
-> HIVE
```

```
$
```

Linux Metastore

```
:
```

1. Java 7
2. Hadoop (Hadoop)
3. MySQL

```
:
```

```
1 : .
```

```
2 : tarball ( : $HOME tarball )
```

```
tar -xvf /home/username/apache-hive-x.y.z-bin.tar.gz
```

```
3 : ( ~/.bashrc )
```

```
export HIVE_HOME=/home/username/apache-hive-x.y.z-bin  
export PATH=$HIVE_HOME/bin:$PATH
```

```
.
```

```
source ~/.bashrc
```

```
4 : mysql JDBC .
```

```
tar -xvf mysql-connector-java-a.b.c.tar.gz
```

```
connector jar mysql-connector-java-abcjar.$HIVE_HOME lib .
```

```
cp mysql-connector-java-a.b.c.jar $HIVE_HOME/lib/
```

```
:
```

```
$HIVE_HOME/conf/ directory hive-site.xml .
```



```

<configuration>
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://localhost/hive_meta</value>
    <description>JDBC connect string for a JDBC metastore</description>
  </property>

  <property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
    <description>Driver class name for a JDBC metastore</description>
  </property>

  <property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>mysqluser</value>
    <description>username to use against metastore database</description>
  </property>

  <property>
    <name>javax.jdo.option.ConnectionPassword</name>
    <value>mysqlpass</value>
    <description>password to use against metastore database</description>
  </property>

  <property>
    <name>datanucleus.autoCreateSchema</name>
    <value>>false</value>
  </property>

  <property>
    <name>datanucleus.fixedDatastore</name>
    <value>>true</value>
  </property>
</configuration>

```

MySQL "username" "password" .

Metastore :

```
$HIVE_HOME/scripts/metastore/upgrade/mysql/
```

MySQL .

```

mysql -u username -ppassword

mysql> create database hive_meta;
mysql> use hive_meta;
mysql> source hive-schema-x.y.z.mysql.sql;
mysql> exit;

```

Metastore :

```
hive --service metastore
```

```
nohup hive --service metastore &
```

HiveServer2 : ()

```
hiveserver2
```

,

```
nohup hiveserver2 metastore &
```

```
: $HIVE_HOME/bin/
```

```
:
```

```
hive hive , beeline .
```

Hive CLI Beeline Hue .

```
:
```

```
$HUE_HOME/desktop/conf/hue.ini .
```

```
[beeswax]
```

```
hive_conf_dir=/home/username/apache-hive-x.y.z-bin/conf
```

: [https://riptutorial.com/ko/hive/topic/1099/-](https://riptutorial.com/ko/hive/topic/1099/)

2: HIVE

Examples

SEQUENCEFILE

SEQUENCEFILE . Gzip Bzip2 TextFile . .

```
CREATE TABLE raw_sequence (line STRING)
STORED AS SEQUENCEFILE;
```

ORC

ORC (Optimized Row Columnar) Hive . . . ORC . ORC .

: <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC>

ORC HortonWorks .

```
CREATE TABLE tab_orc (col1 STRING,
                      col2 STRING,
                      col3 STRING)
STORED AS ORC
TBLPROPERTIES (
  "orc.compress"="SNAPPY",
  "orc.bloom.filter.columns"="col1",
  "orc.create.index" = "true"
)
```

ORC :

```
ALTER TABLE T SET FILEFORMAT ORC;
```

Hive 0.14 CONCATENATE ORC . . .

```
ALTER TABLE T [PARTITION partition_spec] CONCATENATE;
```

Hive 0.13.0 . Dremel . . .

(parquet) . . (parquet) , . .

Cloudera .

: <http://parquet.apache.org/documentation/latest/>

```
CREATE TABLE parquet_table_name (x INT, y STRING) STORED AS PARQUET;
```

AVRO

Avro Hive 0.14.0 .

Avro Apache Hadoop . JSON . Apache Hadoop Hadoop Hadoop .

AVRO : <https://avro.apache.org/docs/1.7.7/spec.html>

```
CREATE TABLE kst
PARTITIONED BY (ds string)
STORED AS AVRO
TBLPROPERTIES (
  'avro.schema.url'='http://schema_provider/kst.avsc');
```

```
CREATE TABLE kst (field1 string, field2 int)
PARTITIONED BY (ds string)
STORED AS AVRO;
```

STORED AS AVRO .

