**Epipog (v1.0)**

***Prelude***

***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

***Command Interface***

Usage: query <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***

First, when no collection, schema, primary keys, indexing, data store, storage types and input format are specified, I default them as follows:

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

Example: read in an input file into the default collection

* java query –I input.txt

Example: read in an input file into a named collection

* query –I input.txt –C collection1

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

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

Example: select fields from default collection

* query –s STB,TITLE,VIEW\_TIME

Example: select fields from named collection

* query –s STB,TITLE,VIEW\_TIME –C collection1

Example: select fields from named collection in PSV storage format

* query –s STB,TITLE,VIEW\_TIME –C collection2 –d psv

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

* query –I input.txt –S STB:string64,TITLE:string64,PROVIDER:string64,DATE:date,REV:float,VIEW\_TIME:time –P STB,TITLE,DATE

Example: select fields and order by a field

* query –s STB,TITLE,VIEW\_TIME –o TITLE

Example: input tab delimited input file

* query –i input.tsv –F tsv

**Bugs**

1. Does not handle non-ASCII characters in input
2. Can’t do 2nd level sort
3. Index does not handle collisions

**Code Improvements**

1. In the Select() method of BinaryStore, should use index to jump to position of next record. If dirty, then skip reading.
2. In the Select() method of JSONStore, should use JSONObject parser (instead of hand written).
3. Command line option/argument parsing should use a standardized getopt() library (instead of hand written).
4. Make a jar file.
5. Add config file for default settings