Hive Authorization

Shreepadma Venugopalan
(shreepadma@cloudera.com)

Motivation

Much of Hive's popularity with users is a result of its ability to access data in a Hadoop cluster like in a relational database system such as MySQL, Postgres, Oracle etc. As Hive is being increasingly adopted by the users of a traditional RDBMS, these users have come to expect from Hive the features and functionality that are standard in a traditional RDBMS. Among other things, traditional RDBMSs have supported a rich catalog system and fine grained role based authorization system to restrict access to database objects such as tables, indexes, schemas, rows, columns etc and to resources such as connect, CPU etc. Users of Hive expect a fine-grained authorization system whose semantics mirror that of a traditional RDBMS.

While this project aims to build a fine grained authorization system for Hive, it is important to note that the benefits of this project go beyond Hive. Other language runtimes such as Pig will be able to leverage this project when they access Hive's data and metadata.

Requirements/Use Cases

In the absence of a working authorization system for Hive, users of Hive access the underlying Hadoop cluster either in an un secure manner or use coarse grained HDFS file permissions. While HDFS file permissions provide coarse grained authorization, it is unable to provide fine grained RDBMS like authorization that users expect from Hive. Furthermore, the file permission checks are performed at runtime resulting in run time errors when users don't have the necessary permissions.

The new authorization system for Hive should address the following use cases/requirements,

- Allow the ability to create and administer roles for users much like in an RDBMS.
- Allow users to QUERY only those tables/views for which their role has the necessary privileges.
- Allow users to LOAD and INSERT into only those tables for which their role has the necessary privileges.
- Allow users to QUERY only those columns in a table for which their role has the necessary privileges. This will be achieved through the use of views.
- Provide finer-grained metadata read privileges that allow users to only view metadata of the object. Data read privilege on the object will subsume the metadata read privilege on the object e.g., users who have the privilege to QUERY a table will also be allowed to view the table metadata.
- Allow only those users whose role has the necessary privileges to CREATE tables.

- Allow only those users whose role has the necessary privileges to CREATE databases.
- Allow only those users whose role has the necessary privileges to DROP tables.
- Allow only those users whose role has the necessary privileges to ALTER tables.
- Don't allow users to circumvent any of the above through other operations. For instance, if a user doesn't have privileges to QUERY a table, then the user shouldn't be allowed to circumvent it through other mechanisms e.g., by creating a view on top of a table.
- Log audit information that allows admins to determine access patterns.
- Allow users to access only those rows in a table that match a predicate.

Goals

- The goal of an authorization system is to enforce access control that prevents unauthorized and/or malicious users from compromising the system. While the ultimate goal of the project is a completely secure authorization system, it is possible that initial versions of our system might have security holes that malicious users may exploit. We will plug these holes as and when they are exposed and converge to a secure system. It is important to understand that such security is an ongoing effort.
- A desired goal of the project is to define the authorization model in such a way that it can be extended to add in finer grained privileges later.
- Authorization will be supported independent of the underlying authentication source.
- In version 1 of the project, row level access control will be achieved through the use of views.
- In version 1 of the project, column level access control will be achieved through the use of views.

