As the metastore moves from Hive only to a shared resource we need a clean way to share it between multiple systems. We propose to add the catalog concept to the metastore and allow different systems to connect to different catalogs in the metastore.

Use Cases:

Use Case 1: Multiple Systems Share a Metastore

In this case multiple systems (e.g., Hive and Spark) share a single metastore instance. By default they cannot see each other's data. Each system may want a different security model as well (e.g. Hive might use Ranger, while a Spark based system might use Storage Based Auth).

Eventually in this use case we would like to be able to join data across catalogs (e.g., a Hive query might be able to read data from a Spark table).

A small extension of this use case is that multiple Hive users may wish to share a metastore but not allow access to each other's data. This is more likely in the cloud, where all data is, in some sense, in one cluster.

Use Case 2: External Systems Accessible From Hive

Hive can currently connect to external systems such as Druid, HBase, etc. via its StorageHandler interface. Being able to represent a full external system as a separate catalog will enable Hive to "auto" import from these external systems without polluting its own namespace. Eventually we may be able to use this to connect to other Hive metastores as well.

High Level Requirements:

1. A system administrator must be able to create catalogs. At creation time the administrator will also choose a security model for the catalog (Ranger, Sentry, SQL Standard, or Storage Based).
2. A system administrator must be able to drop empty catalogs.
3. All users have access to every catalog. Inside the catalog the chosen security model will determine what (if anything) the user can access.
4. Users must be able to list catalogs.
5. Thrift calls will be added to enable HiveServer2 to ask the metastore which security model to use with a catalog.
6. A default catalog for operations will be defined in the configuration file for HiveServer2 and command line clients. The metastore itself will not assume a default catalog (except for backwards compatibility, see below).
7. A user must be able to specify which catalog to connect to. This may be by SQL command or element of the JDBC connection string. A user must be able to switch catalogs during a session.
8. The system must continue to work for all clients with no catalog specified. This is for backwards compatibility. A default catalog named "hive" will be assumed in this case. When no catalog is specified at connection time it will be assumed that the user is connecting to the catalog configured in the system's configuration file. This includes not breaking old thrift clients that do not know about catalogs.
9. It is not yet determined what types of cross catalog queries and DML statements will be supported. The initial version(s) should not do anything that prevents cross catalog queries or DML statements in the future.
10. The 'hive' catalog will be created at system install similar to the way the 'default' database is created today.

Possible Future Use for Caching:
We need a way in the metastore to limit the number of objects and transactions to a size that can be managed by a single server so that we can effectively cache data, transactions, and locks in the metastore. However, for obvious reasons, we do not want to limit the metastore to running on a single server. Catalogs may offer a natural place to define caching that scopes size of objects and transactions reasonably while not limiting the overall size of the metastore.

Technical Approach
Changes to the metastore database:
A new table CTLGS will be added to record the catalog.

A new column will be added to the DBS table, CTLG_NAME. This will contain the name of the catalog the database is in. This will be the string name not the id from the CTGLS table. This is done to prevent the need to join against the CTLGS table every time a database, table, or partition is looked up. A foreign key will be placed on this column that references the NAME column in the CTLGS table. This has the downside that changing a catalog name is very difficult. In the short term there are no plans to enable renaming catalogs. Even if we want to allow that in the future it will not be a common event we should optimize for.

For backwards compatibility part of the upgrade script will include creating a new 'hive' catalog and updating every existing database to be a part of that catalog.

At system startup in a newly installed system, HiveMetaStore will create the 'hive' catalog in the same code that it currently creates the 'default' database (which will now be placed in the 'hive' catalog).

Changes to Thrift API:
A new Catalog Thrift object will be defined. It will have a name, a description, and a location.

All of the Thrift calls that give database name as an argument (for example, getTable which takes database name and table name) will have to be changed to also accept catalog name. Unfortunately, Thrift does not support adding arguments to service calls. To work around this new versions of each of these methods will be added. These new versions will take a request struct so that in the future arguments can be added without adding new versions of the method. For backwards compatibility the old Thrift calls will be maintained. Internally they will be redirected to the new versions with the catalog set to 'hive'.

For example, currently there is a method:
```
Table get_table(1: string dbname, 2:string tbl_name)
```

This will be changed to:
```
struct GetTableRequest {
  1: string catName;
  2: string dbName;
  3: string tableName;
}
// Deprecated, for backwards compatibility
Table get_table(1: string dbname, 2:string tbl_name)
```

```
Table get_table2(1: GetTableRequest rqst)
```