Sports Arbitrage and More

A Real-Time System for Sports Arbitrage.

Submitted May 2009, in partial fulfilment of
the conditions of the award of the degree Software Systems BSc (Hons) G601

Jonathan Stuteley
Jas06u

School of Computer Science and Information Technology
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature ________________________

Date _____/_____/_____

Abstract

In certain situations it is possible to make a guaranteed profit by placing proportional bets with multiple bookmakers on all of the outcomes of a single sporting event.

This project attempts to automate the process of finding these profitable opportunities by:

- Acquiring odds data from bookmakers.
- Interpreting the odds data to group comparable odds together.
- Processing the data to find collections of outcome odds that could potentially form a profitable opportunity.
- Evaluating the profitability of opportunities.
- Presenting opportunities and odds data to the end user, so that they could potentially take advantage of them.

Contents

1. Motivation For The Work

In finance "arbitrage" is defined as '…the practice of taking advantage of a price differential between two or more markets…'(Wikipedia). Sports arbitrage is taking advantage of situations where it is possible to bet on all outcomes of a sporting event and make a guaranteed profit.

I shall attempt to explain why these situations occur by drawing parallels with currency exchange (in *italics*).

- When bookmakers (bookies) set the odds for the outcome of an event, they set them so that it is impossible to make a profit by betting on all event outcomes (through them).

  It is impossible to make a profit by buying foreign currency from a local travel agent then immediately selling it back to them as the agent's buy price is lower than its sell price.

- As different bookies take bets on the same events and want to get more of the market share of wagers than other bookies the odds get more competitive. As odds vary between bookies and change with time, profitable opportunities (including sports arbitrage) can occur.

  Different foreign currency exchanges sell the same currency at different exchange rates. American dollars may be cheaper at travel agent #1 than they are at travel agent #2, and it may be possible to make a profit from buying and selling American dollars.

- A profitable opportunity might occur on a sporting fixture involving team #1 and team #2 in an event where there must be a winner (no tie allowed). This might consist of betting on team #1 to win with bookmaker #1 and team #2 to win with bookmaker #2.

  It may be possible to make a profit by buying American dollars from travel agent #1 and selling them to travel agent #2.

- Profit is possible with sports arbitrage by betting proportionally on all of the outcomes of a sporting event as long as the reciprocals of all of the (decimal) odds taken add up to less than 1.

  Profit is possible with buying and selling foreign currency if a person can buy at a lower price than he/she can sell at.

In both of these situations (sports arbitrage and the currency exchange example) the key to being able to make a profit is having accurate odds information (*or buy and sell prices*) so that an informed decision about the profitability of a situation can be made.

This project attempts to automate the acquisition and interpretation of odds data before presenting opportunities and a rating as to their profitability to the user of the end product so that they might take advantage of it.

2. Description Of The Work

Initial Systems Request

Appendix A details the systems request received from the project's customer.

During the initial project meeting the customer divulged the information that a prototype system had been produced to clarify their requirements and test whether a sports arbitrage system was indeed plausible. The customer was able to give this project some access to the prototype code, full access to the database and access to all of the documentation that was produced (although this was limited).

The prototype was later analysed (documented in the Related Work section) and the following revised specification was developed and approved by the customer.

Revised Specification

The revisions from the initial Systems Request that were identified by this project are in *italics*.

Functional Requirements:

- Be able to retrieve data from up to 100 bookmakers/sources of odds.
- Record odds from numerous events and various sports.
- *Be able to record various types of odds given for different occurrences (e.g. result odds, score odds, handicap result odds etc).*
- Be able to recognize when odds from different bookmakers are comparable (i.e. predict the same outcome of the same event).
- *Be able to identify various types of "profitable opportunities" including but not limited to "arbitrages" and "middles".*
- *Be able to identify "profitable opportunities" as soon as the data is available and highlight them to a user.*
- *Allow the user to view data in at least the following ways:-*
    1. *All profitable opportunities, sorted by profitability.*
    2. *Sporting events, sorted by start time/date.*

  After selection of a fixture:

    3. *All odds on this event grouped by their type and then grouped with others that they logically should be (e.g. all "Home Win" odds grouped with "Draw" and "Away Win" odds).*
    4. *All opportunities on this event, sorted by profitability, with all the odds of the bet types that it consists of.*

  After selection of an opportunity:

    5. *All odds for the constituent bet types.*

- Allows an "administrator" to amend participant alias data.
- Keep a history of all odds collected for future analysis. This should be able to be backed up by administrator actions.

Non-functional Requirements:

- Due to the involvement of multiple project teams working towards similar goals, the customer would like the data storage and its interface(s) implemented as soon as possible. The customer can then let the other projects use these with a view to having multiple products that can be interfaced with the same "core" component.
- *The system should be implemented with a view to cater for easy expansion of types of odds stored and types of opportunities calculated.*
- The system should be implemented so that the client's programming staff can maintain the supplied data miners (and parsers) and write more of their own.
- *The system should be well documented to facilitate the above two requirements*.
- The system should cater for data retrieval from xml and html sources.
- *It should be possible for the data acquisition to be done in (numerous) separate locations then submitted to a central location for storage.*
- It should be possible to view data as a user from any computer (via the internet).
- The system should be installable on standard Microsoft XP/Vista based PCs.
- *The system should run on one PC but be optimal on 3+, allowing for the other requirements to be fulfilled.*

Other Project Considerations

Here is a list of considerations that need to be taken into account:

- Bookmakers may not like people trying to find sports arbitrage opportunities. Although on the face of it they may get some business that they may not have otherwise got, bookmakers as a whole lose out. This may lead bookmakers to employ countermeasures.
- A bookmaker may stop accepting bets (or suspend accounts) from users that try to access their website from an IP address that interrogates their website too regularly.
- Bookmakers may stop an IP address from accessing their website faster than at a pre-defined rate.
- Bookmakers may sometimes input odds wrongly. These could be incorrect or just transposed (e.g. between home win and away win).
- Bookmakers may get suspicious of accounts that place bets using strange stakes (e.g. £52.61).

Real-Time Term Clarification

The term "real-time" is a rather relative term that implies a quick and highly responsive system but this is relative to the context of the system's operation. This project will be real-time in that the data and information presented should be a true reflection of the odds market at the present time, with opportunities that can be acted upon. In the Related Work section of this report a rival piece of software "…aim(s) to deliver arbs to you within 60 seconds of their existence". Although no response restrictions have been requested by the customer, this project certainly aims to compete with this response time, although a lot is dependant on network connection speeds.

3. Project Planning

Choice of Design Methodology

The project did not conform to a standard development methodology as there was a prototype of a system (produced for the customer prior to project commencement) that contained certain characteristics that were requested to be included in the new system. There was some access to the prototype code, full access to the database and access to all of the documentation that was produced (although this was limited). A Hybrid of methodologies was most appropriate here.

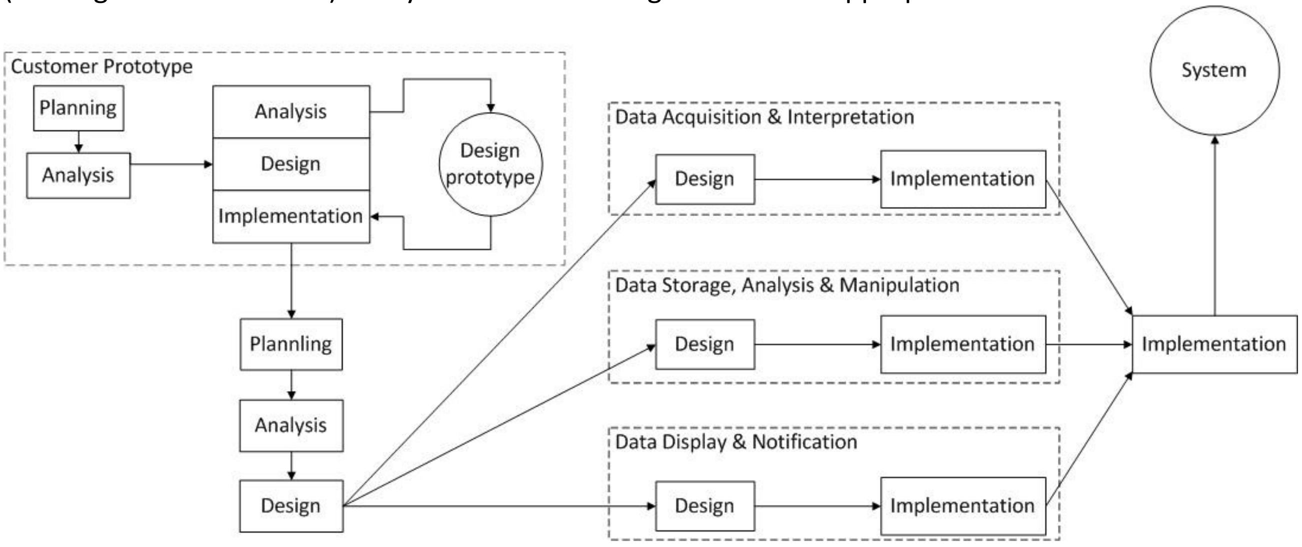


Figure 3.1 - Project Design Methodology

With access to a prototype and with the customer's evaluation of it, the project could be considered to have started in the middle of the Throwaway Prototyping methodology, at the stage where evaluation of the prototype was taking place. More work was required on evaluation of the prototype, before a system design could be produced and a design methodology decided upon for moving forward.

