DataBoss

Let the Boss manage your records

Outline

- Introduction
- Design
- User Guide
- Summary
- References

Introduction

Following the requirements we needed to design and implement a program to do the following:

- Display
- Add
- Delete
- Search
- Help

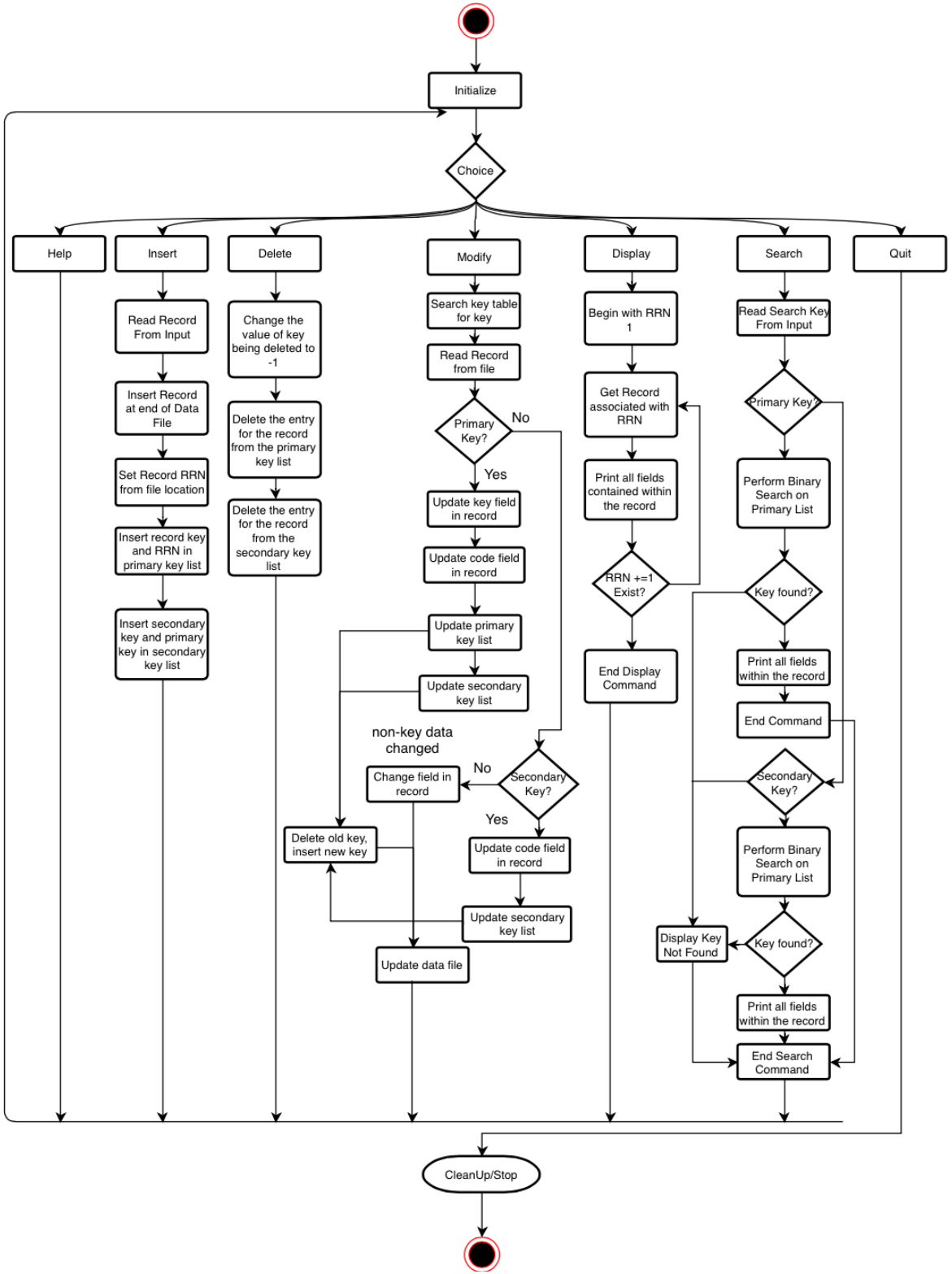
On a data file, primary index, and a secondary index (represented as an inverted list)

