JethroData Quick Start Guide

Last Update: 13 July 2015

Thank you for downloading JethroData! The following guide will take you through the steps of having the latest version of JethroData installed and functioning. More information regarding the full functionality of JethroData can be found in our website.

1 JethroData Highlights and High-Level Architecture

JethroData is an innovative index-based SQL engine that enables interactive BI on Big Data. It fully indexes select datasets on Hadoop HDFS or Amazon S3 – every single column is indexed. Queries use the indexes to access only the data they need instead of performing a full scan, leading to a much faster response time and lower system resources utilization.

1.1 JethroData Highlights

Easy To Get Started: Just install it on one or few dedicated servers, point it to your Hadoop cluster or Amazon S3 bucket, load some data and start querying, .When working with Hadoop –JethroData is safe and easy to implement as it only connects remotely as an HDFS client, without installing new services or running resource-intensive computations inside the cluster (MapReduce, Spark or others).

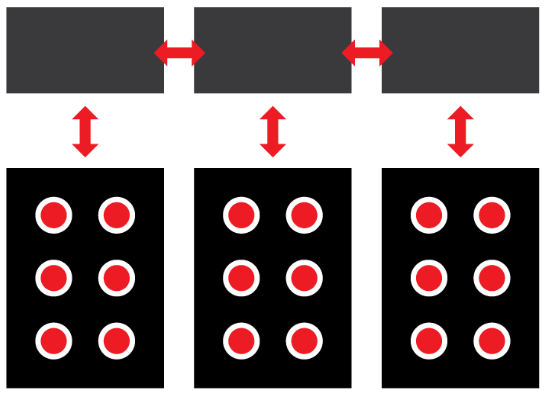
Performance From a Unique Indexing Technology: The key to JethroData's superior performance is its unique indexing technology. JethroData's indexes are sorted, multi-hierarchy, compressed bitmaps. They are created automatically for every column, and are written in an efficient, append-only fashion – avoiding expensive random writes and locking. Queries use indexes to read only the data they need, instead of performing full scans, leading to faster response time.

Scalability and High-Availability: JethroData's nodes are stateless and highly elastic, allowing easy scale-out to meet concurrency requirements. JethroData's index and column files are stored as standard files on HDFS or Amazon S3 and benefit from their native scalability and high availability.

Minimal Hadoop Cluster Load: Other SQL-on-Hadoop solutions uses a brute force method – each node of the cluster scans and processes its local data for every query. In contrast, JethroData leverages its indexes to surgically fetch only the relevant data for each query, dramatically reducing the load on the shared Hadoop cluster – freeing it for other computations and supporting more concurrent queries.

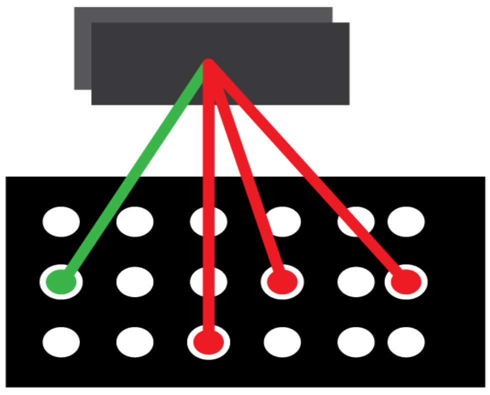
Hadoop Full Scan / Brute Force vs. **Index Access**

(all SQL-on-Hadoop Solutions)
1. Read entire dataset. Every time.



Impact on Hadoop Cluster
Massive number of unnecessary I/O
Extensive cluster I/O, CPU and memory usage