Following further analysis of the prototype, it was evident that the project broke down into three main sections, which could be developed independently (following initial overall system design).

These sections were:

- Data acquisition and interpretation.
- Data storage, analysis and manipulation.
- Data display and notification.

Communications between these sections' development processes would only be required in order to confirm the interfaces between those parts of the system. This conformed to the standard design methodology of Parallel Development, which has been employed for the remainder of the project.

Time Plan

One of the requirements of the project was to produce the data storage part of the system (and its interfaces) at the earliest possible time, so that other projects (aimed solely at the data acquisition or data display sections of this project's specification) could use this common module. The overall result for the customer being a full product (this project) with optional data acquisition and display modules that could be plugged in, instead of the original ones. The data storage and its interface was therefore the priority when constructing the time plan for the project. A Gantt chart showing the project schedule is included as Appendix B.

Plan B

Overall, the project is keeping quite close to the schedule although the time plan does not allow much slack time to take up unforeseen extensions in workload or reduction in manpower. This will be closely monitored and should there be the need to cut the workload down a list of timesaving initiatives has been drawn up. This is not a negative step, it is taking heed of the old adage 'prepare for the worst, hope the best'.

- Curtailment of Data Display & Notification module. – It is known that the customer has another project team working solely on a module to carry out this function. Although a module is needed by this project to demonstrate the capabilities of the rest of the package, a fairly simple implementation would suffice.

- Reduction in the number of data miners produced. – Although the number of miners that this project will include has not been set, it is hoped that it will be at least six. The project needs to produce at least two data miners to demonstrate mining both types of data source (HTML and XML). As with the display, miners are also required to demonstrate the rest of the package. It is fully appreciated however that the ability to easily create and add more data miners to the package is a more important function and that the quality of this should not be sacrificed for the quantity of supplied miners.

- Reduction in the number of opportunity types searched for. – The implementation of the opportunity finder is in two stages; a "Basic" followed by a later "Real" implementation. The number of plausible opportunity types is quite high (possibly up to 100 just on football alone) with some being mathematically quite complicated. As above, the ability to easily create new opportunity types and to demonstrate functionality outweighs the need to develop many.

4. Related Work

Other sports arbitrage systems are frequently advertised online but they all cost to use and download. It is a bit of a black subject as surely a working system makes the owner/writer money. If it were doing this then why would they want to advertise the fact or share the creation by selling the software (or a service), when the attention might lead bookmakers to counter the arbitrage efforts?

Mike Moffatt of “Your Guide to Economics” at About.com makes the following argument about sports arbitrage software available on the Internet:

> ‘…There's a more fundamental reason why I believe such software is not likely to work. If someone has discovered a system where they can make a lot of money easily by betting on sporting events, why would they tell you about it? Sure, they make money from selling the software. But by letting the world know about their system, they'd end up killing the golden goose. […] If these pieces of software truly worked, the writers of such software would have to make an awful lot of money by selling the software, because by doing so, they're making it a lot more difficult to use the software for themselves to make easy money...’

Below is a short analysis of the customer’s prototype followed by the results of limited research into the features of other systems advertised on the Internet. Then there is a description of the features that make this project’s system unique.

Short Analysis of Prototype

Prototype Design

The prototype was developed in two stages by different teams. The first team focused on showing that the system was feasible whilst the second worked on improving the data acquisition part of the system. Below is a diagram of the overall system design.

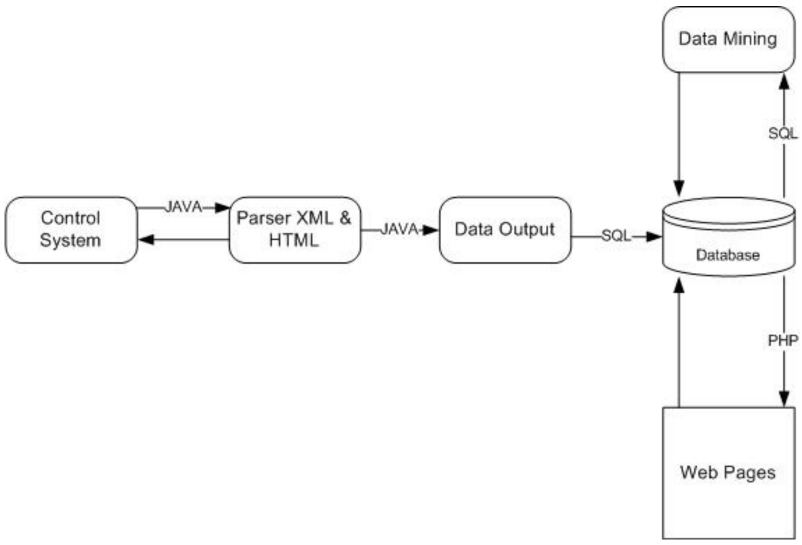


Figure 4.1 - Overall design of customer’s prototype

This design had a lot of merit. It separated the system into four modules that had distinct responsibilities:

- Controlling and monitoring the data acquisition.
- Parsing data.
- Storing data.
- Displaying data.

The running prototype worked effectively and (on the whole) the customer was happy with it. Analysis during this project showed many positive points in the prototype's design and implementation, but below are the negatives that were identified.

Prototype Critique

The customer identified the following areas for improvement (how this affected the System Request is in *italics*):

1. The PHP data view needed to be refreshed to get up-to-date data. It did not change when the data changed. – *Requested data view to change "as soon as the data is available".*
2. There was no history of how the odds changed over time. – *Requested a record of all odds to be kept.*
3. Certain parsers did not work as well as they had originally, in fact one didn't work at all. The code was difficult to follow and there was no documentation on how the parser/miner worked, so it could not be fixed. – *Requested that the data mining (and parsing) code be implemented so that it could be maintained by the customer's programming staff and that documentation facilitated this too.*

Extra criticisms identified by this project (how this affected the Revised Specification is in *italics*):

4. The ability to record odds data was very limited at just win, lose or draw. - *This needed to be extended and preferably unlimited.*
5. The ability to record only arbitrages limited the system with consideration of (4). - *This needed to be extended and preferably unlimited.*
6. The data acquisition all being done by one program on one computer was a potential bottle-neck due to bookmaker limits as to how often odds can be retrieved from a single IP address. Also, adding lots of miners/parsers may have put too much load on a single processor. – *Allowing the data store to run independently from the data acquisition and on a separate machine would allow the load to be shared. Multiple data acquisition modules could then be running on separate machines and feeding the same data store via remote connection.*

Other Software

Below are what other sports arbitrage software claim to be their features.

ArbitragePro (www.arbitragepro.com)

- Faster discovery and delivery of arbs. We aim to deliver arbs to you within 60 seconds of their existence.
- More arbs due to the speed of multi-computer scanning.

- Validated arbs - no rules mismatches with tennis, baseball and ice hockey. No inconsistent handicaps in American sports.
- Option to have rules-mismatched arbs in tennis and baseball.
- Filtering by sport, account or arb percentage.
- Set bookmaker currencies.
- Arb calculator that works with multiple currencies.
- Set individual exchange commission rates.
- We recognise the extra value of middles and highlight these to maximize your profit.
- Online service so you can receive the service wherever you are, not just where you receive your email.