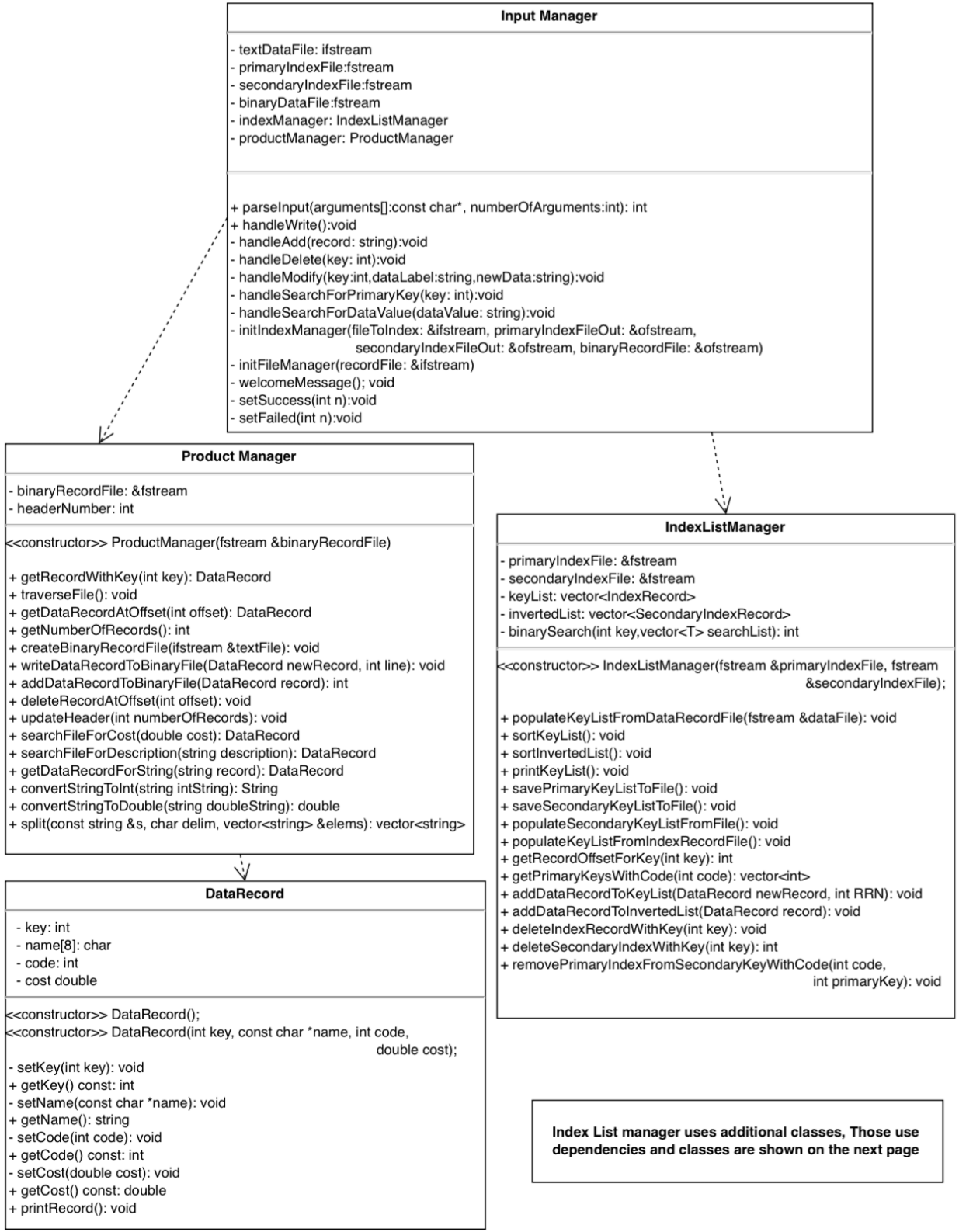
Design

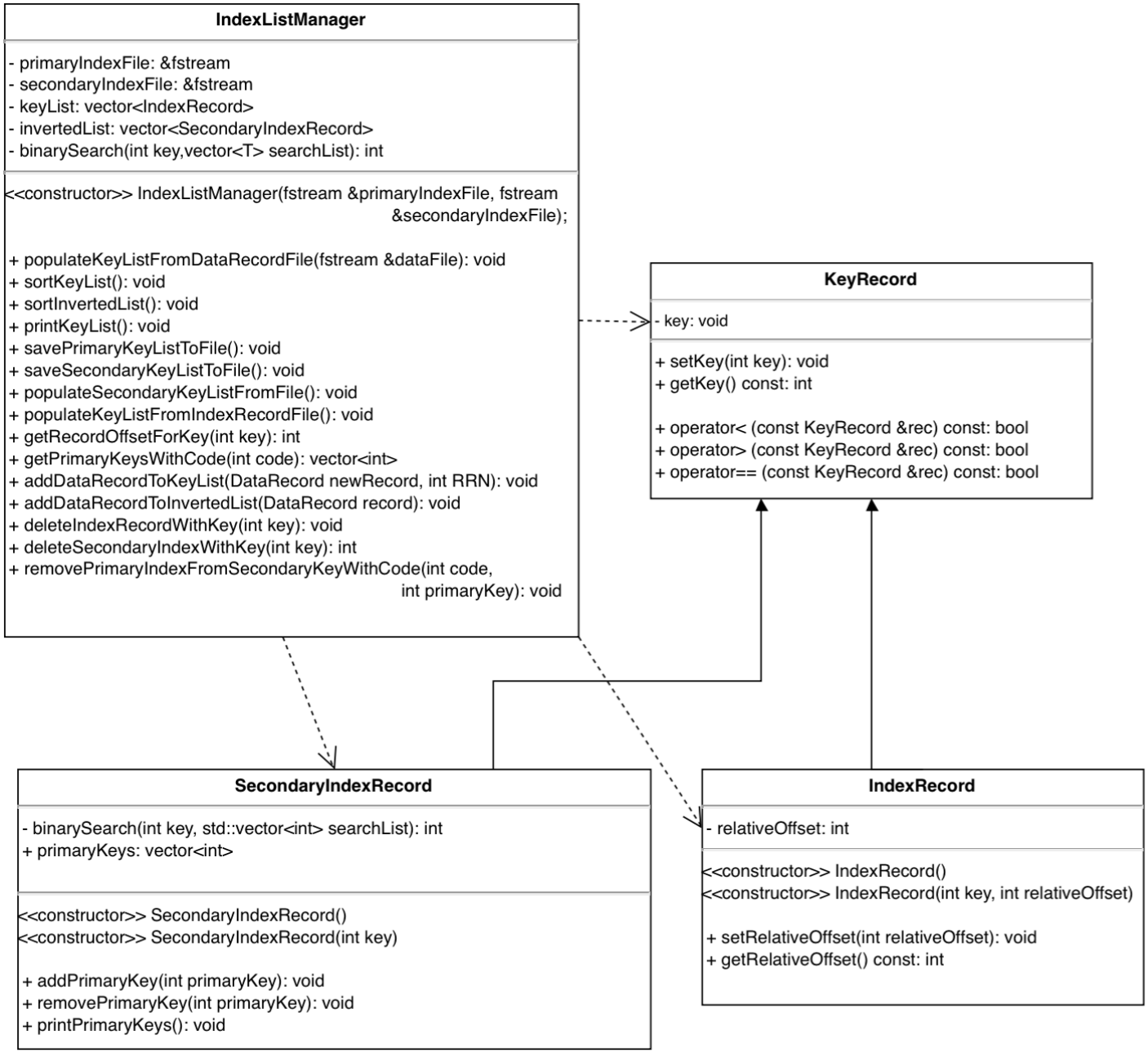
The Planning Process

Design

- Focused on modularity and code re-use
- Inheritance was used, within the record classes
- DataBoss's flowchart matches the proposal flow chart, plus one additional command


Initialize
Choice
Help
Insert
Delete
Modify
Display
Search
Quit
Read Record From Input
Insert Record at end of Data File
Set Record RRN from file location
Insert record key and RRN in primary key list
Insert secondary key and primary key in secondary key list
Change the value of key being deleted to -1
Delete the entry for the record from the primary key list
Delete the entry for the record from the secondary key list
Search key table for key
Read Record from file
Primary Key?
No
Yes
Update key field in record
Update code field in record
Update primary key list
Update secondary key list
non-key data changed
Secondary Key?
No
Change field in record
Yes
Update code field in record
Delete old key, insert new key
Update secondary key list
Update data file
Begin with RRN 1
Get Record associated with RRN
Print all fields contained within the record
RRN +=1 Exist?
End Display Command
Read Search Key From Input
Primary Key?
Perform Binary Search on Primary List
Key found?
Print all fields within the record
End Command
Secondary Key?
Perform Binary Search on Primary List
Key found?
Display Key Not Found
Print all fields within the record
End Search Command
CleanUp/Stop