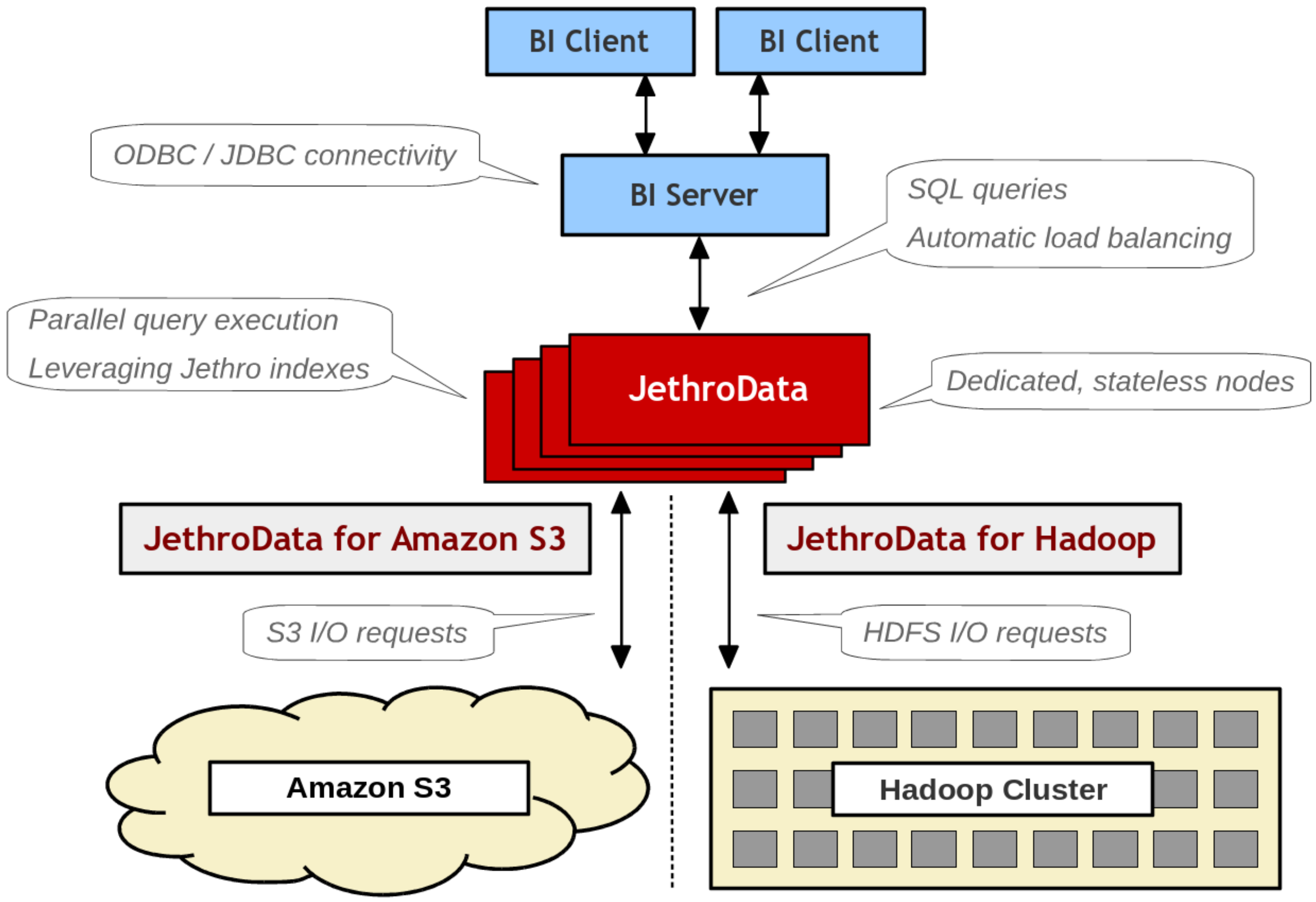
(JethroData)
1. Analyze indexes
2. Fetch only relevant data



Impact on Hadoop Cluster
Drastically lower cluster load
Minimal cluster I/O, CPU and memory usage

1.2 High-Level Architecture

JethroData's architecture is optimized to deliver great performance using indexes.



SQL Interface: BI tools connect to JethroData using JDBC or ODBC and issue standard SQL queries. The driver automatically load-balances SQL statement across all JethroData hosts.

Query Processing: JethroData runs on one or few dedicated, higher-end hosts optimized for SQL processing – with extra memory and CPU cores, and local SSD for caching. The query hosts are stateless, and new ones can be dynamically added to support additional concurrent users.

Storage Layer: JethroData stores its files (e.g. indexes) in an existing Hadoop cluster or in an Amazon S3 bucket. With Hadoop, it uses a standard HDFS client (*libhdfs*) and is compatible with all common Hadoop distributions. JethroData only generates a light I/O load on HDFS – offloading SQL processing from Hadoop and enabling sharing the cluster between online users and batch processing.

Data Loading and Indexes: a loader service processes input files and creates query-optimized column and index files, which are encoded, compressed and then stored on HDFS or Amazon S3. This service can run on its own host or on one of the query processing hosts.

2 Installing JethroData on a Server

2.1 Requirements

The following are the Hadoop cluster requirements, and the minimum and recommended hardware and software requirement for a JethroData host:

Hadoop Cluster Requirements

JethroData is designed to work with all modern Hadoop 2.x distributions. JethroData only needs access to HDFS – it does not run MapReduce / Tez / Spark jobs on the cluster. It supports both unsecured and secured clusters (using Kerberos). Also, as JethroData works on its own dedicated nodes, no JethroData software is installed on the Hadoop cluster.

The current release is certified on Cloudera CDH 4.x and 5.x, Hortonworks HDP 2.x, Amazon EMR 3.x and MapR 4.0.x. MapR support is delivered by a dedicated installer (a separate RPM).