Software from Sports-arbitrage.com

- This software downloads the lines directly from the sportsbooks to your computer!
- Easily spot bad lines! Speculators can instantly compare every price available on an event.
- Set WRF software to download multiple books at the same time, many opportunities only last a few minutes. If you have a fast computer and a nice internet connection, you can get the lines within a few seconds after the sportsbooks update their servers.
- Have an alarm sound on your computer speakers when an arbitrage is found.

What Makes This Project's Software Unique

The aim of this project is to create a sports arbitrage system even though it is dubious as to whether it is possible to actually make money from sports arbitrage. If it is possible to make money then by using the same piece of software as hundreds of other people, who will be notified about the same opportunities at the same time, a user would have to be very alert in order to take advantage of the opportunity before bookies realise what is happening and change their odds to remove the opportunity.

By producing a brand new piece of software from scratch, this project is maximizing the possibility of identifying an opportunity and notifying a user before any other system does because it scans a different range of bookies in a different way.

This project embraces the best parts of the customer's prototype, features claimed by other systems and ideas generated by the development team. It must also be pointed out that this software, unlike software that is available to purchase (of the type identified above), is aimed at a customer that has the necessary resources to add to the software, on a programming level. As a result some of the features identified will not be available from a user interface but will be code or database design features that have been defined or agreed by this specific customer. The benefits of many of these features will be evident to the user of the product.

Features include:

- Definable bet types for use with definable sports and definable opportunity types. This makes the system capable of being used with a huge (possibly unbounded) range of bet types from all kinds of sports and other bet-able events.
- More frequent scanning of bookies odds by optional use of multi-computer scanning.

- Online end-user interface that allows user to access the system from any computer with Internet access.
- Definable opportunity types to take advantage of "middles" as well as arbitrages.
- Define payout rules for individual sport-bookie combinations to highlight issues that might affect validity of opportunities.
- By displaying all of the available odds, not just the best ones, it is easy to spot incorrect odds (normally bookies input errors) and evaluate whether to take advantage of the alternative odds from another bookie.

Unique Selling Point

The first feature mentioned in the system features was "Definable bet types for use with definable sports and definable opportunity types". As far as this project is aware (from the limited knowledge of working systems due to Mike Moffat's Golden Goose argument) no other system offers this level definition freedom. This feature gives the system the ability to search for a potentially huge range of opportunity types. This also means that it doesn't necessarily need to compete with other systems on speed of data acquisition, speed of user notification or range of bookies interrogated even though it should do just that. (The only time constraint found was ArbitragePro's claim to notify within 60 secs of odds change.)

5. Design

Overall System Design

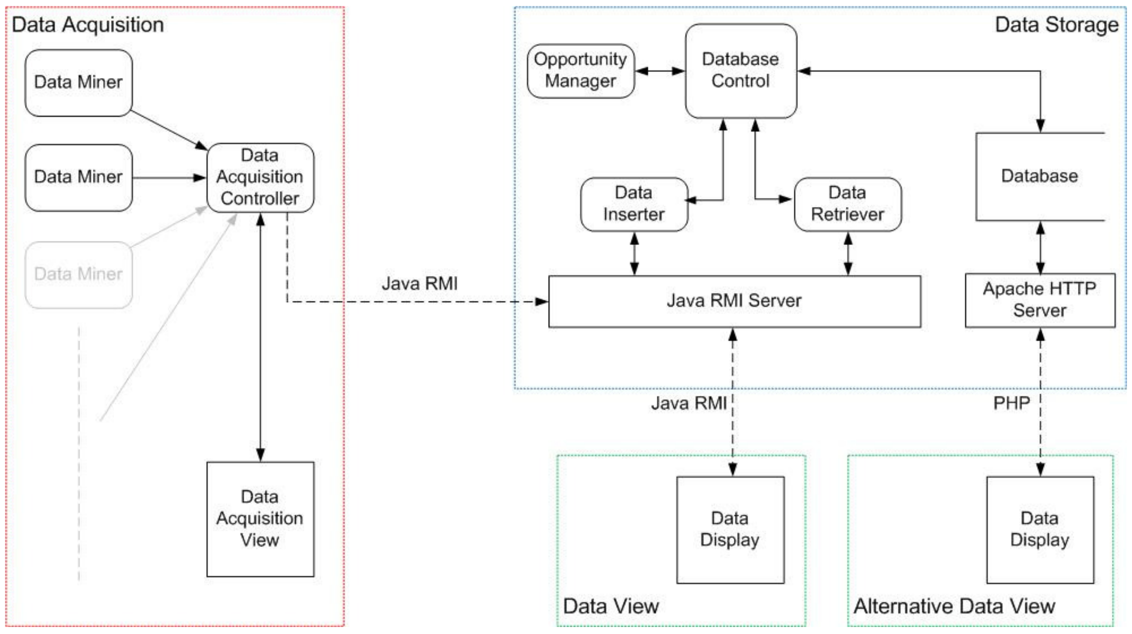


Figure 5.1 - Overall Design.

Options have been left open for a couple of possible Data View modules as depicted on the overall design above. This is due to both being equally viable and the possibility of other projects creating a data view that interfaces with this the Data Storage module of this project.

Communal Files

There are certain data containers, interfaces and constants that will be used by multiple programs in this system, so there will be an archive (.jar) file that contains all of these that will be required by the java programs; certainly the Data Acquisition and Data Storage modules.

The data containers keep all submitted data standardized and encapsulated in meaningful packages that contain information about real objects, while the constants allow the code to reference by name instead of using seemingly random numbers. The constants often reflect entries in the “Manually Inserted Reference Information” part of the database.

There will be at least the following objects:

- A BetSubmissionObject – holding possible bet data.
- An EventSubmissionObject – holding event data.
- A SubmissionObject – holding a single EventSubmissionObject and multiple BetSubmissionObjects.
- A class for holding constants that can be referenced.

Data Storage

The Data Storage module of this system has three major responsibilities:

1. To insert data into the database that it receives from Data Acquisition modules.
2. To manage the data in the database.
3. To retrieve data from the database that is requested by the Data View (should this be implemented).

The only one of these that is (currently) important from an external point of view is the first; the insertion of data. A submission is carried out by calling three methods on the Data Inserter (via the RMI Server). The first requests authority to submit, the second submits data (this can be called multiple times) and the third notifies that submission has finished.

To start submitting the submitter must request to submit its current bookmaker's data by calling a method and passing in the bookie's ID. If authority is granted (if it is not currently accepting data about that bookie) a unique identifier is returned to submit the data with. This is so that two bookies don't submit similar data at the same time and confuse the system when it tries to delete data that wasn't updated in the most recent submission. This authority request also starts a timer that automatically notify itself of the senders desire to finish submitting if it is taking more than a pre-defined time to complete its submission. This is designed to stop a submitter from blocking other submitters indefinitely if it takes too long to submit or stops submitting due to error conditions (loss of network connection etc).

As responsibilities one and two (from above) both require access to the database and will be running concurrently, it is possible that the data insertion mechanism will read some data and later rely on it even though the data management has already deleted it. There is therefore a need to guard against this happening by introducing a locking mechanism to implement atomic actions for multiple accesses to the database.

Database

Data Flow Diagrams for inputting data into the database (following data acquisition) can be found in Appendix C. These were developed alongside the database tables (Appendix D) whilst considering all of the operations and checks that need to be carried out on new data and following the insertion of new data. Sample database data and descriptions of table columns are in Appendix E.

The data storage requirements for this project fall into four main categories:

- Manually Inserted Reference Information (6 tables) –

  This is data that doesn't change and is used to classify data into sections and will therefore be repeated as easy to reference Java constants for use in the code. It is also used to describe data for display purposes by way of referencing to it.
- Current Market Information (6 tables) –

  This is defined as all of the data that has been collected and inserted into the database (primary data). It consists of event, bet and bookie information as well as references to data in the Manually Inserted Reference Information section. The "Bookies" table is in this section but would be equally at home in the Manually Inserted Reference Information section as some of its data is manually inserted.

- Current Opportunity Information (2 tables) –
  This is the data representing the opportunities that have been derived from the Current Market Information. This information is managed by the Opportunity Finder.
- Past Information (5 tables) –
  The history of bets and opportunities is secondary data written to by the Data Inserter and the Opportunity Finder respectively. This data will need to be manually managed as the customer didn't want a mechanism to automatically control/restrict the size of this section.

