**Epipog (v1.01 , Oct. 19, 2016)**

***Prelude***

EpiPog is an “academic” and “open source” program for teaching and developing NoSQL model of databases.

***License***

Licensed under the create commons license: CC-BY

***Command Interface***

Usage: epipog <options>

-i inputfile # import input file

-s field(s) # select fields ( use ‘\*’ for all)

-o field(s) # order by fields

-d datastore # type of data store (binary,psv,csv,json), default: binary

-F format # format of input file (psv,csv,tsv), default: csv

-C collection # name of the data collection

-S schema # the schema (key,type pairs)

-t storage # storage type (single,multi), default: single

-I index # index type (linked,binary), default: linked

-P primary # primary keys for indexing

-O sort # sorting algorithm (insert,quick), default: insert

-n # no header in input file, used retained/specified schema

-f filter # filter [STUBBED]

-c cache # size of cache [STUBBED]

-V # vacuum (remove deleted items) from collection [STUBBED]

***Command Line Examples***

When no collection, schema, primary keys, indexing, data store, storage types and input format are specified, they default as follows:

Collection: is named ‘tmp’  
Schema: no schema  
Primary: no primary keys  
Indexing: uses unsorted linear (linkedindex)  
Data Store: uses fixed-length records (binary)  
Storage Type uses Single: monolithic file under /tmp directory  
Input Format: CSV

Example: read in an input file into the default collection (for csv input)

* java epipog –I input.txt

Example: read in an input file into a named collection (for csv input)

* java epipog –I input.txt –C collection1

Example: read in an input file into a named collection and store as PSV format (for csv input)

* epipog –I input.txt –C collection2 –d psv

Example: select fields from default collection (for binary storage format)

* epipog –s country,state,city

Example: select fields from named collection (for binary storage format)

* epipog –s country,state,city –C collection1

Example: select fields from named collection in PSV storage format

* epipog –s country,state,city –C collection2 –d psv

Example: read in an input file and dynamically specify a schema and primary keys for indexing

* epipog –I input.txt -Scountry:string64,state:string64,city:string64 –P state,city

Example: select fields and order by a field

* epipog –s country,state –o country

Example: input tab delimited input file

* epipog –i input.tsv –F tsv

***File Structure***

Each component (public class) is contained wholly within its own file and corresponding file name.

Query.java – command line interface  
  
DataStore.java – abstract layer for data stores  
BinaryStore.java - derived class for representing data store as a fixed-length records.  
SVStore.java – abstract layer for representing data stores as character delimited file.  
PSVStore.java – derived class for representing data store as PSV format.   
CSVStore.java – derived class for representing data store as PSV format.  
JSONStore.java – derived class for representing data store as JSON objects.

Storage.java – abstract layer for on-disk storage  
SingleFileStorage.java – derived class for storing collection as a single monolithic file  
MultiFileStorage.java – derived class for storing collection as multiple files (stubbed)

Index.java – abstract layer for indexing.  
LinkedIndex.java – derived layer for representing index as an unsorted linear index.  
BinaryTreeIndex.java – derived layer for representing index as a sorted binary tree (stubbed).

Parse.java - abstract layer for parsing input file and inserting into the datastore  
SVParse.java - abstract layer for parsing a character delimited input file and inserting into datastore  
PSVParse.java - derived layer for parsing a PSV input file and inserting into the datastore  
CSVParse.java - derived layer for parsing a CSV input file and inserting into the datastore  
TSVParse.java - derived layer for parsing a TSV input file and inserting into the datastore

Data.java – abstract layer for returned data types from queries  
DataString.java – abstract layer for returned data string types from queries  
DataString16.java, … DataString128.java – derived layer for returned data types from queries  
DataShort.java,DataInteger.java,DataLong.java - derived layer for returned data types from queries  
DataFloat.java,DataDouble.java – derived layer for returned data types from queries  
DataDate.java,DataTime.java – derived layer for returned date types from queries

Sort.java – abstract layer for sorting results  
InsertSort.java – derived layer for sorting results  
QuickSort.java – derived layer for sorting results

QueryException.java – custom exception handler for the data store layer  
StorageException.java – custom exception handler for the storage layer.

Schema.java – schema handling