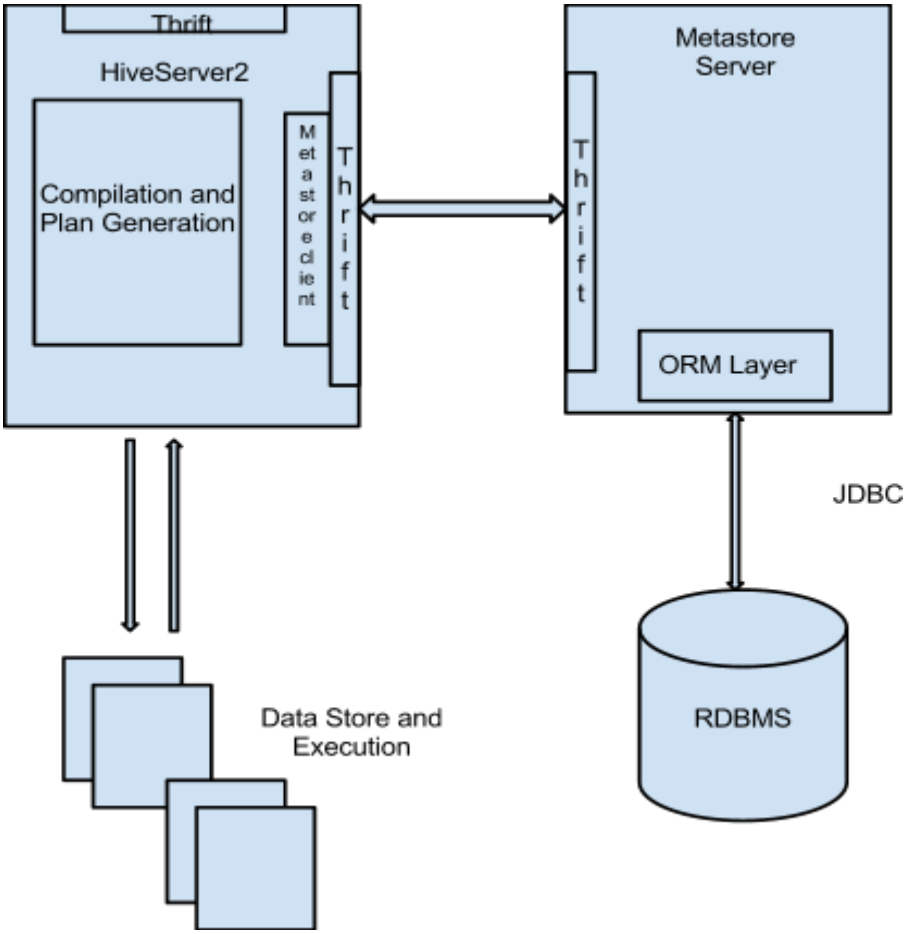
Non-Goals

- In version 1 of the project, privileges will be supported only on Hive objects and not on resources such as CONNECT, CPU etc. We will also not support rate limiting queries.
- Authorization will not be supported when the cluster is accessed through the standalone Hive CLI.
- Permissions on partitions in a table is something that has been considered and will be supported in a subsequent version of the project.
- An explicit DENY privilege will not be supported.
- Hive will not perform user account management. In a subsequent version, we can add a CONNECT privilege, if needed, to restrict users connecting to HiveServer/HiveServer2.

Technical Approach

As we talk about a fine grained RDBMS like authorization model for Hive, it is important to understand the similarities and the differences between the two systems. While Hive shares some of the characteristics of an RDBMS such as allowing data access through SQL, a rich catalog object system, etc., it also differs from an RDBMS in some ways. An RDBMS, unlike Hive, is a tightly coupled system i.e., it is not possible for users without admin privileges to access data and metadata in an RDBMS outside of the SQL engine. Hive on the other hand is a much more loosely coupled system that allows access to both data and metadata outside of

its compilation and execution engine. For instance, metastore service allows access and the ability to mutate Hive metadata outside of the HiveCLI/HiveServer. While this characteristic of Hive makes it flexible and attractive to some users, it also makes security more complex when compared to an RDBMS.



Since an RDBMS allows only SQL access to both metadata and data, the authorization model has to secure only the SQL operations. The underlying dictionary operations are encapsulated beneath a SQL interface and hence can simply leverage the authorization model for SQL operations. However, in the case of Hive, both the SQL operations and the metadata operations exposed by the metastore service have to be secured. It is also important to note that there is not a one to one mapping between SQL operations and metadata operations. Hence the privilege set that applies to the SQL operations has to be expanded and in some cases tweaked to support secure metadata operations. For instance, MySQL doesn't distinguish between privileges to read a table's data and its metadata. However, Hive has to provide fine grained metadata read only privileges to secure the table metadata read operation through the API that the metastore service exposes.

In addition to complicating the security model, supporting secure SQL and metadata operations also complicates the implementation. In order to secure the metadata API it becomes necessary to implement authorization checks within the metastore service. However, such checks are not sufficient. Checks for some SQL operations such as SELECT can't be performed exclusively in the metastore service and have to be performed in Hiveserver2.

User Visible Functionality

Hive's data model borrows heavily from MySQL and hence we have attempted to model Hive's authorization model to closely mirror that of MySQL. However, we have deviated from MySQL wherever Hive's data model has. We have made adjustments to the privilege model to allow metadata access outside of SQL operations.