JethroData Host – Cloud Instance

If you install JethroData on a cloud instance, use a memory-optimized instance with at least 8 vCPUs, about 64GB of RAM and local SSD (recommended is twice or more the CPUs and RAM).

On **Amazon AWS**, minimal instance type is r3.2xlarge, recommended **r3.4xlarge / r3.8xlarge**.
On **Microsoft Azure**, minimal instance type is D13, recommended **D14**.

In addition, if using **Amazon Elastic MapReduce (EMR)** as the Hadoop distribution, it is best to allocate the JethroData hosts as part of the Master group. This makes sure that the EMR Hadoop client software is pre-installed and pre-configured.

JethroData Host – Physical Hardware

If you use a physical server for JethroData software, follow these recommendations:

CPU – use at least a modern 2-socket server (ex: 2 x 8-core Intel server).
Memory – minimum 64GB of RAM, recommended 128GB/256GB (or higher).
Disk – At least 5GB free in both `/opt` and `/var`. (indexes are stored on HDFS, not locally).
In addition, allocate storage for local caching. It is recommended to provide a dedicated SSD drive (few hundreds gigabytes or more) on its own mount point.

Network Bandwidth – to Hadoop cluster, at least 1Gb/s link, recommended 10Gb/s link.
To SQL clients (BI tools etc) – recommended 1Gb/s link.

JethroData Host – Location

JethroData software should be installed on host located next to the Hadoop cluster.
This means in the same data center for on-premise installations and same region and availability zone for AWS installation (or the equivalent in other cloud environments).

If you re-use an existing Hadoop DataNode for JethroData, decommission it from the cluster first to avoid JethroData's or Hadoop's processes from being killed under occasional memory pressure.

JethroData Host – Operating System

JethroData is certified on 64-bit RHEL/CentOS 6.x and on Amazon Linux (when using Amazon AWS).

2.2 JethroData Host Setup

Basic Setup

- Install **Java** (for the Hadoop client), if not installed.
  It is recommended to use the same version that is installed in the Hadoop cluster.
- **Local storage** - verify there is at least 5GB available under both `/opt` and `/var/log`, where JethroData stores binaries and logs.
- **Local caching** – prepare a local directory on a dedicated SSD. JethroData will use it to cache files. As an administrator, you need to set up the directory, including auto-mounting it (if using a separate mount point). For example, as `root`:

```
mkdir /mnt/jethro_local_cache
-- make a file system on the local SSD and auto-mount it (not shown)
chmod 700 /mnt/jethro_local_cache
```

If you can not set up a dedicated SSD for caching, create a dedicated directory on your internal disk (less recommended).

- **Kerberos Client** (optional) – if access to HDFS is secured, install and configure a Kerberos client.
  Installation – `yum install krb5-workstation`
  Configuration – copy `/etc/krb5.conf` from any DataNode to the same location locally.

Install Hadoop Client

In order for JethroData to connect to the Hadoop cluster, a distribution-specific set of Hadoop Client libraries needs to be installed and configured on the JethroData host.

Skip this step and the next one if the JethroData host already has Hadoop libraries installed and configured. For example, if you have decommissioned a Hadoop DataNode to be used as JethroData host, it already has the correct version of your Hadoop binaries and libraries installed and configured.

Cloudera CDH 4.x – run the following as `root`:

```
rpm --quiet --import
http://archive.cloudera.com/cdh4/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
wget http://archive.cloudera.com/cdh4/one-click-
install/redhat/6/x86_64/cloudera-cdh-4-0.x86_64.rpm
yum -y localinstall cloudera-cdh-4-0.x86_64.rpm
yum -y install hadoop-client
```

Cloudera CDH 5.x – run the following as `root`:

```
rpm --quiet --import
http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
wget http://archive.cloudera.com/cdh5/one-click-
install/redhat/6/x86_64/cloudera-cdh-5-0.x86_64.rpm
yum -y localinstall cloudera-cdh-5-0.x86_64.rpm
yum -y install hadoop-client
```

Hortonworks HDP 2.1 – run the following as `root`:

```
wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/GA/2.1-latest/hdp.repo -O /etc/yum.repos.d/hdp.repo

yum -y install hadoop-client
```

Hortonworks HDP 2.2 – run the following as `root`:

```
wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/GA/2.2.0.0/hdp.repo -O /etc/yum.repos.d/hdp.repo

yum -y install hadoop-client
```

MapR 4.x (all editions) - run the following as `root`:

Instructions are based on "Install and Use the MapR Client" on MapR website

- Download the latest 64-bit MapR Client software from http://package.mapr.com/ , for example:
  `wget http://package.mapr.com/releases/v4.0.2/redhat/mapr-client-4.0.2.29870.GA-1.x86_64.rpm`

For other options, see "Preparing Packages and Repositories" on MapR website.