Database Control

This part of the system is responsible for accessing the database and will have methods for all access types required. Each method will have at least one database query associated with it. This is the only part of the system that accesses the database directly.

Data Inserter

Data Flow Diagrams (DFDs) for this section (located in Appendix C) may help when trying to understand the flow and manipulation of data in this section. There are nine DFDs entitled:

• Overview	- An overview of the data insertion operation.
• Input Event	- Overview of inputting/locating an event.
• Event with 1 ID	- Locating an event with only 1 referenced alias.
• Event with no Ids	- Locating an event with no referenced aliases.
• Insert/Update bet	- Inserting or updating a possible bet.
• Current Data Management	- Keeping Current Market Info' section up-to-date.
• Opportunity Finder	- Inserting into the Current Opportunity section.
• Opportunity Updater	- Updating Opportunities Past and Current.
• Post-Insertion Checks	- Updating Current Market Information.

Data will be submitted to the Data Inserter in the standard objects that were mentioned above in the Communal Files section.

Event Identification

In order for this project to produce a functional piece of software, it is imperative that data from bookie A is matched against data from bookie B. In order to do this bets from bookie A on an event have to be matched against the same event identified by bookie B, using three critical pieces of data; the event start timestamp (time/date) and the participant aliases. Although time is easy to compare (as long as the same datum is being used) participant names are not, especially when Manchester United can be identified by "Man U", "Man Utd", "Manchester Utd" or its full title to name a few possibilities. Worse still, the bookie may miss-spell whichever abbreviation they choose!

This event identification problem will be addressed using a number of techniques around the same theme of keeping an alias list for all known participants. The techniques are as follows:

1. First of all the alias string will be processed in the following way so that it observes a standard form. This is to limit the number of aliases that are saved in the database and to give an alias the best chance of being recognised. The processing steps are:

- Full stops are replaced with spaces.
- Spaces at the start and end of the String are removed.
- Any capital letters are turned to lower case.
- Multiple spaces are replaced with single spaces.
- All characters not between “a” and “z” (except the space character) are replaced with the “?” character.

2. All of a participant’s aliases will map to the participant they’re assigned to, so any of the inserted aliases for Manchester United will identify the same participant.
3. If an unknown alias is used, the system will insert it into the acceptance list only if the other participant and timestamp correctly identify a current event and there is a “basic” string match between the unknown alias and one of the known aliases for that participant. (See DFD “Event with 1 ID”.)
4. If two unknown aliases are used, the system will insert the aliases into the acceptance list only if there is an event already defined that has “good” string matches between the two unknown aliases and the aliases of the participants of an event. (See DFD “Event with no Ids”.)

If participant IDs and a timestamp do not identify a defined event (after an attempt to transpose the aliases) a new event is inserted into the Current Market Information and Past Information sections of the database. (See DFD “Input Event”.)

String Matching

String matching is a potential area for a lot of future research. In this project (under the heading of Event Identification) the requirement for two levels of string matching has been identified:

1. A “basic” string match for use when the strings are suspected to be a match and quick confirmation is required.
2. A “good” string match when the string is being tested against another speculatively and needs to be a closer match in order to pass.

Possible Bet Entry/Update (See DFD “Insert/Update bet”.)

In order to insert bet information into the database the type of the bet must first be identified in order to be able to compare it with like bets. Each bet type will be identified using the event that it takes place on, the style of the bet and the bet style mutator.

Event –	This was retrieved or generated during the Event Identification phase.
Bet Style –	This was classified before its submission and could take values of “Result Participant#1 win”, “Asian Handicap Participant#2 win”, “Handicap Result Draw” or many more.
Bet Style Mutator –	Certain bet styles need number to go with them to make sense. A football match result home win would not need a mutator, but an Asian handicap away win would need a number like “+2” or ”-0.5” for example, to make any sense and define the bet so that it could be compared against others of exactly the same type.

If this bet type has already been identified a search will be carried out looking for this bet type and bookie combination. If one is found it will update the odds and timestamp, otherwise a new possible bet will be created. Either way, the best odds for this bet type will then be updated only if it is required. If it is then an entry in the Past Information section of the database will be created to say that this bookie was offering these odds on this bet type at this time.

If these three pieces of data do not identify a bet type, new ones will be created in the Current Market Information and Past Information sections of the database. The opportunity finder will then notify the Opportunity Finder to look for opportunities involving this new type of bet. (See DFD "Opportunity Finder".)

If the best odds for this bet type are changed then the Opportunity Manager will be notified to update the data of the dependent opportunities. (See DFD "Opportunity Updater".)

Current Market Information Management

There will be two mechanisms at work under this heading that together ensure that data in the Current Market Information section of the database is up-to-date and relevant. Their specific purposes are:

1. To remove any possible bets that belong to the bookie that has just submitted but who's odds were not updated. This is because the odds can't be relied upon to be valid. (See DFD "Current Data Management".)
2. To remove any information that relates to an event that has passed its start timestamp (started). (See DFD "Post-Insertion Checks".)

Opportunity Manager

This section has control over inserting into and updating the Current Opportunity Information section of the database. This responsibility falls under two headings:

- Opportunity Finder – This will be notified when a new bet type is created. It will check the bet type against a list of opportunity templates to find if it features in any of them. If the bet type appears in a template a search is carried out for the other bet types required to create the opportunity. If found, the relative secondary data is entered into the database. (See DFD "Opportunity Finder".)
- Opportunity Updater – It is important to keep the secondary information in the database synchronised with the primary data. When a bet type's best odds are changed fields in both the Opportunities and Opportunity Bet List database tables will need updating. (See DFD "Opportunity Updater".)

The deletion of opportunities will be done by Current Market Information Management after the start of the event that it relates to.

The opportunity manager will have a list of opportunity definitions that describe a collection of bet styles and mutators that are required before a valid opportunity can be created. These opportunity definitions will often have a lot of common features that are the same apart from a variable or two. It makes sense to design a standard opportunity definition that all other definitions can inherit from and specialise.

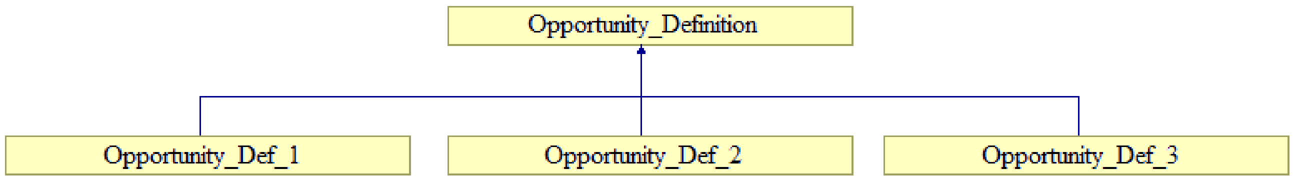


Figure 5.2 – Opportunity Definition Inheritance Hierarchy.

Data Retriever

(Due to time constraints this has not been designed as part of this project.)

Centralised Error Reporting

Error reporting will be done from a central location provided by a singleton pattern. The benefits of this are that it is accessible from anywhere in the module and that it reduces the occurrence of similar code. It also allows for the centralised generation of error messages that are displayed to the user and written to the log file at the same time.

Data Acquisition

This module will be a program in its own right that can run on the same computer as the Data Storage module or a different one, as long as has access to a copy of the communal files. Of course, in order to submit data to the Data Storage module from a remote location there will have to be a network connection too, but this will also be required for the module to run using “live” data.

Data Acquisition can be broken down into five sections, all of which are described below. It will conform to the MVC pattern of coding in that there will be a data model that contains all of the transient program information. This data model will be updated by all parts of the program and is the sole source of information that the data view displays data from. The main advantage of having this centralised data area is that it decouples the Data View from the rest of the program, allowing it to be swapped for another with minimal effort.

Acquisition Control

This is the part of the module that initialises the system components and sets them in motion. It will read the XML configuration file of miner information, set variables, create objects and have overall control of the module. Following program initialisation the acquisition control becomes the Control in the MVC pattern, allowing a user to interact with this module.