```
ROW FORMAT SERDE
  'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS INPUTFORMAT
  'org.apache.hadoop.hive ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive ql.io.avro.AvroContainerOutputFormat'
```

TextFile hive.default.fileformat . . , csv ID, , "" . .

```
CREATE TABLE employees (id int, name string, salary double) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',';
```

```
LOAD DATA LOCAL INPATH '/home/ourcsvfile.csv' OVERWRITE INTO TABLE employees;
```

:

```
SELECT * FROM employees;
```

HIVE : <https://riptutorial.com/ko/hive/topic/4513/hive-->

3: SELECT

- SELECT [| DISTINCT] select_expr, select_expr, select_expr,
- FROM table_reference
- [WHERE where_condition]
- [GROUP BY col_list]
- []
- [col_list]
- [LIMIT n]

Examples

```
SELECT . . .
```

```
SELECT Name, Position
FROM Employees;
```

```
* . . .
```

```
SELECT *
FROM Employees;
```

```
amount > 10 sales region = "US" . . .
```

```
SELECT * FROM sales WHERE amount > 10 AND region = "US"
```

```
. name address . . .
```

```
SELECT name, address.* FROM Employees
```

```
LIKE ('%' ) . . . city "New" . . .
```

```
SELECT name, city FROM Employees WHERE city LIKE 'New%'
```

```
RLIKE Java . . . name "smith" "son" . . .
```

```
SELECT name, address FROM Employee WHERE name RLIKE '.*(smith|son).*'
```

```
. . .
```

```
SELECT upper(name) FROM Employees
```

```
, , , , . . .
```

```
LIMIT . 10 . . .
```

```
SELECT * FROM Employees LIMIT 10
```

:

(Employee)

	INT
F_Name	
L_Name	

* .

```
Select * from Employee
```

(ID,)

.

```
Select ID, Name from Employee
```

1

1 .

```
Select `(ID)?+.` from Employee
```

. :NAME .

```
Select `(. *NAME$)?+.` from Employee
```

SELECT : <https://riptutorial.com/ko/hive/topic/4133/select->

4: Sqoop

HDQ Sqoop CREATE TABLE . Hive Sqoop --hive-import .

RDBMS HIVE . () Hive .

--hive-import Sqoop HDFS .

--hive-table . RDBMS .

Examples

```
$ sqoop import --connect jdbc:mysql://10.0.0.100/hadooptest
--username hadoopuser -P
--table table_name --hive-import --hive-table hive_table_name
```

Sqoop : <https://riptutorial.com/ko/hive/topic/10685/sqoop---->

5:

- CREATE [TEMPORARY] [EXTERNAL] TABLE [] [db_name.] table_name
[col_name data_type [COMMENT col_comment], ...] [CLISTTERED BY (col_name, col_name, ...) [col_comment] (: Hive 0.10.0)] ON ((col_value, col_value, col_value, col_value, col_value, col_name, ...) ...), (col_value, col_value, ...) [DOIRECTORIES]
[[ROW FORMAT row_format] [] | STORED BY
'storage.handler.class.name'[SERDEPROPERTIES (...)] [hdfs_path] [TBLPROPERTIES
(property_name = property_value, ...)]
[AS select_statement];
- CREATE [TEMPORARY] [EXTERNAL] TABLE [] [db_name.] table_name LIKE
existing_table_or_view_name [LOCATION hdfs_path];
- data_type : primitive_type, array_type, map_type, struct_type, union_type
- primitive_type : TINYINT, SMALLINT, INT, BIGINT, BOOLEAN, FLOAT, DOUBLE, STRING, BINARY, TIMESTAMP, DECIMAL, DECIMAL (,), DATE, VARCHAR, CHAR
- array_type : ARRAY < >
- map_type : MAP <primitive_type, data_type>
- struct_type : STRUCT <col_name : data_type [col_comment], ...>
- union_type : UNIONTYPE < , , ...>
- row_format : DELIMITED [FIELDS TERMINATED BY char [ESCAPED BY char]] [char]
[MAP KEYS TERMINATED BY char] [LINES TERMINATED BY char] [NULL DEFINED AS
char]
, SERDE serde_name [SERDEPROPERTIES (property_name = property_value,
property_name = property_value, ...)]
- file_format : : SEQUENCEFILE, TEXTFILE, RCFILE, ORC, PARQUET, AVRO,
INPUTFORMAT input_format_classname OUTPUTFORMAT output_format_classname
- CREATE (DATABASE | SCHEMA) [] database_name [COMMENT database_comment]
[LOCATION hdfs_path] [WITH DBPROPERTIES (property_name = property_value, ...)];

HIVE . . .

- use database; use database; use database;
- SELECT current_database()
- create table DDL SHOW CREATE TABLE tablename
- DESCRIBE tablename serde DESCRIBE FORMATTED tablename . DESCRIBE DESC .

Examples

. Ctrl-A (^A) . hive-site.xml hive.metastore.warehouse.dir .

