Feature 1: Add a new table

Our intent was to recognize where Cassandra allowed users to add a new table to any designated schema. First we began by searching the source code for any instance of "Add Table" and were directed to the `addSSTable` method in the Columnfamilystore file. This is the first time we felt confused as ColumnFamilyStore sounds unfamiliar for us and we did not understand why a table is connected to a store. This method called another method named `addSSTables`, with a plural s. Now we found that Cassandra allows to add more than one SSTable at a time. The `addSSTable` will make a singletonList of the one input SSTable to match the input requirements of `addSSTables`.

```
public void addSSTable(SSTableReader sstable)
{
    assert sstable.getColumnFamilyName().equals(name);
    addSSTables(Collections.singletonList(sstable));
}

public void addSSTables(Collection<SSTableReader> sstables)
{
    data.addSSTables(sstables);
    CompactionManager.instance.submitBackground(this);
}
```

(In the comments, it says this method is called after a BinaryMemtable flushes its in-memory data. This information is cached in the ColumnFamilyStore. We want to find more about Memtable and ColumnFamilyStore later.)

In `addSSTables` there is `data.addSSTables`. So we followed this track to the implementation of `addSSTables` in the lifecycle/tracker.java file in the same db folder. This method called three separate methods which we followed individually. `addInitialSSTable` looks great. But `maybeIncrementalBackup` and `notifyAdded` seems like irrelevant to the adding. We go to the first method called.

```
public void addInitialSSTables(Iterable<SSTableReader> sstables)
{
    if (!isDummy())
        setupOnline(sstables);
    apply(updateLiveSet(emptySet(), sstables));
    maybeFail(updateSizeTracking(emptySet(), sstables, null));
    // no notifications or backup necessary
}
```

The `addInitialSSTable` method had some integrated code regarding setting up and updating, which seems quite different from what we expected. At this point the team hit a dead end and reconvened at the top to find a better entry point to recognizing the "Add Table" feature.

We checked the document and other online sources, and learned more about Cassandra first. Cassandra's data model is a four-dimensional or five-dimensional model based on the column family. (This explains our confuse about ColumnFamilyStore.) It uses the methods of memtable and sstable for storage. Before Cassandra writes data, it is necessary to record the commitlog first, and then write the data to the memtable corresponding to the column family. Memtable is a

memory structure that sorts the data according to the key. When certain conditions are met, the data of memtable will be updated to the disk in batches and stored as sstable.

It seems that our impression of "add a table" in Cassandra is adding to memtable. We tried to search "add memtable" but found no related classes/methods. So we went to Memtable itself and hoped we could find something. This class is huge with plenty of parameters and none of them looked relevant to our goal. We kept looking for other classed named with Memtable but failed.

(After two days...)

We found a `Table` class when we were working on the second feature. So we went back here after that.

```
public Builder add(TableMetadata table)
{
    tables.put(table.name, table);

    tablesById.put(table.id, table);

    table.indexes
         .stream()
         .filter(i -> !i.isCustom())
         .map(i -> CassandraIndex.indexCfsMetadata(table, i))
         .forEach(i -> indexTables.put(i.indexName().get(), i));

    return this;
}
```

This looked much better. We give the table a name, an ID and other info as Metadata.

Folder	File	Method	Why	Priority	Notes
db	ColumnFamilyStore	addSSTable	looks right	high	why column family store?
db	ColumnFamilyStore	addSSTables	called in previous method	high	difference?
	Memtable related?		appeared	low	just for better understanding
	ColumnFamilyStore?		appeared	low	may not be useful
db/lifecycle	Tracker	addSSTables	called in previous method	high	same method name again?
		addInitialSSTables		high	
		maybeIncrementallyBackup		low	
		notifyAdded		low	
db	Memtable	/			

Folder	File	Method	Relevant	Relevant how	confidence	Notes
db	ColumnFamilyStore	addSSTable	yes	initial start	med	why column family store?
db	ColumnFamilyStore	addSSTables	yes	called in previous	med	difference?
db/lifecycle	Tracker	addSSTables	yes	called in previous	med	keep calling others
db/lifecycle	Tracker	addInitialSSTables	yes	called in previous	med	too many similar method names...
schema	Tables	Builder.build	yes	called	high	life is interesting

Feature 2: Notify user when a table is added

We first locate at a folder named notifications assuming that this feature will be implemented there. Then in the folder we found a java class named `SSTableAddedNotification` that seems like the right track. From the comment at the heading of this class (attached below), we are confirming this java class is a good start to look for this feature.

```
/**
 * Notification sent after SSTables are added to their {@link
org.apache.cassandra.db.ColumnFamilyStore}.
 */
```