- Remove any existing MapR software, and install the `mapr-client.x86_64` software you downloaded in the previous step, for example:
  ```
  rpm -e mapr-fileserver mapr-core mapr-client
  yum install mapr-client-4.0.1.27334.GA-1.x86_64.rpm
  ```

Amazon EMR (all versions)

In order to have the appropriate Hadoop libraries installed, it is recommended to create the JethroData hosts(s) as Master hosts in EMR. This will make sure the JethroData host has the right Hadoop software and that the Hadoop configuration files points to the Hadoop cluster.

2.3 Hadoop Cluster Setup

- **Create `jethro` O/S User for the Hadoop Cluster**
  (if you can not use `jethro` as the Hadoop user name, contact us for a workaround).
  - **Non-Kerberos** - create a Hadoop O/S user and group called `jethro` on the NameNode, and on any node that may serve as NameNode (like a Standby NameNode, if configured).
    If `jethro` O/S user and group does not exist on the NameNode, each access from JethroData to the NameNode will generate a warning and a stack trace in the NameNode logs, flooding its log and significantly impacting performance. See HADOOP-10755 for details.
  - **Kerberos** – create a Kerberos principal called `jethro` and generate a keytab file for it.
    For example, if using MIT KDC:
    ```
    kadmin.local -q "addprinc -randkey jethro"
    kadmin.local -q "ktadd -k jethro.hadoop.keytab jethro"
    ```
    (Later on you will securely copy the keytab file to the JethroData host and make it owned by the `jethro` O/S user. It will be used to run `kinit` command).
- **Create Directories on HDFS**
  As Hadoop `hdfs` user, create a root HDFS directory for Jethro files, owned by `jethro` Hadoop user. In this document we assume it is `/user/jethro/instances`:

```
hadoop fs -mkdir /user/jethro
```

```
hadoop fs -mkdir /user/jethro/instances
hadoop fs -chmod -R 740    /user/jethro
hadoop fs -chown -R jethro /user/jethro
```

- **Verify Network Access**– verify that there is access from the JethroData host to the NameNode and all DataNodes – it might require changing firewall rules to open ports.

Configure Hadoop Client

If you installed a new Hadoop Client, you need to configure it to point to your Hadoop cluster.

For CDH 4.x / 5.x and HDP 2.x distributions

Copy `core-site.xml` and `hdfs-site.xml` from any DataNode of your Hadoop cluster to the JethroData host, to the `/etc/hadoop/conf/` directory.

For MapR distribution

Configure the MapR Hadoop Client by running `configure.sh.` Use the `-c` option for client configuration and also `-C` option, listing all the CLDB nodes:

```
/opt/mapr/server/configure.sh -c -C mynode01:7222
```

If using Kerberos, add `-secure` option after `-c` (see "configure.sh" on MapR website for details).

2.4 Installing JethroData Software

JethroData installation runs as `root`. It extracts JethroData software to `/opt/jethro` and automatically creates and configures the `jethro` O/S user and Linux service.

- Connect as `root` and download the JethroData RPM to a local directory:
```
wget http://www.jethrodata.com/latest-rpm
```
- Install the RPM you downloaded with the following command (use the name of your version):
```
rpm -Uvh {downloaded_file.rpm}
```
- Change the owner of the JethroData cache directory to the `jethro` user (the user was created during the RPM installation):
```
chown jethro:jethro /mnt/jethro_local_cache
```
- If using Kerberos, securely copy the keytab file that was generated in section 3.2 to the `jethro` home directory and change its permissions accordingly:
```
scp kdc_owner@kdc_server:jethro.hadoop.keytab /home/jethro
chown jethro:jethro /home/jethro/jethro.hadoop.keytab
chmod 600 /home/jethro/jethro.hadoop.keytab
```
- Optionally, set the password for the `jethro` O/S user by running `passwd jethro`.

The rest of the setup is done as `jethro` O/S user.

```
su – jethro
```

2.5 Test HDFS Connectivity As `jethro` User

- If using Kerberos, run `kinit` to enable HDFS access for your terminal session:
  `kinit -k -t ~/jethro.hadoop.keytab jethro`
- Verify write permissions as `jethro` O/S user on the HDFS directory.
  Use the following commands to create and delete a new file from the HDFS directory:

```
hadoop fs -ls                /user/jethro/instances/testfile
hadoop fs -touchz            /user/jethro/instances/testfile
hadoop fs -ls                /user/jethro/instances/testfile
hadoop fs -rm -skipTrash /user/jethro/instances/testfile
hadoop fs -ls                /user/jethro/instances/testfile
```

Make sure that the `testfile` was successfully created and then deleted.