```
CREATE TABLE view
(time INT,
id BIGINT,
url STRING,
referrer_url STRING,
add STRING COMMENT 'IP of the User')
COMMENT 'This is view table'
PARTITIONED BY(date STRING, region STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\001'
STORED AS SEQUENCEFILE;
```

. ctrl-A , .

```
CREATE EXTERNAL TABLE view
(time INT,
id BIGINT,
url STRING,
referrer_url STRING,
add STRING COMMENT 'IP of the User')
COMMENT 'This is view table'
PARTITIONED BY(date STRING, region STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\001'
STORED AS SEQUENCEFILE
LOCATION '<hdfs_location>';
```

CTAS (Select Table As Select) .

CTAS , SELECT HiveQL SELECT . CTAS CREATE SELECT SerDe .

CTAS .

- .
- .
- .

```
CREATE TABLE new_key_value_store
ROW FORMAT SERDE "org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe"
STORED AS RCFile
AS
SELECT * FROM page_view
SORT BY url, add;
```

:

LIKE CREATE TABLE . CTAS , . .

```
CREATE TABLE empty_page_views
```

```
LIKE page_views;
```

```
CREATE DATABASE IF NOT EXISTS db_name  
COMMENT 'TEST DATABASE'  
LOCATION /PATH/HDFS/DATABASE/;
```

ACID .

0.14 ACID . UPDATE / DELETE / INSERT .

hive-site.xml .

```
hive.support.concurrency = true  
hive.enforce.bucketing = true  
hive.exec.dynamic.partition.mode = nonstrict  
hive.txn.manager =org.apache.hadoop.hive.ql.lockmgr.DbTxnManager  
hive.compactor.initiator.on = true  
hive.compactor.worker.threads = 1
```

orc .

```
create table Sample_Table(  
col1 Int,  
col2 String,  
col3 String)  
clustered by (col3) into 3 buckets  
stored as orc  
TBLPROPERTIES ('transactional'='true');
```

HIVE_HBASE

Hive-Hbase . : 0.11.0 HBase : 0.94.2 : 0.20.2

```
CREATE TABLE hbase_hive  
(id string,  
col1 string,  
col2 string,  
col3 int)  
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'  
WITH SERDEPROPERTIES  
("hbase.columns.mapping" = ":key,cf1:col1,cf1:col2,cf1:col3")  
TBLPROPERTIES ("hbase.table.name" = "hive_hbase");
```

:

.

```
CREATE TABLE new_table_name LIKE existing_table_name;
```

: <https://riptutorial.com/ko/hive/topic/3328/---->

6:

- :
- INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1 = val1, partcol2 = val2 ...) []] select_statement1 FROM from_statement;
- INSERT INTO TABLE tablename1 [(partcol1 = val1, partcol2 = val2 ...)] select_statement1 FROM from_statement;
- INSERT INTO TABLE tablename1 [PARTITION (partcol1 = val1, partcol2 = val2 ...) (z, y) select_statement1 FROM from_statement;
- ():
- FROM from_statement
INSERT OVERWRITE TABLE tablename1 [(partcol1 = val1, partcol2 = val2 ...) []]
select_statement1
[INSERT OVERWRITE TABLE tablename2 [PARTITION ... []] select_statement2]
[INSERT INTO TABLE tablename2 [PARTITION ...] select_statement2] ...;
- FROM from_statement
INSERT INTO TABLE tablename1 [(partcol1 = val1, partcol2 = val2 ...)] select_statement1
[INSERT INTO TABLE tablename2 [PARTITION ...] select_statement2]
[tablename2 [... []] select_statement2] ...;
- ():
- INSERT OVERWRITE TABLE tablename PARTITION (partcol1 [= val1], partcol2 [= val2] ...) select_statement FROM from_statement;
- INSERT INTO TABLE PARTITION (partcol1 [= val1], partcol2 [= val2] ...) select_statement FROM from_statement;

select . DML . select . .

insert into select .

Examples

```
insert overwrite table yourTargetTable select * from yourSourceTable;
```

INSERT INTO .