Miner Scheduler

The Miner Scheduler does exactly what one would expect, it has full control over which miners mine and when. In order to do this efficiently there will be numerous threads of operation working at any one time in a multiple producer, multiple consumer environment. This needs to be managed correctly to avoid the notorious “Race Conditions” or “Loss of Increment” on the shared variable(s).

There will be a producer thread for each of the miners instantiated (identical miner objects can be mining different URLs from the same bookmaker), which will place its miner into a list of waiting jobs every time its time period elapses. The consumer threads will do the actual data mining. They will take jobs from the job list and process them before coming back for another. The number of consumer threads will be user definable, and will need to be set with consideration of the processing power and network capacity of the computer that the Data Acquisition module is being run on, as will the time periods between successive calls to a miner.

Control View

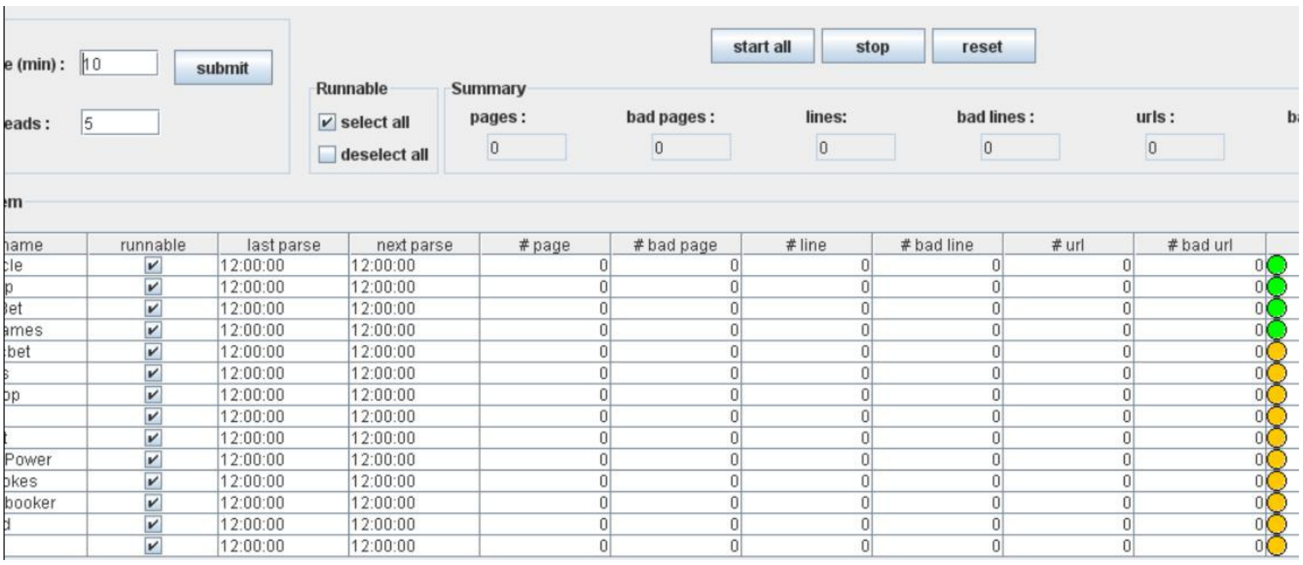
The view will display all of the information that is useful to the user and provide the facility by which the user can change what is happening. Facilities that this will offer are:
Display:

- Miner description (location of data being mined).
- Miner status (working, waiting, failed, etc).
- Miner stats (events processed, bets processed).
- Server connection details (host, port).
- Server connection status.
- Miner thread status'.
- Connection proxy information.

Control:

- Stop and start miners.
- Change the number of mining threads.
- Set proxy information.
- Change server details.

The view used by the customer's prototype performs almost all of these tasks and looks like a neat solution. The customer is also happy with this view, so it is the aim of this project to create something that resembles this in looks and function.



e (min) : 10

eads : 5

☑ select all
☐ deselect all

pages :	bad pages :	lines:	bad lines :	urls :
0	0	0	0	0

name	runnable	last parse	next parse	# page	# bad page	# line	# bad line	# url	# bad url	
cle	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟢
p	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟢
Bet	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟢
ames	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟢
bet	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟡
s	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟡
op	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟡
	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟡
t	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟡
Power	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟡
okes	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟡
booker	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟡
d	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟡
	✓	12:00:00	12:00:00	0	0	0	0	0	0	🟡

Figure 5.3 – Prototype Screenshot.

Miners

There will be XML miners and HTML miners and even though HTML is far more difficult to extract data from there are many similarities between these two types. There will therefore be a hierarchy of inheritance associated with the miners, involving characteristics common to both XML miners and HTML miners.

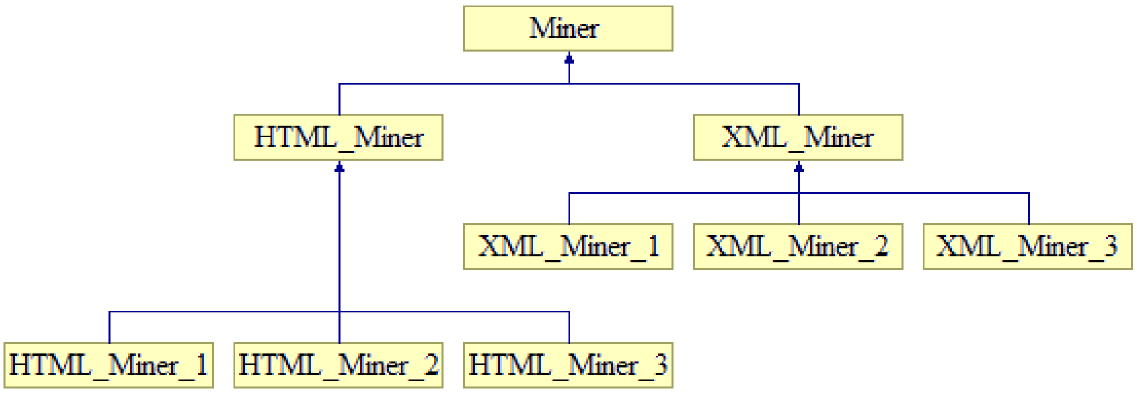


Figure 5.4 – Miner Inheritance Hierarchy.

Each miner is tailored to the specific requirements of the code that it is designed to mine, so they will have to be designed as they are implemented. No matter what the implementation is like, they will need to have features that are common to all miners. They need to:

- Extract event information from data:
  - Event start time and date.
  - Event participants.
  - Sport type. *
  - Event description (optional).

- Extract associated bet type information:
  - Bet style. *
  - Bet style mutator (for certain bet styles only).
  - Odds.
  - URL, so that a user can take advantage of the bet quickly.

- Create submittable objects with the data.
- Update statistics about the data parsed.
- Notify the data model as to their status.

*(Classified as a communal object constant value).

Miners also need to provide statistical feedback to the user as to their success (events found, useful events, possible bets found, useful possible bets).

Data Submitter

Data submission will work on a multiple producer, single consumer basis. There will be lots of miners producing submittable objects and adding them to a list of objects that are ready for submission. A single submission thread will be the consumer that takes these objects out of the list and sends them off to the Data Storage module. Again there are concurrency issues that require attention in order to ensure that none of the objects go missing.

Data View

The Data View Module of this project was not properly designed or implemented due to time constraints covered in the evaluation section of this report. Included below however is some design information about what would have been created.

Data Retrieval and Display

Below is a list of database query descriptions (based on information from the project specification) that this section will be based around.

- View all opportunities, sorted by profitability, for a single opportunity type.

- View all profitable opportunities, sorted by profitability.
- View all profitable opportunities, sorted by profitability, for a single opportunity type.

- View all sporting events, sorted by start time/date.
- View all sporting events, sorted by start time/date, for a single sport type.
- View all sporting events, sorted by first participant name, for a single sport type.
- View all sporting events, sorted by second participant name, for a single sport type.
- View all sporting events, sorted by event description, for a single sport type.

- View all opportunities on a single event, sorted by profitability.

- View all possible bets on a single opportunity, grouped by bet type, ordered by odds.

- View all possible bets on a single event, grouped by bet type, ordered by odds.

- View the possible bets and stake percentages that a user needs to take advantage of a single opportunity (odds in original retrieved incomparable state, not processed comparable state).