2.6 Creating A JethroData Instance

A JethroData instance includes an entire database that is stored on HDFS – all the tables, data, indexes and metadata.

To create a new, empty instance, run the `JethroAdmin` **`create-instance`** command and provide the following parameters:

- An instance name of your choice (`demo` in the following example).
- An HDFS storage path – the HDFS directory owned by user `jethro` that you previously created.
- Local caching parameters – the local path and maximum size of the JethroData cache directory that you previously created (step 1).

For example:

```
JethroAdmin create-instance demo -storage-path=/user/jethro/instances
-cache-path=/mnt/jethro_local_cache -cache-size=80G
```

If the Hadoop cluster is secured, follow the additional required Kerberos tickets setup.

Once the instance is created, start it by running:

```
service jethro start
```

2.7 Tips For Working From Shell

- **$JETHRO_HOME** – this environment variable (pre-defined for `jethro` O/S user) points to the JethroData home directory (`/opt/jethro/current`).
- **BASH Auto-Complete** – the `jethro` O/S user has automatic auto-complete of command options in the shell. This includes auto-complete for instance name (for example, when running `JethroClient` or `JethroAdmin`) and also JethroAdmin options. Just press tab twice when typing command options. For example: `JethroAdmin c`, press tab twice and the option will automatically be completed to `create-instance`.

2.8 Local Directory Structure

JethroData software is installed under `/opt/jethro`.

The directory structure supports multiple installed binary versions – for example, for easier upgrade / downgrade. It also supports multiple instances – for example, running several dev and test instances from the same host.

`/opt/jethro` – the top JethroData directory
`/opt/jethro/jethro-`*`version`* – one directory per installed version of JethroData
`/opt/jethro/current` – a symbolic link to the current used version of JethroData
`/opt/jethro/instances` – metadata for all instances
`/opt/jethro/instances/{`*`instance_name`*`}` – metadata for a specific instance
`/var/log/jethro` – top directory for JethroData logs
`/var/log/jethro/`*`instance_name`* – log directory for a specific instance

3 Connecting to JethroServer

JethroData software includes three types of clients – an ODBC driver, a JDBC driver and a Linux CLI client (`JethroClient`).

3.1 JethroClient

JethroClient is a SQL command-line interface (CLI) for JethroServer. Run it with:
`JethroClient` *`instance_name connect_string`* ~~[~~`-p password` ~~]~~`[options]`

Connect string

The connect string is a list of one or more `JethroServer` network addresses, specified as host:port pairs. When specifying multiple `JethroServer` network addresses, they need to be separated by semi-colon (;) and the entire connect string needs to be quoted. Specifying multiple addresses automatically enables client-side load balancing. For example:

```
JethroClient demo localhost:9111 -p jethro
```

Or if connecting to more than one host:

```
JethroClient demo "10.0.0.77:9111;10.0.0.88:9111" -p jethro
```

3.2 JDBC Driver

Jethro's JDBC driver allows BI tools and other applications to connect to the JethroServer using the JDBC API.

The JDBC driver is included in the JethroData RPM and can be found under `$JETHRO_HOME/jdbc` and at the download page on JethroData web site. It consists of two JAR files (`JethroDataJDBC.jar` and `protobuf-java-2.4.1.jar`) that must both be available to the JDBC client – typically, copied to the client machine and included in the Java application CLASSPATH.

Once the driver is in the CLASSPATH, you need to provide the following parameters in order to connect to the JethroServer:

- ***class***: `com.jethrodata.JethroDriver`
- ***url***: `jdbc:JethroData://`*`host:port[;host:port].../instance_name`*
  The URL includes a connect string – a list of one or more `JethroServer` network addresses. Specifying multiple addresses turns on client-side load balancing.

URL examples:

```
jdbc:JethroData://localhost:9111/demo
jdbc:JethroData://10.0.0.77:9111;10.0.0.88:9111/demo
```

In addition, you will need to provide a user name and a password (default is `jethro/jethro`)

3.3 ODBC Driver

We currently provide an ODBC driver for Windows 64-bit on JethroData website, with its own documentation for installation and for setting access to JethroData from various BI tools.

3.4 Desktop SQL GUI

Any general-purpose SQL GUI tool should work with JethroData, for example SQL Workbench:

1. Download the appropriate SQL Workbench for your OS and client.
2. Follow the instructions on the SQL Workbench site for installation.
3. Configure the JethroData driver by adding a new one in the 'File -> Manage Drivers' window, as follows:

   Name: JethroData

   Library: Copy the `JethroDataJDBC.jar` and `protobuf-java-2.4.1.jar` to a local directory on your computer (can be found at `$JETHRO_HOME/jdbc` or on our website), and add them here.