```
INSERT INTO table yourTargetTable SELECT * FROM yourSourceTable;
```

```
INSERT INTO TABLE yourTargetTable PARTITION (state=CA, city=LIVERMORE)
select * FROM yourSourceTable;
```

Dynamic Partition inserts are disabled by default. These are the relevant configuration properties for dynamic partition inserts:

```
SET hive.exec.dynamic.partition=true;
SET hive.exec.dynamic.partition.mode=non-strict
```

```
INSERT INTO TABLE yourTargetTable PARTITION (state=CA, city=LIVERMORE) (date,time)
select * FROM yourSourceTable;
```

():

```
FROM table_name

INSERT OVERWRITE TABLE table_one SELECT table_name.column_one,table_name.column_two

INSERT OVERWRITE TABLE table_two SELECT table_name.column_two WHERE table_name.column_one
== 'something'
```

: <https://riptutorial.com/ko/hive/topic/1744/>

7: (UDAF)

Examples

UDAF

- org.apache.hadoop.hive.ql.exec.hive.UDAF **Java** UDAFEvaluator

- 5

- init() - . Column () .
- iterate() - . (.). true .
- terminatePartial() - Hive . . .
- merge() - Hive .
- terminate() - .

```
public class MeanUDAF extends UDAF {
// Define Logging
static final Log LOG = LogFactory.getLog(MeanUDAF.class.getName());
public static class MeanUDAFEvaluator implements UDAFEvaluator {
/**
 * Use Column class to serialize intermediate computation
 * This is our groupByColumn
 */
public static class Column {
double sum = 0;
int count = 0;
}
private Column col = null;
public MeanUDAFEvaluator() {
super();
init();
}
// A - Initialize evaluator - indicating that no values have been
// aggregated yet.
public void init() {
LOG.debug("Initialize evaluator");
col = new Column();
}
// B- Iterate every time there is a new value to be aggregated
public boolean iterate(double value) throws HiveException {
LOG.debug("Iterating over each value for aggregation");
if (col == null)
throw new HiveException("Item is not initialized");
col.sum = col.sum + value;
col.count = col.count + 1;
return true;
}
// C - Called when Hive wants partially aggregated results.
public Column terminatePartial() {
LOG.debug("Return partially aggregated results");
return col;
}
// D - Called when Hive decides to combine one partial aggregation with another
```

```
public boolean merge(Column other) {
LOG.debug("merging by combining partial aggregation");
if(other == null) {
return true;
}
col.sum += other.sum;
col.count += other.count;
return true;
}
// E - Called when the final result of the aggregation needed.
public double terminate(){
LOG.debug("At the end of last record of the group - returning final result");
return col.sum/col.count;
}
}
}
```

```
hive> CREATE TEMPORARY FUNCTION <FUNCTION NAME> AS 'JAR PATH.jar';
hive> select id, mean_udf(amount) from table group by id;
```

(UDAF) : <https://riptutorial.com/ko/hive/topic/5137/-----udaf->

8: (UDTF)

Examples

UDTF

org.apache.hadoop.hive.ql.udf.generic.GenericUDTF

```
1.we specify input and output parameters
abstract StructObjectInspector initialize(ObjectInspector[] args)
                                throws UDFArgumentException;

2.we process an input record and write out any resulting records
abstract void process(Object[] record) throws HiveException;

3.function is Called to notify the UDTF that there are no more rows to process.
   Clean up code or additional output can be produced here.
abstract void close() throws HiveException;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.apache.hadoop.hive.ql.exec.UDFArgumentException;
import org.apache.hadoop.hive.ql.metadata.HiveException;
import org.apache.hadoop.hive.ql.udf.generic.GenericUDTF;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspectorFactory;
import org.apache.hadoop.hive.serde2.objectinspector.PrimitiveObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.StructObjectInspector;
import
org.apache.hadoop.hive.serde2.objectinspector.primitive.PrimitiveObjectInspectorFactory;

public class NameParserGenericUDTF extends GenericUDTF {
    private PrimitiveObjectInspector stringOI = null;

