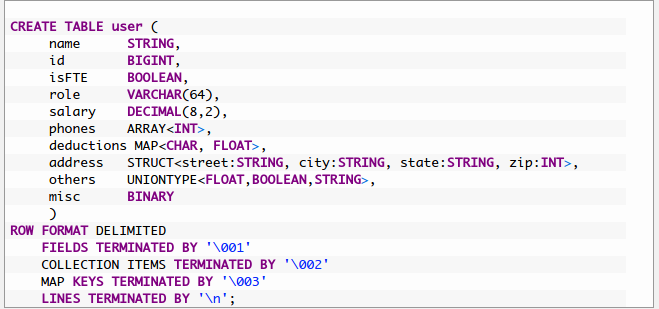
## Hive Data modeling

Hive data modeling differs from RDBMS data modeling in several terms due to its immutable nature( HDFS ) and lack of transaction support. Below are some of the key points to be considerd:

* you will see that the Hadoop schema consolidates many of the small dimension tables into a few larger dimensions by joining them during the ETL process
* Consider complex types in Hive. They let you store data together for a row and avoid duplicating it and at the same time by not creating normalized tables, you avoid potentially expensive joins.

**Nested data types** like struck, map and array. This is a good middle ground between normalization and denormalization



e.g. consumer indv. Storing the multiple records for storing various contact number for the member, instead we can leverage array data type to store multiple consumer contact numbers into the single row one field.

* Create data sets that are pre-joined—in other words, preaggregated. The idea is to minimize the amount of work queries have to do by doing as much as possible in advance.Since data duplication is a lesser concern, almost any type of processing that occurs frequently in a large number of queries is worth doing once and reusing.
* To facilitate bucket joins ensure frequently joining datasets follow the  number of buckets of one data set is a multiple of the other.
* If the data in the buckets is sorted, it is also possible to use a merge join and not store

the entire bucket in memory when joining.

Tools like Pig / Spark can be used for sorting the data while loading.

* Having the number of buckets as a power of two is quite common.
* In cases when there are multiple common query patterns and it is challenging to decide on one partitioning key, you have the option of storing the same data set multiple times, each with different physical organization. This is considered an anti-pattern in relational databases, but with Hadoop, this solution can make sense
* You will need to know which columns will be used for joining and filtering before deciding on partitioning and bucketing of the data.
* When underlying storage format is not ORC, then you can consider building indexes on the table

Types of Indexes in Hive support are

* Compact Indexing
* Bitmap Indexing
* Once table is loaded, we should run statistics on the table to help CBO ( cost based optimizer )

CBO requires both column level / Table level statistics to be gathered

ANALYZE TABLE <table\_name> COMPUTE STATISTICS; Column-level statistics (critical):

* Avoid too-many partitions on the table which can detoriate the performance
* Per table there is limit for 2000 dynamic partitions on the table.
* If possible try to leverage ORC/Parquet as Storage format for the table to achieve better performance.