Account Management Statements

We will support the following account management statements:

- GRANT
- REVOKE
- SHOW GRANTS
- CREATE ROLE
- DROP ROLE
- RENAME ROLE

GRANT

GRANT priv_type [,priv_type]
ON [object_type] priv_level
TO role_specification;

priv_type: Please refer to table 1 for valid priv_types

object_type:
TABLE
VIEW
FUNCTION

priv_level:
*
.
db_name.*
db_name.tbl_name
tbl_name
db_name.function_name
db_name.view_name
view_name

function_name
db_name

role_specification:
role

Deviation from MySQL: MySQL allows privileges to be granted to users. We will instead support privileges to be granted to roles. Role will be the most granular unit to which privileges can be granted. Note that there is a many to many mapping between users and roles.

REVOKE

REVOKE priv_type [,priv_type]
ON priv_level
FROM role_specification;

REVOKE ALL from role_specification;

role_specifications: role

SHOW GRANTS

SHOW GRANTS FOR role;

CREATE ROLE

CREATE ROLE role_name;

DELETE ROLE

DELETE ROLE role_name;

ALTER ROLE

ALTER ROLE role_name [ADD USER user_name] | [DELETE USER user_name] | [WITH NAME new_role_name];

Only superuser role can administer roles. All roles will be owned by the superuser role.

Privileges

Hive privileges will apply in different context and at different granularity of operations:

Administrative privileges: These privileges allow an admin to manage the operation of a Hive warehouse. These privileges are global because they don't apply to any one database.

Database privileges: These privileges apply to a database and to all objects within it. These privileges can be granted at the database level or globally such that they apply to all databases.

Object privileges: These privileges apply to object such as tables, indexes, views, functions within a database. These can be granted at the database level such that they apply to all objects within a database or specific objects in a database or globally for objects of a certain type within all databases.

Table 1. Valid Privilege types and objects they apply to

Privilege	Object
CREATE [DATABASE/TABLE/INDEX/VIEW/ FUNCTION]	Database, Table, Index, View, Function
DROP	Database, Table, Index, View
ALTER	Table, Index, View
ALTER DATABASE	Database
INSERT	Table
SELECT	Table, View
SHOW	Database, Table, View, Index, Routine, Lock
ALL	Shorthand for all privileges at a given privilege level
SHUTDOWN	Server
TRANSFORM	Server

Table 2. Privilege hierarchy for GRANT and REVOKE

Base Object	Granular privileges on Object	Container object that contains the base object	Privileges on container object that implies privileges on the base object
DATABASE	CREATE DATABASE	SERVER	CREATE DATABASE
SERVER	CREATE DATABASE	Not applicable	Not applicable
DATABASE	CREATE TABLE	SERVER	CREATE TABLE
SERVER	CREATE TABLE	Not applicable	Not Applicable
DATABASE	CREATE VIEW	SERVER	CREATE VIEW

SERVER	CREATE VIEW	Not applicable	Not applicable
TABLE	CREATE INDEX	DATABASE	CREATE INDEX
DATABASE	CREATE INDEX	SERVER	CREATE INDEX
SERVER	CREATE INDEX	Not applicable	Not applicable
DATABASE	CREATE FUNCTION	SERVER	CREATE FUNCTION
SERVER	CREATE FUNCTION	Not applicable	Not applicable
DATABASE	ALTER DATABASE	SERVER	ALTER DATABASE
SERVER	ALTER DATABASE	Not applicable	Not applicable
TABLE	ALTER	DATABASE	ALTER
VIEW	ALTER	DATABASE	ALTER
INDEX	ALTER	DATABASE	ALTER
DATABASE	ALTER	SERVER	ALTER
SERVER	ALTER	Not applicable	Not applicable
TABLE	SHOW	DATABASE	SHOW
DATABASE	SHOW	SERVER	SHOW
SERVER	SHOW	Not applicable	Not applicable
INDEX	SHOW	TABLE	SHOW
FUNCTION	SHOW	DATABASE	SHOW
TABLE	SELECT	DATABASE	SELECT
VIEW	SELECT	DATABASE	SELECT
DATABASE	SELECT	SERVER	SELECT
SERVER	SELECT	Not applicable	Not applicable
TABLE	INSERT	DATABASE	INSERT
DATABASE	INSERT	SERVER	INSERT
SERVER	INSERT	Not applicable	Not applicable
TABLE	DROP	DATABASE	DROP