    //Defining input argument as string.
    @Override
    public StructObjectInspector initialize(ObjectInspector[] args) throws
UDFArgumentException {
        if (args.length != 1) {
            throw new UDFArgumentException("NameParserGenericUDTF() takes exactly one
argument");
        }

        if (args[0].getCategory() != ObjectInspector.Category.PRIMITIVE
            && ((PrimitiveObjectInspector) args[0]).getPrimitiveCategory() !=
PrimitiveObjectInspector.PrimitiveCategory.STRING) {
            throw new UDFArgumentException("NameParserGenericUDTF() takes a string as a
parameter");
        }
    }
}
```



```

// input
stringOI = (PrimitiveObjectInspector) args[0];

// output
List<String> fieldNames = new ArrayList<String>(2);
List<ObjectInspector> fieldOIs = new ArrayList<ObjectInspector>(2);
fieldNames.add("name");
fieldNames.add("surname");
fieldOIs.add(PrimitiveObjectInspectorFactory.javaStringObjectInspector);
fieldOIs.add(PrimitiveObjectInspectorFactory.javaStringObjectInspector);
return ObjectInspectorFactory.getStandardStructObjectInspector(fieldNames, fieldOIs);
}

public ArrayList<Object[]> processInputRecord(String name){
    ArrayList<Object[]> result = new ArrayList<Object[]>();

    // ignoring null or empty input
    if (name == null || name.isEmpty()) {
        return result;
    }

    String[] tokens = name.split("\\s+");

    if (tokens.length == 2){
        result.add(new Object[] { tokens[0], tokens[1] });
    }else if (tokens.length == 4 && tokens[1].equals("and")){
        result.add(new Object[] { tokens[0], tokens[3] });
        result.add(new Object[] { tokens[2], tokens[3] });
    }

    return result;
}

@Override
public void process(Object[] record) throws HiveException {
    final String name = stringOI.getPrimitiveJavaObject(record[0]).toString();
    ArrayList<Object[]> results = processInputRecord(name);

    Iterator<Object[]> it = results.iterator();

    while (it.hasNext()){
        Object[] r = it.next();
        forward(r);
    }
}

@Override
public void close() throws HiveException {
    // do nothing
}
}

```

jar .

```
hive> CREATE TEMPORARY FUNCTION process_names as 'jar.path.NameParserGenericUDTF';
```

Here we will pass input as full name and break it into first and last name.

```
hive> SELECT
```

```
t.name,  
t.surname  
FROM people  
  lateral view process_names(name) t as name, surname;  
  
Teena Carter  
John Brownewr
```

(UDTF) : <https://riptutorial.com/ko/hive/topic/6502/-----udtf->

9:

Examples

```
CREATE INDEX index_name
ON TABLE base_table_name (col_name, ...)
AS 'index.handler.class.name'
[WITH DEFERRED REBUILD]
[IDXPROPERTIES (property_name=property_value, ...)]
[IN TABLE index_table_name]
[PARTITIONED BY (col_name, ...)]
[
  [ ROW FORMAT ...] STORED AS ...
  | STORED BY ...
]
[LOCATION hdfs_path]
[TBLPROPERTIES (...)]
```

:

```
CREATE INDEX index_salary ON TABLE employee(salary) AS
'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED REBUILD;
```

ALTER INDEX index_name ON [PARTITION (...)] REBUILD

```
DROP INDEX <index_name> ON <table_name>
```

WITH DEFERRED REBUILD CREATE INDEX ().

ALTER INDEX REBUILD .

: <https://riptutorial.com/ko/hive/topic/6365/>


```
insert into all_floating_numeric_types values (1.4E-45,4.9E-324);
insert into all_floating_numeric_types values (3.4028235E38,1.7976931348623157E308);
```

```
CREATE TABLE all_binary_types(
  c_boolean boolean,
  c_binary binary
);
```

:

```
insert into all_binary_types values (0,1234);
insert into all_binary_types values (1,4321);
```

:

- boolean true false .
- base64 .

```
CREATE TABLE array_data_type(
  c_array array<string>)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '&';
```

data.csv :

```
arr1&arr2
arr2&arr4
```

/tmp data.csv .