   Classname: com.jethrodata.JethroDriver
4. Click on File -> Connect window and set the following:

   Driver: JethroData

   URL: `jdbc:JethroData://`*`host:port[;host:port].../instance_name`* (as described in section 3.2)

   Username/Password: default is `jethro/Jethro`
5. You can name the connection and click on the 'save' button for further use.
6. Click on OK to connect.

4 Creating DB and SQL Queries

The JethroData loader utility `JethroLoader` allows a fast and efficient data loading from text files into a JethroData table. It parses its input into rows and loads them into an existing JethroData table.

In production environments where large data loads are running during user activity, we recommend to run the `JethroLoader` from a separate host, to minimize the impact on running queries.

This chapter will use the following distinction between an input file and a table:

- The input file is made of **records** (usually, each line is a separate record), each record is broken into one or more **fields**.
- A JethroData table is made of **rows,** each row has one or more **columns**.

4.1 Create Table

To create a regular table, run a **CREATE TABLE** statement and provide the table name and list of columns. For example:

```
CREATE TABLE my_table (a INT, b STRING, c TIMESTAMP);
```

The supported **data types** are:

```
INT | BIGINT | FLOAT | DOUBLE | STRING | TIMESTAMP
```

JethroData indexes all columns automatically – there is no need to run CREATE INDEX commands.

JethroData also supports work with views and partitioned tables. Check the reference guide for more details, as well as information on each data type and more table functions such as renaming and dropping tables and columns drop columns

The CREATE TABLE script can be also loaded from a file, using the command:

```
JethroClient {instance_name} {instance_address:port} -p {user} -i {scriptfile}
```

For example, the CREATE TABLE for a table called sales_demo:

```
CREATE TABLE sales_demo
(
customer_sk                 INTEGER,
customer_salutation         STRING,
customer_first_name         STRING,
customer_last_name          STRING,
customer_preferred_flag     STRING,
customer_birth_year         INTEGER,
customer_birth_country      STRING,
customer_gender             STRING,
customer_marital_status     STRING,
customer_education_status   STRING,
store_sk                    INTEGER,
store_name                  STRING,
store_city                  STRING,
store_county                STRING,
store_state                 STRING,
store_country               STRING,
item_product_name           STRING,
item_class                  STRING,
item_category                STRING,
item_size                    STRING,
item_color                   STRING,
sale_date                    TIMESTAMP,
sale_is_holiday              STRING,
sale_is_weekend              STRING,
quantity                     DOUBLE,
list_price                   DOUBLE,
net_paid                     DOUBLE,
net_profit                   DOUBLE
);
```

Can be loaded with (for example):

```
JethroClient demo localhost:9111 -p jethro -i sales_demo.ddl
```

4.2 Description File

The description (.desc) file describes the structure of the input file, the way to map it to the schema and special formatting rules. It has three sections:

1. **Table-level section** – describe the format of the input file and some processing options.
2. **Column-level section** – a mapping between the fields in the file and columns in the table.
3. ***(Optional)*** **Record description section** – special clause for handling variable format files – files with different format per line (discussed in section **שגיאה! מקור ההפניה לא נמצא.**)

For example:

```
TABLE sales_demo
row format delimited
   fields terminated by '|'
   null defined as 'NULL'
(
customer_sk,
customer_salutation,
customer_first_name,
customer_last_name,
customer_preferred_flag,
customer_birth_year,
customer_birth_country,
customer_gender,
customer_marital_status,
customer_education_status,
store_sk,
store_name,
store_city,
store_county,
store_state,
store_country,
item_product_name,
item_class,
item_category,
item_size,
item_color,
sale_date,
sale_is_holiday,
sale_is_weekend,
quantity,
list_price,
net_paid,
net_profit
)
```

4.3 Loading Data

In order to load data, we will use the JethroLoader command line tool. It expects three parameters:

```
JethroLoader instance-name description-file
             stdin | input-location [input-location]...
```

- *Instance Name* – the name of the local instance to connect.
- *Description File* – the .desc (description) file.
- *Input location*– either stdin (to load from a pipe) or a list of one or one locations.
  Each location is a path of a file or directory to be loaded.
  Files can be a mix of uncompressed files and compressed files (.gz or .gzip extension) – compressed files are automatically uncompressed in-memory when they are read.

Examples:

- `JethroLoader demo t1.desc t1_input_dir`
- `JethroLoader demo t1.desc t1.part1.csv t1.part2.csv`
- `JethroLoader demo t1.desc HDFS://home/jethro/files/t1.gz`
- `sed ... | JethroLoader demo t1.desc stdin`

NOTE – when loading from a pipe, the program that sends data to `JethroLoader` may fail in the middle. In that case, the loader is not be notified of error - it gets a standard end-of-file (EOF) notification for its input, and therefore will commit the and exit without error.
It is your responsibility to monitor for such cases and respond accordingly.