However this is implemented there would have been various view options from a "home" page. Upon selecting a higher level view (e.g. all events) there would have to be the facility to select a lower level view (e.g. all opportunities on a single event) and then maybe a lower level again (e.g. possible bets on an opportunity).

Alias Management Interface

This section needs to fulfil the basic requirement of being able to insert new participants into the database as this is not possible in any other way. It also needs to supply a method for managing the aliases collected. For these purposes the user needs to be able to:

- View already inserted participant information.
- View already inserted alias information.
- Insert a participant and (processed) alias version of the participant's primary name.
- Amend the participant id that aliases are related to.

There is no need to insert aliases into the database as all aliases that are mined are kept in the database even if they are not recognised as relating to any of the known participants.

Continuity of Service

As this will be a distributed system it is no good if when a network connection is interrupted an element of the system has to be manually reset/restarted, especially if these elements are a large distance apart. To minimise the need for this there will be contingency built into the system at all of the connecting stages:

- Database to Data Storage module.
- Data Storage module to Data Acquisition module.
- Data Acquisition module to The Internet.

Contingency will be based on the ability of one of the elements to detect the loss of connection and re-establish a connection via the same means by which it made the connection in the first place.

Java RMI

Following a brief look into Java RMI it is the opinion of this project that it will not be difficult to implement for the basic requirements of this project. There is no specific design associated with this, just the writing of both the Data Inserter and Data Retriever so that the RMI server can relay the methods calls to them.

6. Implementation

Software

As preferred programming language of the workforce it was decided early in the project development lifecycle that Java would be the preferred language for writing code as long as it was suitable for the purpose. When it became clear that remote connections between modules would be needed, the combination of Java's popularity and its RMI (Remote Method Invocation) facility made it an easy choice to make. Java was therefore selected to be used for the main bulk of the code (Database Controller, Data Inserter, Data Retriever, Opportunity Finder, Data Acquisition Controller, Data Miners and Data Acquisition View).

There are many common database packages successfully used today so the choice between them for this project was not initially a simple one. The following factors helped to reduce the list of possible solutions:

- The project workforce had experience with mySQL only.
- The customer would need to access the database manually in order to update certain reference data, so an included easy to use interface would be preferable. If this facility wasn't supplied by the used software then this project may have had to supply suitable software.
- The customer had experience of a piece of WAMP (Windows, Apache, mySQL, PHP) software called EasyPHP and was very happy with its performance.
- There had to be support for the database access in the language chosen for the Data Storage Module.

After researching similar WAMP software packages including Server2Go and Apache2Server it was this project's view that EasyPHP was quite a popular choice for projects of this scale and that there weren't any major benefits of any of its rival WAMPs. Due to the customer's preference and the fact that EasyPHP used software familiar to the project workforce (mySQL and PHP), it was decided that EasyPHP would be used in the project to supply the mySQL database software.

Using a WAMP like this has some added bonuses:

- The overall project design loaded minimal processing onto the display module. In fact the only processing comes from the areas of participant manipulation and the possible inclusion of a stake calculator, which are minimal. The display module therefore lends itself to the possibility of using Server Side Processing (like PHP) to produce HTML views of the data, and PHP is an integral part of the WAMP software.
- A graphical interface to access the database structure and data.

About halfway through the project (late Dec '08/early Jan '09) a problem was experienced with the EasyPHP WAMP software when it suddenly (and seemingly inexplicably) stopped functioning. Re-installation did not cure the problem so an alternative needed to be found.

XAMPP was selected as the replacement due to the following reasons:

- Like EasyPHP it used mySQL for its database.

- Like EasyPHP it used PHP for its server-side processing of web pages.
- It used the same admin interface to mySQL as EasyPHP, called phpMyAdmin.
- Installation of the software was quick and easy.
- The software looked very professional.
- Unlike Easy PHP (French), all documentation was in English including the website.
- The release was recent (Dec '08) and the impression was that it was well maintained and supported.

XAMPP was used without any problems for the remainder of the project and it is the opinion of this project that it is in fact a superior product to the original first choice of EasyPHP.

Communal Files

The communal data objects have been defined and are described below.

- BetSubmissionObject – Object that contains all data that makes up a bet.
- EventSubmissionObject – Object that contains all data that makes up an event.
- SubmissionObject – Object that contains an EventSubmissionObject and a Vector of BetSubmissionObjects that are related to the event.
- SAMConstants – Constants used by different (separated) parts of the system. Using these makes sure that different programs define certain important parameters (like bet style ID) the same, without (seemingly) random numbers in the code. These are currently defined for Sport Types, Bet Styles and Opportunity types but this will be expanded by the customer as they identify more and more opportunities that they would like to monitor.
- DBControllerInterface – this defines the methods that the Data Acquisition module can call on the Data Storage RMI Server.

Data Storage

This module was implemented, as designed, to handle submitted data and manage the current data in the database. An addition to the original design was a simple GUI, which is documented later in this section.

The potential concurrency issues concerning database access highlighted at the design stage are addressed using a read-write lock in the DataStorage class. I mention it here specifically as any future additions to the module (like implementation of a Data Retriever) should make use of this lock in order to guarantee the concurrent access to the database.

Submission Process

The submission procedure was originally based on the use of three methods; the first to request permission to submit, the second to submit and the third to notify of submission completion. Following this notification the system deleted all data for the bookie that had just submitted that didn't fall between the timestamps generated at the first and last method calls, thus getting rid of any data that hadn't just been updated.

This system was found to work but:

- It assumed that submissions were of large volumes of data.

- It assumed that submissions contained all of the data available for the bookie.
- It assumed that any data not updated was no longer useful.
- As RMI method calls are synchronous the actual submission call was having to wait until the data had all been processed before it returned and allowed the submitter to get on with submitting anything else.

This was found to be unworkable when a parser was written for Pinnacle, who only deliver information about the markets that have changed since a data miner's last data request, not a full listing of odds every time. Problems were also highlighted when the first HTML miner was written and it was appreciated just how much slower these operate, due to the amount of data they need to download compared to an XML miner. In a system where accurate data is required, it was not satisfactory to wait minutes between retrieving data and submitting it.

There were radical changes implemented in order to maximize the efficiency of the submission process:

- The idea of blanket purging of all old data was abandoned and a new data persistence was built in that was focussed on the individual odds as apposed to the bookmaker. This allowed market data to stay in the system for a maximum pre-defined time (30 minutes say), after which it was deleted.
- Submission now only requires the data to be sent. The authority and notification calls were abandoned as they were surplus to requirement with respect to the above.
- Processing of data was given a separate thread of execution and a buffer of data was established so that the submitter no longer had to wait for data to be processed, just for it to be placed into the buffer. The submitter could then get on with the next submission whilst the new processing thread got on with processing the data.

Database

The database was implemented true to the design in Appendix D, but the design has evolved slightly as the process has progressed. This is mainly due to the customer's enthusiasm for the improvements to the specification that this project suggested.

One such change is that there is now a Stake Percentage column in the Opportunity Bet List table that advises the user what percentage of their stake needs to be bet on the associated Bet Type. Originally this system was meant to advise the user of where the opportunity lay, and the taking advantage of it was the user's responsibility. As the customer investigated different kinds of opportunities that the system could find, the customer wanted to implement more and more complex combinations of bets. So complex in fact that it would be difficult for a user to use the data to take advantage of an opportunity without the extra information that the new column holds.

Database Control

This is the section of the code that contains all of the commands that access the database. It consists of a Java Interface (DatabaseInterface) that defines all of the generic methods needed to perform the necessary operations on the database. There are then concrete implementations of these methods in the SQLController class that implements this interface. As mySQL has been chosen as the database software, this class connects to the project database on the mySQL server (started by the WAMP software), via windows data sources ODBC (Open Database Connectivity). The object then issues the SQL commands required to achieve the method's purpose.

Writing the database access in this way allows the database software to be changed at a later date with minimal change to the existing code. The only change would be referencing the new database access object class (that would need writing) that inherits from the interface, which would be a single line of code. All of the method calls would then go to the new database controller.

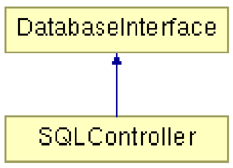


Figure 6.1 – Database Control Inheritance Hierarchy.

