

Impala权限管理机制

在Impala中，权限管理的作用主要是确定某个用户是否有权访问某些资源，用户对于这些资源具有哪种访问权限等，这里涉及到三个概念：用户，资源和权限。对于Impala 1.1之后的版本，可以直接集成Apache Sentry服务来实现Impala的权限管理，由于Impala可以和hive共享元数据库，包括权限机制。

- 用户

在不开启权限认证的impala集群中，没有用户的概念存在了，所有访问Impala集群的操作都是使用impala账号执行的；对于开启权限的Impala集群，则使用的是运行impala-shell脚本的账号作为访问用户（Linux用户）；对于使用kerberos认证的集群，impala的访问用户则是当前kinit认证的用户，除此之外，还可以通过使用代理的方式提交查询（经过测试发现不可以直接通过UserGroupInformation代理，但是可以通过thrift接口代理，例如hue），此时访问用户是被代理用户。

- 权限（操作）

SELECT权限：允许执行查询操作，包括查询数据和元数据。INSERT权限：写入操作，主要指INSERT和LOAD DATA操作。ALL权限：对于表和数据库的DDL操作，对于URI的create external table之类的操作也需要该权限。以上权限是和资源有密切关系的，对于不同的资源，不同的操作需要不同的权限。如下图：

Operation	Scope	Privileges	URI
EXPLAIN	TABLE; COLUMN	SELECT	
LOAD DATA	TABLE	INSERT	URI
CREATE DATABASE	SERVER	ALL	
DROP DATABASE	DATABASE	ALL	
CREATE TABLE	DATABASE	ALL	
DROP TABLE	TABLE	ALL	
DESCRIBE TABLE -Output shows <i>all</i> columns if the user has table level-privileges or SELECT privilege on at least one table column	TABLE	SELECT/INSERT	
ALTER TABLE .. ADD COLUMNS	TABLE	ALL on DATABASE	
ALTER TABLE .. REPLACE COLUMNS	TABLE	ALL on DATABASE	
ALTER TABLE .. CHANGE column	TABLE	ALL on DATABASE	
ALTER TABLE .. RENAME	TABLE	ALL on DATABASE	
ALTER TABLE .. SET TBLPROPERTIES	TABLE	ALL on DATABASE	
ALTER TABLE .. SET FILEFORMAT	TABLE	ALL on DATABASE	
ALTER TABLE .. SET LOCATION	TABLE	ALL on DATABASE	URI

<http://blog.csdn.net/yu616568>

ALTER TABLE .. ADD PARTITION	TABLE	ALL on DATABASE	
ALTER TABLE .. ADD PARTITION location	TABLE	ALL on DATABASE	URI
ALTER TABLE .. DROP PARTITION	TABLE	ALL on DATABASE	
ALTER TABLE .. PARTITION SET FILEFORMAT	TABLE	ALL on DATABASE	
ALTER TABLE .. SET SERDEPROPERTIES	TABLE	ALL on DATABASE	
CREATE VIEW -This operation is allowed if you have column-level SELECT access to the columns being used.	DATABASE; SELECT on TABLE;	ALL	
DROP VIEW	VIEW/TABLE	ALL	
ALTER VIEW	You need ALL privilege on the named view and the parent database, plus SELECT privilege for any tables or views referenced by the view query. Once the view is created or altered by a high-privileged system administrator, it can be queried by a lower-privileged user who does not have full query privileges for the base tables.	ALL, SELECT	

<http://blog.csdn.net/yu616568>

ALTER TABLE .. SET LOCATION	TABLE	ALL on DATABASE	URI
CREATE EXTERNAL TABLE	Database (ALL), URI (SELECT)	ALL, SELECT	
SELECT -You can grant the SELECT privilege on a view to give users access to specific columns of a table they do not otherwise have access to. -See Column-level Authorization for details on allowed column-level operations.	VIEW/TABLE; COLUMN	SELECT	
USE <dbName>	Any		
CREATE FUNCTION	SERVER	ALL	
DROP FUNCTION	SERVER	ALL	
REFRESH <table name> or REFRESH <table name> PARTITION (<partition_spec>)	TABLE	SELECT/INSERT	
INVALIDATE METADATA	SERVER	ALL	
INVALIDATE METADATA <table name>	TABLE	SELECT/INSERT	
COMPUTE STATS	TABLE	ALL	
SHOW TABLE STATS, SHOW PARTITIONS	TABLE	SELECT/INSERT	
SHOW COLUMN STATS	TABLE	SELECT/INSERT	og.csdn.net/yu616568
SHOW FUNCTIONS	DATABASE	SELECT	
SHOW TABLES		No special privileges needed to issue the statement, but only shows objects you are authorized for	
SHOW DATABASES, SHOW SCHEMAS		No special privileges needed to issue the statement, but only shows objects you are authorized for	og.csdn.net/yu616568