4.4 Loading JethroData demo data

JethroData's demo data featuring the sales_demo table that was presented earlier can be easily downloaded and loaded to JethroData. It includes 100 million rows (about 3GB). A lighter 10M rows version (about 300MB) is also available.

100m Download: `http://www.jethrodata.com/demodata-100m`
10m Download: `http://www.jethrodata.com/demodata-10m`

Open the downloaded archive using `tar -xvf {filename}`

Along with the compressed .csv, the archive contains the CREATE TABLE script (.ddl file), the .desc file and various SQL queries.

Creating the table

Use `JethroClient` to run the CREATE TABLE script. You need to specify the instance name, the host and port of the `JethroServer`, the password (default is `jethro`) and in our case, also the script file:

```
JethroClient demo localhost:9111 -p jethro -i sales_demo.ddl
```

Loading the demo data

Use `JethroLoader` to load the CSV file. Since loading would take a while, we will run it in the background:

```
JethroLoader demo sales_demo.desc sales_demo.100m.csv.gz &
```

Check the loader results by looking at its report file. The report file will be in the current directory, named `loader_report_timestamp_`*`{pid}`*`.log`, for example:

```
more loader_report_20140630_124723_9359
```

4.5 Run Queries

Now you can connect to the JethroData instance through the shell or a client and run SQL queries. To ensure that the data load was successful, these basic queries can be used:

1. List all tables

```
SHOW tables;
```

2. List the columns and their types for a specific table:

```
DESC table_name;
```

3. List extended information about each table and column:

```
SHOW tables extended;
```

4. Check the number of rows inserted to a specific table:

```
SELECT COUNT(*)from table_name;
```

5 Managing JethroData Services

Concepts

A JethroData **instance** includes an entire database – all the tables, data, indexes and metadata. One or many JethroData hosts can be connected (attached) to the same instance. For example, you can add more hosts to handle more concurrent users (scale-out).

Each host can run the following **services** per attached instance.

- **JethroServer** – a stateless service that receives SQL commands from clients (ODBC / JDBC / JethroClient), executes them and returns the results back.
- **JethroMaint** – a background service that runs non-intrusive maintenance operations – optimizing the indexes and removing unused files. When multiple hosts access the same JethroData instance, only one of them should run this service.
  If this service is not running over a long period of time, you may experience degraded performance, especially after large data loads (you can check it with SHOW TABLES MAINT command).

Common Deployment Options:

- Typical **production** environment has a a single JethroData instance, and multiple hosts are attached to it. Each of those production hosts has its services automatically running all the time.
- Typical **development / test** environment has one or few JethroData hosts, each attached to multiple instances (dev / test / training etc), and not all the services are running at the same time. The administrator starts and stop the services on demand (to avoid memory pressure).

5.1 Using JethroData `service` Command

The JethroData installation RPM creates a linux O/S service for starting and stopping JethroData services. It simplifies and automates the startup of JethroData services for production environments, while providing the administrator full control for dev /test environments.

The `service` command is run as `jethro` O/S user (no `root` privileges required). It allows to optionally choose a specific instance and service to start or stop. For example:

- Starting all services that are marked for autostart: (see discussion of autostart below)
  ```
  $ service jethro start
  Instance demo service JethroServer started, pid: 24320, port: 9111
  Instance demo service JethroMaint  started, pid: 24407
  Instance demo service JethroServer started, pid: 24506, port: 9112
  ```
  This command also runs automatically during the host startup.
- Stopping all running services (also runs during the host shutdown):
  ```
  $ service jethro stop
  ```
- Listing all currently running services:
  ```
  $ service jethro status
  Instance demo service JethroMaint running, pid: 24407
  Instance demo service JethroServer running, pid: 24320 ,port: 9111
  Instance demo service JethroServer running, pid: 24506 ,port: 9112
  ```
- Starting `JethroServer` service of a specific instance:

```
$ service jethro start demo
```

- Stopping `JethroServer` service of a specific instance:
  ```
  $ service jethro stop demo
  ```
- Starting `JethroMaint` service of a specific instance:
  ```
  $ service jethro start demo maint
  ```
- Starting both services of a specific instance:
  ```
  $ service jethro start demo all
  ```

Services Autostart

The `service` command automatically monitors any JethroData service that it started (`JethroServer` and `JethroMaint`). If it detects that a running service was unexpectedly stopped, it will restart it.