VIEW	DROP	DATABASE	DROP
INDEX	DROP	TABLE	DROP
FUNCTION	DROP	DATABASE	DROP
DATABASE	DROP	SERVER	DROP
SERVER	DROP	Not applicable	Not applicable
TABLE	LOCK TABLE	DATABASE	LOCK TABLE
DATABASE	LOCK TABLE	SERVER	LOCK TABLE
SERVER	LOCK TABLE	Not applicable	Not applicable
SERVER	GRANT	Not applicable	Not applicable
SERVER	REVOKE	Not applicable	Not applicable
SERVER	SHOW GRANTS	Not applicable	Not applicable

1. SHOW on a table implies the user has privileges for both listing the table via a SHOW TABLES statement as well as viewing the table metadata via a DESC TABLE statement.

2. ALTER DATABASE allows a user to alter the properties of the database and not the properties of the objects such as tables, indexes etc. that a database hosts. ALTER allows a user to alter the properties of the objects such as tables, indexes etc. contained within a database. If there is no requirement for such fine grained privileges, we can have ALTER subsume ALTER DATABASE.

3. In version1 of the project, GRANT/REVOKE/SHOW GRANTS can't be explicitly granted to users. Only users with ALL privilege at SERVER level will have GRANT/REVOKE/SHOW GRANTS privilege.

4. MySQL doesn't support fine grained metadata read privileges on a table. Users who have a SELECT privilege on a table are allowed to read the table metadata. However , we will support fine grained metadata read only privileges on a table. Note that SELECT on a table subsumes the metadata read privileges but the privilege to read the metadata doesn't include the privilege to perform a SELECT on a table.Similarly INSERT , ALTER, DROP subsume the privilege to read an object's metadata. Note that this contradicts requirement # 5. However, requirement # 5 is still open and needs to be discussed.

5. We will support a generic SHOW/DROP/ALTER privilege in version 1 of the project. We will add finer grained privileges on specific objects for e.g., SHOW TABLE, if needed in a subsequent version of the project.

6. Note that while privileges can be granted on a container object, the privilege may not always apply to the container object. For instance, LOCK TABLE can be granted on SERVER, but LOCK TABLE applies only to a TABLE and not to other objects contained within the SERVER.

7. ALL on SERVER means ADMIN privileges on the server.

Privileges for SQL Operations

Hive exposes a rich set of SQL operations. The following table outlines how privileges will apply to the different operations that Hive exposes through SQL.

Hive SQL Operation	**Privileges**
EXPLAIN	SELECT
LOAD	INSERT
EXPORT	SELECT, SHOW
IMPORT	CREATE, INSERT
ANALYZE	INSERT, SELECT
CREATEDATABASE	CREATE
DROPDATABASE	DROP
SWITCHDATABASE	SHOW
DROPTABLE	DROP
DESCTABLE	SHOW
DESCFUNCTION	SHOW
MSCK	INSERT
ALTER TABLE ADD COLS	ALTER
ALTER TABLE REPLACE COLS	ALTER
ALTER TABLE RENAME COL	ALTER
ALTER TABLE RENAME PART	ALTER, DROP, CREATE
ALTER TABLE RENAME	ALTER
ALTER TABLE DROP PART	ALTER, DROP
ALTER TABLE ADD PART	ALTER, CREATE
ALTER TABLE ARCHIVE	ALTER, INSERT

ALTER TABLE UNARCHIVE	ALTER, INSERT
ALTER TABLE PROPERTIES	ALTER
ALTER TABLE SERIALIZER	ALTER
ALTER PARTITION SERIALIZER	ALTER
ALTER TABLE SERDEPROPS	ALTER
ALTER PARTITION SERDEPROPS	ALTER
ALTER TABLE CLUSTER SORT	ALTER
SHOW DATABASE	SHOW
SHOW TABLE	SHOW
SHOW COLUMNS	SHOW
SHOW TABLE STATUS	SHOW
SHOW TABLE PROPERTIES	SHOW
SHOW CREATE TABLE	SHOW
SHOW FUNCTIONS	SHOW
SHOW PARTITIONS	SHOW
SHOW INDEXES	SHOW
SHOW LOCKS	SHOW
CREATE FUNCTION	CREATE
CREATE VIEW	CREATE
CREATE INDEX	CREATE
DROP FUNCTION	DROP
DROP VIEW	DROP
DROP INDEX	DROP
ALTER INDEX REBUILD	ALTER
ALTER VIEW PROPERTIES	ALTER
GRANT PRIVILEGE	ALL on SERVER
REVOKE PRIVILEGE	ALL on SERVER