• 资源

- Server: 服务标识，因为sentry可以同时被多个服务集成，因此需要通过该标识指定服务。
- URI: 文件标识符，通常是HDFS文件路径。
- Database: 数据库

- Table: 数据库表
- Column: 表字段，在CDH 5.5 / Impala 2.3之后的版本支持。

除此之外资源是有层次关系的，上面的资源层级关系如下：

```
Server
URI
Database
  Table
    Column
```

当对某个用户授予某个资源的某种权限之后，默认情况下资源会集成这种权限，例如对于某个用户授予了Database A的读权限，那么该用户就可以访问Database A下面的所有表的所有字段了。

并不是所有的资源都需要SELECT和INSERT权限，他们的关系如下：

Valid privilege types and objects they apply to

Privilege	Object
INSERT	DB, TABLE
SELECT	DB, TABLE, VIEW, COLUMN
ALL	SERVER, TABLE, DB, URI

配置Impala权限机制

开启impala的权限机制是针对每一个impalad的，权限认证（impala层）只在impalad的front层做，所以权限是由访问的impalad节点决定，因此可能存在不同的impalad节点权限机制不一样。impala可以支持两种形式的权限管理：基于Sentry服务和基于配置文件（这个配置文件其实就是Sentry的配置文件）。对于impalad开启权限机制需要如下的流程：

- 1.配置-servername参数，该参数为impala开启了sentry认证，它的值是当前服务的标识，通常使用--servername=server1。
- 2.如果使用sentry服务进行授权（非CM部署），那么就需要通过参数--sentry_config=指定sentry的配置（sentry-site.xml配置路径），主要是指定sentry服务的ip和端口等信息。
- 3.如果使用配置文件的方式（较早的版本使用），则需要指定--authorizationpolicyfile=/user/impala/policy.cfg --authorization_policyproviderclass=org.apache.sentry.provider.file.LocalGroupResourceAuthorizationProvider参数，前者指定详细的配置文件的路径（该文件是HDFS的路径），后者指定policy类（这样意味着它是可以扩展的）。
- 4.使用基于文件的方式会周期性的加载配置文件以判断是否有权限的更新，目前该配置不是配置项，配置为5分钟。而依赖于sentry服务的方式是实时的通过sentry获取的，所以可以通过调用sentry接口以修改权限。

使用Sentry服务

授权方式：直接通过hive或者impala的GRANT和REVOKE语句更新权限配置。

使用配置文件的方式

授权方式：离线的修改配置文件并更新到HDFS上，等待下一次刷新policy配置生效。

文件格式：可以适用普通的ini文件格式，文件被分成四个区块：users、groups、roles和databases。对于roles区块，格式如下：

```
server=server_name->db=database_name->table=table_name->action=SELECT
server=server_name->db=database_name->table=table_name->action=CREATE
server=server_name->db=database_name->table=table_name->action=ALL
```

最常用的是groups和roles，前者对组到角色的映射，后者给每一个角色授予每一个资源的权限。users可以指定用户到组的映射，而不是依赖于OS级或者HDFS提供的愈合组的映射关系。对于使用kerberos方式的访问，用户就是执行kinit的用户或者被代理的用户（通过代理的方式被代理用户甚至可以是任何不存在的kerberos用户，例如corp账号），而通过users的配置就可以找到该用户所归属的组，进而通过groups中的配置找到改组对应的role，然后再通过role中的资源权限配置信息确定用户的操作是否被授权。

除此之外，还可以通过配置文件为不同的database指定不同的配置文件，这也就是databases这个区块的作用，例如：

```
[databases]
# Defines the location of the per-DB policy files for the 'customers' and 'sales' databases.
customers = hdfs://ha-nn-uri/etc/access/customers.ini
sales = hdfs://ha-nn-uri/etc/access/sales.ini
```

分别为customers和sales指定了不同的配置文件，然后在不同的配置文件里面指定不同资源的授权访问情况。不过需要开启该机制需要在启动时加上JAVA操作：-Dsentry.allow.uri.db.policyfile=true