Input Manager
- textDataFile: ifstream
- primaryIndexFile:fstream
- secondaryIndexFile:fstream
- binaryDataFile:fstream
- indexManager: IndexListManager
- productManager: ProductManager
+ parseInput(arguments[]:const char*, numberOfArguments:int): int
+ handleWrite():void
- handleAdd(record: string):void
- handleDelete(key: int):void
- handleModify(key:int,dataLabel:string,newData:string):void
- handleSearchForPrimaryKey(key: int):void
- handleSearchForDataValue(dataValue: string):void
- initIndexManager(fileToIndex: &ifstream, primaryIndexFileOut: &ofstream,
secondaryIndexFileOut: &ofstream, binaryRecordFile: &ofstream)
- initFileManager(recordFile: &ifstream)
- welcomeMessage(); void
- setSuccess(int n):void
- setFailed(int n):void
Product Manager
- binaryRecordFile: &fstream
- headerNumber: int
<<constructor>> ProductManager(fstream &binaryRecordFile)
+ getRecordWithKey(int key): DataRecord
+ traverseFile(): void
+ getDataRecordAtOffset(int offset): DataRecord
+ getNumberOfRecords(): int
+ createBinaryRecordFile(ifstream &textFile): void
+ writeDataRecordToBinaryFile(DataRecord newRecord, int line): void
+ addDataRecordToBinaryFile(DataRecord record): int
+ deleteRecordAtOffset(int offset): void
+ updateHeader(int numberOfRecords): void
+ searchFileForCost(double cost): DataRecord
+ searchFileForDescription(string description): DataRecord
+ getDataRecordForString(string record): DataRecord
+ convertStringToInt(string intString): String
+ convertStringToDouble(string doubleString): double
+ split(const string &s, char delim, vector<string> &elems): vector<string>
IndexListManager
- primaryIndexFile: &fstream
- secondaryIndexFile: &fstream
- keyList: vector<IndexRecord>
- invertedList: vector<SecondaryIndexRecord>
- binarySearch(int key,vector<T> searchList): int
<<constructor>> IndexListManager(fstream &primaryIndexFile, fstream
&secondaryIndexFile);
+ populateKeyListFromDataRecordFile(fstream &dataFile): void
+ sortKeyList(): void
+ sortInvertedList(): void
+ printKeyList(): void
+ savePrimaryKeyListToFile(): void
+ saveSecondaryKeyListToFile(): void
+ populateSecondaryKeyListFromFile(): void
+ populateKeyListFromIndexRecordFile(): void
+ getRecordOffsetForKey(int key): int
+ getPrimaryKeysWithCode(int code): vector<int>
+ addDataRecordToKeyList(DataRecord newRecord, int RRN): void
+ addDataRecordToInvertedList(DataRecord record): void
+ deleteIndexRecordWithKey(int key): void
+ deleteSecondaryIndexWithKey(int key): int
+ removePrimaryIndexFromSecondaryKeyWithCode(int code,
int primaryKey): void
DataRecord
- key: int
- name[8]: char
- code: int
- cost double
<<constructor>> DataRecord();
<<constructor>> DataRecord(int key, const char *name, int code,
double cost);
- setKey(int key): void
+ getKey() const: int
- setName(const char *name): void
+ getName(): string
- setCode(int code): void
+ getCode() const: int
- setCost(double cost): void
+ getCost() const: double
+ printRecord(): void
Index List manager uses additional classes, Those use dependencies and classes are shown on the next page


IndexListManager
- primaryIndexFile: &fstream
- secondaryIndexFile: &fstream
- keyList: vector<IndexRecord>
- invertedList: vector<SecondaryIndexRecord>
- binarySearch(int key,vector<T> searchList): int
<<constructor>> IndexListManager(fstream &primaryIndexFile, fstream &secondaryIndexFile);
+ populateKeyListFromDataRecordFile(fstream &dataFile): void
+ sortKeyList(): void
+ sortInvertedList(): void
+ printKeyList(): void
+ savePrimaryKeyListToFile(): void
+ saveSecondaryKeyListToFile(): void
+ populateSecondaryKeyListFromFile(): void
+ populateKeyListFromIndexRecordFile(): void
+ getRecordOffsetForKey(int key): int
+ getPrimaryKeysWithCode(int code): vector<int>
+ addDataRecordToKeyList(DataRecord newRecord, int RRN): void
+ addDataRecordToInvertedList(DataRecord record): void
+ deleteIndexRecordWithKey(int key): void
+ deleteSecondaryIndexWithKey(int key): int
+ removePrimaryIndexFromSecondaryKeyWithCode(int code, int primaryKey): void
KeyRecord
- key: void
+ setKey(int key): void
+ getKey() const: int
+ operator< (const KeyRecord &rec) const: bool
+ operator> (const KeyRecord &rec) const: bool
+ operator== (const KeyRecord &rec) const: bool
SecondaryIndexRecord
- binarySearch(int key, std::vector<int> searchList): int
+ primaryKeys: vector<int>
<<constructor>> SecondaryIndexRecord()
<<constructor>> SecondaryIndexRecord(int key)
+ addPrimaryKey(int primaryKey): void
+ removePrimaryKey(int primaryKey): void
+ printPrimaryKeys(): void
IndexRecord
- relativeOffset: int
<<constructor>> IndexRecord()
<<constructor>> IndexRecord(int key, int relativeOffset)
+ setRelativeOffset(int relativeOffset): void
+ getRelativeOffset() const: int

User Guide

How to use DataBoss

Commands

- Display - Displays all of the data currently stored by DataBoss
- Add - Adds a record to the data file, primary index and secondary index
- Delete - removes a record from the data file, primary index or secondary index.
- Search - searches the primary index, secondary index or the data file
- Help - displays available commands and proper syntax to execute the commands

Summary

- DataBoss is a simple record management solution
- We will now demonstrate DataBoss….