When look at `SSTableAddedNotification` constructor inside this class, we found that only two values are assigned, a `MemTable` and an iterable of `SSTableReader`. From the Cassadra official document, we learned that an SSTable were flushed into a disk from a MemTable and Memtables are in-memory structures where Cassandra buffers writes. In general, there is one active Memtable per table. Eventually, memtables are flushed onto disk and become immutable SSTables.

```
public SSTableAddedNotification(Iterable<SSTableReader> added, @Nullable
Memtable memtable)
    {
        this.added = added;
        this.memtable = memtable;
    }
```

We scanned through the whole class but could not find anything similar to a notification function. We referred back to the official document and figured that a MemTable will flush the ssTable data to disk when the memory usage of memtable exceeds the configured threshold. Or it will flush when CommitLog approaches its maximum size. We tried to locate any relevant method inside memtable class, there was one that looks similar to it with a flush function but doesn't do anything that looks like a notification but merely flush data when size limit is reached.

```
public Memtable(AtomicReference<CommitLogPosition> commitLogLowerBound,
ColumnFamilyStore cfs)
    {
        this.cfs = cfs;
        this.commitLogLowerBound = commitLogLowerBound;
        this.allocator = MEMORY_POOL.newAllocator();
        this.initialComparator = cfs.metadata().comparator;
        this.cfs.scheduleFlush();
        this.columnsCollector = new
ColumnsCollector(cfs.metadata().regularAndStaticColumns());
    }
```

We decided to look for another trace: SSTableReader, after reading 2000+ lines of code in this class and found nothing that looks right, we move on by using the search function from IntelliJ and queries include: "table added", "added table", and finally locate at another file `Table` from schema folder. Two functions caught our eyes, `with` and `without`. Both include a string that tells if a table already exist or does not exist in the database. However, this method only changes if the table exist but does not tell whether a table is successfully added.

(This looked like what we needed in the first feature!)

```
public Tables with(TableMetadata table)
{
    if (get(table.name).isPresent())
        throw new IllegalStateException(String.format("Table %s already
exists", table.name));

    return builder().add(this).add(table).build();
}
```

```
public Tables without(String name)
{
    TableMetadata table =
        get(name).orElseThrow(() -> new
IllegalStateException(String.format("Table %s doesn't exists", name)));

    return without(table);
}
```

We keep searching different keywords such as "added", and within 100+ search result, we locate another seemly right method called `notifyAdded` inside IO/LifeCycle.

```
Throwable notifyAdded(Iterable<SSTableReader> added, Memtable memtable,
Throwable accumulate)
{
    INotification notification = new SSTableAddedNotification(added,
memtable);
    for (INotificationConsumer subscriber : subscribers)
    {
        try
        {
            subscriber.handleNotification(notification, this);
        }
        catch (Throwable t)
        {
```

```
            accumulate = merge(accumulate, t);
        }
    }
    return accumulate;
}
```

The method creates a SSTableAddedNotification class at the beginning, and this object is called for each subscriber to handle the notification. It seems like the this is kind of a java observer and observable model that whenever the observable(database, table, etc) is changed, every subscriber that is bound to this obseravable will be notified for any changes. The next thing we look at is `handleNotification` method in side SecondaryIndexManager folder. In this method, buiding index will create added notification message in the buildIndexBlocking.

```
public void handleNotification(INotification notification, Object sender)
{
    if (!indexes.isEmpty() && notification instanceof
SSTableAddedNotification)
    {
        SSTableAddedNotification notice = (SSTableAddedNotification)
notification;

        // SSTables asociated to a memtable come from a flush, so their
contents have already been indexed
        if (!notice.memtable().isPresent())
            buildIndexesBlocking(Lists.newArrayList(notice.added),
                                 indexes.values()
                                 .stream()
                                 .filter(Index::shouldBuildBlocking)
                                 .collect(Collectors.toSet()),
                                 false);
    }
}
```

Folder	File	Method	Why	Priority	Notes
Notifications	SSTable AddedNotification	SSTable AddedNotification	looks right	High	not too much function is in there
db	MemTable	MemTable	has flush function	medium	not the right track
io	SSTableReader	None	Iterable has this class type	Medium	
schema	Table	With, without		Medium	
IO/Lifecycle	Tracker	NotifyAdded	name looks right	high	
Index	Secondary IndexManager	handleNotification	observable method	High	handling all update of database

A little summary

This is not a great experience for us as we spent so much time walking through the GIANT folder. One of the reasons why we failed for the first feature at the beginning was that we had no experience with the source code of real industry products. We thought that adding a table was quite easy but in reality, it was a series of steps and a lot of concerns when it came to distributed systems, network, scalability and reliability. This gave us a lesson.