```
LOAD DATA LOCAL INPATH '/tmp/data.csv' INTO TABLE array_data_type;
```

CSV /tmp HDFS . HDFS CSV

```
LOAD DATA INPATH '/tmp/data.csv' INTO TABLE array_data_type;
```

```
CREATE TABLE map_data_type(
  c_map map<int,string>)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '&'
MAP KEYS TERMINATED BY '#';
```

data.csv :

```
101#map1&102#map2
103#map3&104#map4
```

:

```
LOAD DATA LOCAL INPATH '/tmp/data.csv' INTO TABLE map_data_type;
```

```
CREATE TABLE struct_data_type(
  c_struct struct<c1:smallint,c2:varchar(30)>)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '&';
```

data.csv :

```
101&struct1
102&struct2
```

:

```
LOAD DATA LOCAL INPATH '/tmp/data.csv' INTO TABLE struct_data_type;
```

```
CREATE TABLE uniontype_data_type(
  c_uniontype uniontype<int, double, array<string>)>
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '&';
```

data.csv :

```
0&10
1&10.23
2&arr1&arr2
```

:

```
LOAD DATA LOCAL INPATH '/tmp/data.csv' INTO TABLE uniontype_data_type;
```

: <https://riptutorial.com/ko/hive/topic/5067/---->

11: (UDF 's)

Examples

UDF

UDF UDF (org.apache.hadoop.hive.ql.exec.UDF) evaluate .

UDF JAR / JAR .

```
import org.apache.hadoop.hive.ql.exec.UDF;

class UDFExample extends UDF {

    public String evaluate(String input) {

        return new String("Hello " + input);
    }
}

hive> ADD JAR <JAR NAME>.jar;
hive> CREATE TEMPORARY FUNCTION helloworld as 'package.name.UDFExample';
hive> select helloworld(name) from test;
```

UDF.

```
package MyHiveUDFs;

import org.apache.commons.lang.StringUtils;
import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.Text;

public class Strip extends UDF {

    private Text result = new Text();
    public Text evaluate(Text str) {
        if(str == null) {
            return null;
        }
        result.set(StringUtils.strip(str.toString()));
        return result;
    }
}
```

jar .

Hive CLI UDF JAR

```
hive> ADD jar /home/cloudera/Hive/hive_udf_trim.jar;
```

JAR CLI .

```
hive> list jars;  
/home/cloudera/Hive/hive_udf_trim.jar
```

```
hive> CREATE TEMPORARY FUNCTION STRIP AS 'MyHiveUDFs.Strip';
```

UDF

```
hive> select strip('  hiveUDF ') from dummy;  
OK  
hiveUDF
```

(UDF 's) : <https://riptutorial.com/ko/hive/topic/4949/-----udf--s->

12:

Examples

/tmp/ca_employees

```
OVERWRITE LOCAL DIRECTORY INSERT '/tmp/ca_employees' SELECT , , FROM se.state = 'CA';
```

```
FROM employees se INSERT OVERWRITE DIRECTORY '/tmp/or_employees' SELECT * WHERE se.cty = 'US' and se.st = 'OR'
INSERT OVERWRITE DIRECTORY '/tmp/ca_employees' SELECT * WHERE se.cty = 'US' and se.st = 'CA'
INSERT OVERWRITE DIRECTORY '/tmp/il_employees' SELECT * WHERE se.cty = 'US' and se.st = 'IL';
```

: <https://riptutorial.com/ko/hive/topic/6530/-->

S. No		Contributors
1		Bhavesh , Community , franklinsijo , johnnyaug , NeoWelkin
2	HIVE	agentv , Alex , Community , johnnyaug , leftjoin , Mzzzzzz , NeoWelkin , tomek , Venkata Karthik
3	SELECT	Ambrish , dev , Jaime Caffarel , johnnyaug
4	Sqoop	NeoWelkin
5		CodingInCircles , dev ヽ , goks , Panther , Venkata Karthik
6		Jared , johnnyaug , Venkata Karthik
7	(UDAF)	dev ヽ , Venkata Karthik
8	(UDTF)	Venkata Karthik
9		Prem Singh Bist
10		dev ヽ
11	(UDF 's)	Ashok , Panther , Venkata Karthik
12		Prem Singh Bist