In addition, the linux `service` will automatically start JethroData services during boot, based on a text configuration file called **`services.ini`** under `/opt/jethro/instances.` This file is automatically populated by the `JethroAdmin` commands. For example, `create-instance` adds a line describing the instance - if this is the only instance on the host, it will set autostart on, else it will set autostart off.

The `services.ini` file has a simple format. Each line in the file controls the autostart of a specific instance, and has the following format:

```
instance_name:port:start_query_service:start_maint_service
```

For example - with an instance named demo, listening at port 9111 (default), with autostart of both services:

```
demo:9111:yes:yes
```

6 Managing JethroData Instances

6.1 Log Files

Logs can be found under /var/log/jethro/*{instance_name}*. It contains five log files:

1. jethro_loader.log – Shows all information regarding JethroLoader's operations.
2. jethroserver.log – Stores all queries run on JethroData and their query plans, running time and more.
3. jethro.log – A generic log featuring more detailed log messages
4. jethro_maint.log – Shows all information regarding JethroMaint's operations.
5. services.log – Shows information regarding the service itself. If there are any issues during service startup or shutdown, it will be logged there.

6.2 Configuration Files

Two important instance configuration files can be found under $JETHRO_HOME/instances/instance_name:

1. local-conf.ini – Features the specific configurations for the instance
2. jethrolog.properties – Controls the log level and info that will be written in the log files mentioned above.

6.3 Managing Instances

Attaching Additional JethroData Hosts To An Instance

Once a JethroData instance is created on HDFS, additional JethroData hosts can be configured to access it. That allows supporting more concurrent users, and in addition, providing high-availability (see section for details).

To attach a JethroData host to an existing instance on HDFS:

- Prepare a new JethroData host – provision an appropriate host, set it up, install JethroData software and verify connectivity, as described in section 2.
- Attach the new host to the existing instance, providing the instance name, the instance directory in HDFS and the location and size of the local cache directory (similar to `create-instance` command). For example:

```
$ JethroAdmin attach-instance demo -storage-path=/user/jethro/instances
   -cache-path=/mnt/jethro_local_cache -cache-size=80G
```

Listing Locally Attached Instances

After running `create-instance` or `attach-instance`, the local JethroData host is attached to the HDFS instance. In other words, the HDFS instance is referred from the local JethroData host.

To list all the instances that are attached to the local JethroData host, run:

```
$ JethroAdmin list-instances
```

6.4 Kerberos-Specific Instance Setup and Parameters

JethroData's current support for accessing Kerberized Hadoop cluster requires the administrator to manually run `kinit` utility to generate tickets for JethroData services. This needs to be done for every JethroData host.

1. As `jethro` user, run `kinit` three times to generate a cache file per JethroData services / utilities (`JethroServer`, `JethroMaint` and `JethroLoader`):
   ```
   kinit -k -t jethro.hadoop.keytab -c jethro.server.cache jethro
   kinit -k -t jethro.hadoop.keytab -c jethro.maint.cache  jethro
   kinit -k -t jethro.hadoop.keytab -c jethro.loader.cache jethro
   ```

   It is recommended to ask for a renewable ticket with long expiration time (based on your site policies), for example adding a `-r999d` option.
2. In `$JETHRO_HOME/instances/`*`instance_name`*`/local-conf.ini` file, uncomment the following three parameters and set each to the full path of the corresponding cache file.
   ```
   hdfs.kerberos.server.ticket.cache.path
   hdfs.kerberos.maint.ticket.cache.path
   hdfs.kerberos.loader.ticket.cache.path
   ```

When the tickets expire, the administrator will need to repeat those steps. So, it is recommended to have a long ticket lifetime. Also, to avoid service disruption, it is best to periodically renew the Kerberos tickets before they expire, for example by running `kinit -R` from a `cron` job.

7 Removing JethroData

When removing JethroData, you may choose one of the following options.

Detaching a JethroData Host From An Existing Instance

To remove any reference to an existing instance from the local JethroData host **without** actually deleting the instance (tables and indexes), run:

```
JethroAdmin detach-instance instance-name
```

For example:

```
JethroAdmin detach-instance demo
```

The detach command also stops any local service that is running on this instance, if any.
You can later attach again to the same instance if needed, by running the `JethroAdmin attach-instance` command.

Deleting an Instance

NOTE: Deleting an instance **removes all its data** and also removes its references from the local JethroData host.

To delete an instance, run:

```
 JethroAdmin delete-instance instance_name
```

For example:

```
JethroAdmin delete-instance demo
```

You will be asked for confirmation before the instance is deleted.

If you had other servers attached to this instance, you will need to detach them as well.

Uninstalling JethroData Software

To remove JethroData software from a host, run as `root`:

```
rpm -e jethro
```

This command does not delete any instance, as the instances are shared across hosts. If you want to delete an instance, you should do it before you uninstall JethroData software.