SHOW GRANTS	ALL on SERVER
ALTER TABLE PROTECT MODE	ALTER
ALTER TABLE FILE FORMAT	ALTER
ALTER TABLE LOCATION	ALL on SERVER
ALTER PARTITION PROTECT MODE	ALTER
ALTER PARTITION FILE FORMAT	ALTER
ALTER PARTITION LOCATION	ALL on SERVER
CREATE TABLE	CREATE
CREATE TABLE AS SELECT	CREATE, SELECT
QUERY	SELECT
ALTER INDEX PROP	ALTER
ALTER DATABASE	ALTER DATABASE
DESC DATABASE	SHOW
ALTER TABLE MERGE FILE	ALTER, SELECT
ALTER PARTITION MERGE FILE	ALTER, SELECT
ALTER TABLE SKEWED	ALTER
ALTER TABLE PARTITION SKEWED LOCATION	ALTER
ADD JAR	ALL on SERVER
TRANSFORM	TRANSFORM on SERVER

Privileges for Direct Metadata Operations

Unlike a traditional RDBMS, Hive allows users to perform metadata operations outside of SQL interface that Hive exposes. Hive's metastore service hosts a thrift server that exposes a metadata API that can be accessed outside of Hive's SQL interface. Users can perform metadata operations through direct Thrift calls or through APIs that wrap around the Thrift API. In order to secure the system providing a privilege based access model for these APIs becomes necessary. The following table outlines the metadata Thrift API and the privileges that will apply to each of the API calls.

Metastore API	**Privilege**
create_database	CREATE
drop_database	DROP
get_database/get_all_databases/ get_databases	SHOW
alter_database	ALTER DATABASE
create_table	CREATE
drop_table	DROP
get_table/get_all_tables/get_tables/ get_table_objects_by_name/ get_table_names_by_fi terl	SHOW
alter_table/alter_table_with_context	ALTER
add_partition/add_partitions/ add_partition_with_context	ALTER, INSERT
append_partition/append_partition_by_name	ALTER, INSERT
drop_partition/drop_partition_by_name	DROP
get_partition* (there are 6 methods)	SHOW
alter_partition/alter_partitions/ alter_partition_with_context	ALTER
rename_partition	ALTER
partition_name_to_val/ partition_names_to_spec	SHOW
add_index	CREATE
drop_index_by_name	DROP
alter_index	ALTER
get_index_by_name/get_indexes/ get_index_names	SHOW
grant_privilege/revoke_privilege/list_privilege	ALL
set_ugi	
get_delegation_token/	

renew_delegation_token/ cancel_delegation_token	
markPartitionForEvent/ IsPartitionMarkedForEvent	

Effect of Privilege Change

When a privilege change is effected, statements that are being executed will not be affected. For instance, if a user is revoked SELECT privilege on a table in the middle of a running query on that table the user will still be able to see the results of the query. This behavior is consistent with the authorization model in most traditional RDBMSs.

System Bootstrap

Most RDBMSs automatically create a user with superuser privileges as part of the database creation. Similarly, we will allow the creation of a dba role and association of user ids with that role as part of the system bootstrap. Since Hive doesn't support the concept of creating a database, we plan to achieve this by specifying the dba role and the associated user ids in hive-site.xml.

Q&A

1. Should those alter operations eg., alter index rebuild that transform data and don't mutate metadata require ALTER privileges?
Answer: For the sake of remaining consistent with the other alter operations, alter index will require ALTER privilege.

2. Should we provide privileges for TYPES? Is this feature even used?
Answer: It looks like TYPES is not used end to end (While there are metastore tests for the metastore Thrift APIs, there are no q file tests). Hence privileges will not be supported for TYPES.

References

1. http://dev.mysql.com/doc/refman/5.1/en/privileges-provided.html
2. https://dev.mysql.com/doc/refman/5.6/en/privilege-system.html
3. http://dev.mysql.com/doc/refman/5.6/en/account-management-sql.html