The constructor of the SQLController takes string arguments describing the database name, access username and password. This means the database name and the access information are defined at runtime, allowing the user to easily change the database that the system uses. This will be very useful for maintenance purposes, allowing the "real" database to be kept safe while a new data miner is being tested on a dummy database.

There were teething problems with the database connection due to there being a compatibility issues between jdk (java development kit), JDBC (Java Database Connectivity) driver, ODBC interface and the Data Storage Module. The problem was that the main development computer was 64bit and there was a mismatch between the software versions so they weren't communicating. Eventually the source of this issue was recognised and it was decided that 32bit versions would be used across the board, which meant using the 32bit ODBC interface that 64bit versions of Windows Vista hide away from users.

Built into the database control section is a mechanism for reconnecting to the database if the connection is dropped. This also displays a notification that there is a problem. As there is often a high rate of database access calls it was necessary to limit these notifications to stop the user from being swamped by notifications, making the program unusable.

If the database connection is lost then the efforts to reconnect would take a little time, and as an attempt to reconnect happens with every new submission that arrives, there is potential for the buffer that the submissions are entered into getting very big. To combat this a limit has been set at 100,000 (100 bookmakers supplying 1000 submissions each) that once reached starts deleting the oldest submission item every time a new one enters the buffer. This has the added bonus of allowing the database connection to be lost temporarily with no/little loss of new data. There is a readout of the current buffer size on the GUI and there are user notifications and a log file entries generated to notify the user that the buffer maximum has being reached.

Data Inserter

The Data Inserter part of the Data Storage module was implemented fairly true to the original design. Some of the changes that were documented at the start of this Data Storage section were relating to data insertion and below are a few other minor change that were required.

Event Identification

There is no longer the requirement for the time of an event to match the time of the event that is trying to match it. This is due to mistakes by bookmakers causing multiple (non-comparable) events being inserted into the database for the same participants on the same day. This caused potential opportunities to be overlooked because odds were attached to seemingly different events. Now, as long as the participants and day of the event matches, two sets of event data are deemed to be the same, and the time is taken from the data set that was used to insert the event into the database, whether it was right or wrong.

It is possible that this will very occasionally cause an "in progress" event, which is normally wiped from the database, to persist. This may get updated with new odds even when the score of the event has changed and these may be compared with odds collected before the event started. This would result in spurious opportunities being highlighted but as soon as a user tried to take advantage of them it would be obvious from the bookmakers' websites that the event was underway.

This is currently an accepted "issue" that on the other side of the argument could produce a lot of useful opportunities that would otherwise be overlooked.

String Matching

Although two levels of string matching were specified in the design and two methods have been implemented there is only one matching algorithm with the "basic" match internally calling the "good" match method. It is a very simple algorithm with the following steps:

1. "?" Characters are removed from both of the strings.
2. Both strings' lengths are checked and a failure is returned if either is less than a pre-defined length. This makes sure there is enough content in the string to compare.
3. A test is then carried out to see if all of the characters of the smaller string feature in the longer string in the same order, and acceptance of a match is only issued if this is true.

String matching could be seen to be an area of research all on it's own, so this project made the decision to stick with the simple solution above. As the program never creates new participants and only accepts new aliases for participants when there is little doubt that their identity is the same (aliases fit an already defined event between two participants) the string matching does not need to be too clever. If it were being used to define new events between unknown aliases then it would need a lot more time spent developing it, but as opportunities will never occur unless multiple bookies supply odds for the same event, if the event hasn't already been defined there is nothing to gain by inserting the odds without near certain participant identification.

Possible Bet Entry/Update

In addition to the original design specification, this section also makes the odds comparable by changing them if the bookie takes a "cut" of any winnings. This translation must be "undone" by a potential data view or else a user will not be able to find the value of the odds that are accredited to the bookmaker concerned.

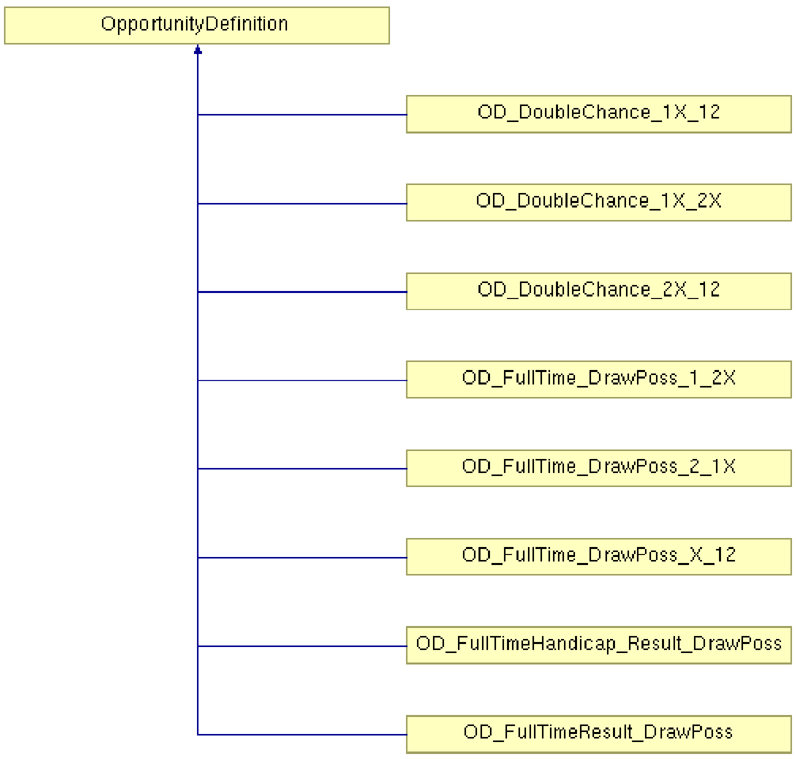
During testing, this part of the module reported many errors in the submitted data. As a result the checking of data and the error reporting for the whole of this module has evolved and become far more robust.

If an event is identified as one in the database but with the participants transposed it is this section of the code that processes the associated bet data to make it valid before its insertion into the database. Bet style id (integer) constants are arranged so that the inverse can be found (away win if the original is home win) by adding one to even ids and subtracting one from odd ids, but this only happens if there is a valid entry in the database for that BetStyleID (there is no inverse for a draw for example). This arrangement allows the odds of a transposed event to be inserted into the database correctly.

Opportunity Manager

The opportunity manager has undergone no conceptual changes since its design, although it has evolved since it was first implemented. Originally, new opportunity definitions had to be registered with the opportunity manager before they could be used. Now, when the module is initialised it searches for opportunity definitions, which means they can be included just by adding the compiled code to the module's working directory. This means that there is no need to re-compile the module code and that including a new opportunity definition is a single step process).

There have been eight opportunity types implemented, using nine different bet types. These do have a lot of inherited code from the OpportunityDefinition superclass, which makes it very easy to write simple new opportunities. It is also possible to write complicated definitions including uneven weightings to the bet types that are required (as the user has stressed is needed). Please see the Opportunity Definition Manual in Appendix M for a full description of how to create new definitions.


OpportunityDefinition
OD_DoubleChance_1X_12
OD_DoubleChance_1X_2X
OD_DoubleChance_2X_12
OD_FullTime_DrawPoss_1_2X
OD_FullTime_DrawPoss_2_1X
OD_FullTime_DrawPoss_X_12
OD_FullTimeHandicap_Result_DrawPoss
OD_FullTimeResult_DrawPoss
Figure 6.2 – Opportunity Definition Inheritance Hierarchy.

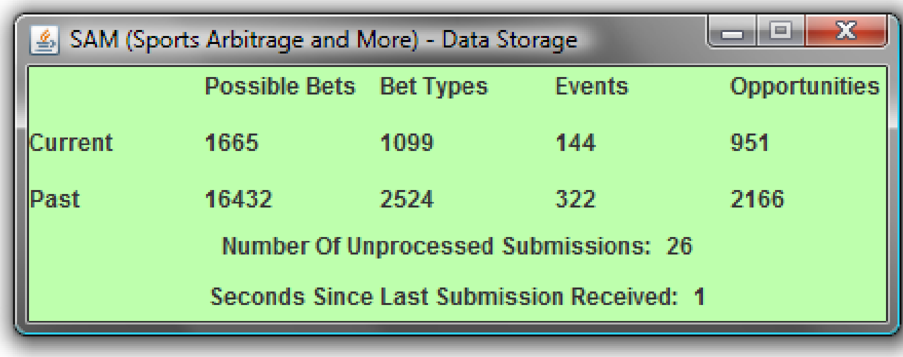
Data Retriever

This module was not implemented due to time constraints.