配置实例

```
[groups]
admin_group = all_database
music_group = music_role
epay_group = epay_role

[roles]

all_database = server=server1->uri=hdfs://hadoop460.lt.163.org:9000/, \
    server=server1

music_role = server=server1->uri=hdfs://hadoop460.lt.163.org:9000/warehouse/music.db/, \
    server=server1->db=music

epay_role = server=server1->uri=hdfs://hadoop460.lt.163.org:9000/warehouse/epay_db.db/ , \
    server=server1->uri=hdfs://hz-cluster3/user/epay/ , \
    server=server1->db=datawarehouse_delta , \
    server=server1->db=datawarehouse_dm , \
    server=server1->db=datawarehouse_dwa , \
    server=server1->db=datawarehouse_dwd

[users]
nrpt = admin_group
impala = admin_group
hue = admin_group
da_music = music_group
epay = epay_group
```

该文件配置了一个权限机制，存在五个用户，分为三个用户组，每一个用户组拥有自己的权限，例如musicrole角色，它对于server1（也就是impala集群的servename），然后对于这个URI拥有ALL的访问权限，对于server1中music数据库拥有ALL权限。epay_role类似这样的设置，不过它授予一个路径的ALL权限，但是授予多个数据库的ALL权限。

除此之外，还存在*标识符，表示对于每一个资源都授予相同的action。

Apache Ranger集成

由于公司使用Apache Ranger作为权限授权服务，而impala本身不能集成ranger服务，因此需要特殊的方案与ranger集成。首先对于ranger而言，用户是普通的用户（corp账号）而不是kinit认证的用户（公共账号），这样的话映射到impala权限配置文件就需要配置不同的corp账号映射到不同的组（可以是公共账号），每一个用户可以映射到多个组，这种关系可以直接从ranger中获取到，毕竟ranger也是通过这个映射来获取真正执行任务的公共账号是哪个，然是仅仅讲公共账号作为组还是不够的，因为每一个账号下可能存在多个角色，每一个角色需要对应一个组，然后再在groups中设置组和impala角色的映射，最后再根据实际的每一个角色的授权信息以确定每一个用户的访问权限。


最后，由于使用基于文件的方案，所以权限的更新不是实时生效的。

细粒度权限

有些场景下，用户需要更细粒度的权限，也就是列粒度的权限，目的是指定不同的用户只能访问某一个表的某些表，而另外一些用户只能看到另外一些列，在Impala集成Sentry服务的情况下是可以完成这种控制的，另外通过上面的权限和资源关系图可以看出COLUMN的权限只需要SELECT，可以通过如下的语句授予和收回某个角色某个列的权限。

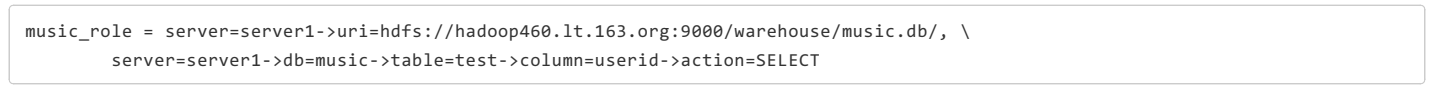
```
GRANT SELECT(column_name) ON TABLE table_name TO ROLE role_name;
REVOKE SELECT(column_name) ON TABLE table_name FROM ROLE role_name;
```

但是上面的场景仅限于集成Sentry服务的情况下，而当使用配置文件的时候，执行如下的查询则会出现这样的异常：



```
AnalysisException: Cannot execute authorization statement using a file based policy. To disable file based policies, restart Impal
```

可以看出GRANT/REVOKE语句是不能在该场景下执行的，毕竟Impala对于配置文件是只读的，但是通过在配置文件中设置COLUMN的权限让Impala重新加载：



```
music_role = server=server1->uri=hdfs://hadoop460.lt.163.org:9000/warehouse/music.db/, \  
server=server1->db=music->table=test->column=userid->action=SELECT
```

该配置对于music这个角色设置对于test这个表userid的SELECT权限，但是通过测试发现“show tables”并不会返回test表，但是执行“select userid from test”是可以正确返回结果的，这可能是因为元数据获取的时候需要获取整个表的信息（没有处理每一个用户获取不同的列的情况）。除此之外，对于column权限的配置不能够在一个entry中配置多个列，所以当用户A需要访问test表中的A，B，C列时需要配置三个sentry，参见[SENTRY-74](#)。