GUI

It was decided late on that there simply had to be some sort of display for the Data Storage module. If there wasn't, a user would have to either monitor the WAMP database interface to check that the module was working (by way of assumption due to there being data in the database), or look at the output window of an IDE that the program was being run from. Of course this second choice would force the user to run an IDE every time that they wanted to use the program, which is highly undesirable.

A simple GUI was designed that allows a user to quickly and easily ascertain whether the module is functioning and what the overall status of the database is. It displays the number of current and past possible bets, bet types, events and opportunities. It is important to display "current" data levels to show what the program's current condition is, and past data as this needs to monitored so that it doesn't get too big (as the customer specified that no past data deletion mechanism was required). There is also a readout of how many submissions have been received without being processed and also an indication of how long ago the last submission was received. The whole interface changes colour from green to red to indicate database connection problems.


SAM (Sports Arbitrage and More) - Data Storage

	Possible Bets	Bet Types	Events	Opportunities
Current	1665	1099	144	951
Past	16432	2524	322	2166

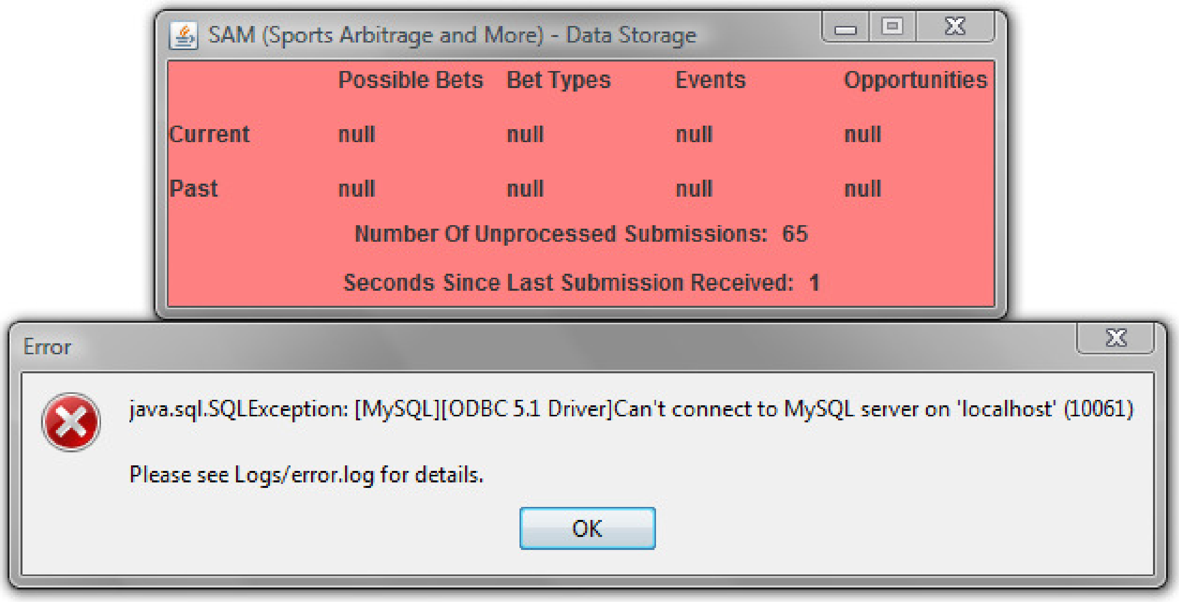
Number Of Unprocessed Submissions: 26

Seconds Since Last Submission Received: 1

Figure 6.3 – Data Storage Module GUI.

Centralised Error Reporting

Along with the GUI, the centralised error reporting (that was implemented as designed) helps to keep the user informed about the module status. The user is notified of problems using dialogue boxes on the screen (figure 6.4) and notifications inserted into a log file, with a reference time and date of when the problem occurred and often some explanation as to why.


SAM (Sports Arbitrage and More) - Data Storage

	Possible Bets	Bet Types	Events	Opportunities
Current	null	null	null	null
Past	null	null	null	null

Number Of Unprocessed Submissions: 65

Seconds Since Last Submission Received: 1

Error

```
java.sql.SQLException: [MySQL][ODBC 5.1 Driver]Can't connect to MySQL server on 'localhost' (10061)

Please see Logs/error.log for details.
```

Figure 6.4 – Data Storage Module GUI with Error Message.

Data Acquisition

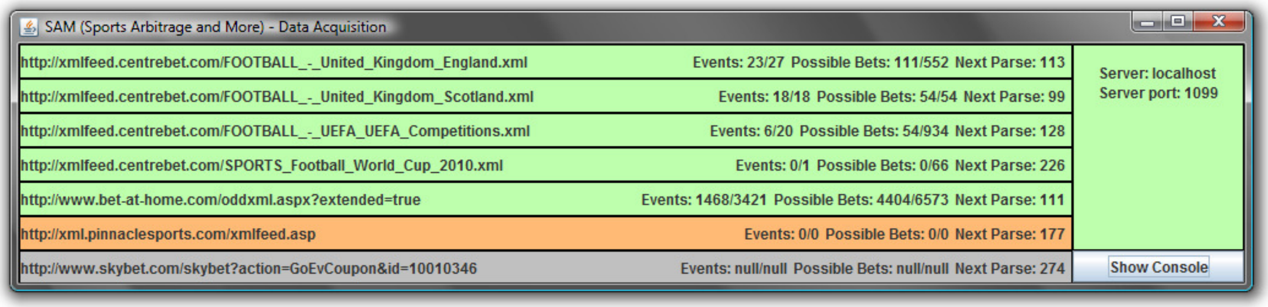
GUI Simplification

It was fairly quickly appreciated, following the Christmas break and exam period, just how little time there was to progress this project in the way it was originally planned. It was decided that the Data Acquisition module would be implemented in a simpler way than it had first been designed, with the intention of reviewing this once a fully functional system had been realised.

It was known from experience that graphical user interfaces take up a lot of programming time, often striving for perfection as little problems can annoy users greatly. It was decided that this part of the Data Acquisition module was the least important to the functionality of the project and therefore the part that would be curtailed.

The control of data acquisition was deemed to be a luxury function as all of the same control could be implemented in a simpler, non-interactive way. The XML configuration file's role was expanded to take up as much of this functionality as possible and give greater control over the initial setup of the module. It is the view of this project that no functionality, just convenience has been sacrificed here.

The view was implemented to be easy to see how the miners (and submission manager) were progressing. By colour coding each element a monitor could see that everything was functioning correctly in just a few seconds. All of the monitoring information can be seen on the GUI representation of the component and a countdown to the next time the miner will mine has been included too. A point to note is that the bet-at-home miner displays such a large number of events mined as the XML file redefines the event for every set of odds that are given.


SAM (Sports Arbitrage and More) - Data Acquisition

```
http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_England.xml      Events: 23/27  Possible Bets: 111/552  Next Parse: 113
http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_Scotland.xml     Events: 18/18  Possible Bets: 54/54  Next Parse: 99
http://xmlfeed.centrebet.com/FOOTBALL_-_UEFA_UEFA_Competitions.xml      Events: 6/20  Possible Bets: 54/934  Next Parse: 128
http://xmlfeed.centrebet.com/SPORTS_Football_World_Cup_2010.xml         Events: 0/1  Possible Bets: 0/66  Next Parse: 226
http://www.bet-at-home.com/oddxml.aspx?extended=true                    Events: 1468/3421  Possible Bets: 4404/6573  Next Parse: 111
http://xml.pinnaclesports.com/xmlfeed.asp                               Events: 0/0  Possible Bets: 0/0  Next Parse: 177
http://www.skybet.com/skybet?action=GoEvCoupon&id=10010346              Events: null/null  Possible Bets: null/null  Next Parse: 274
```

Server: localhost
Server port: 1099

Figure 6.5 – Data Acquisition Module GUI (console hidden).

The view is refreshed on a timer (currently every 200ms) so that information from the data model makes its way to the user fairly quickly. This could have been implemented so that it was updated whenever the data changed but this could have been twenty times a second (or more) and there was no need for such accuracy.

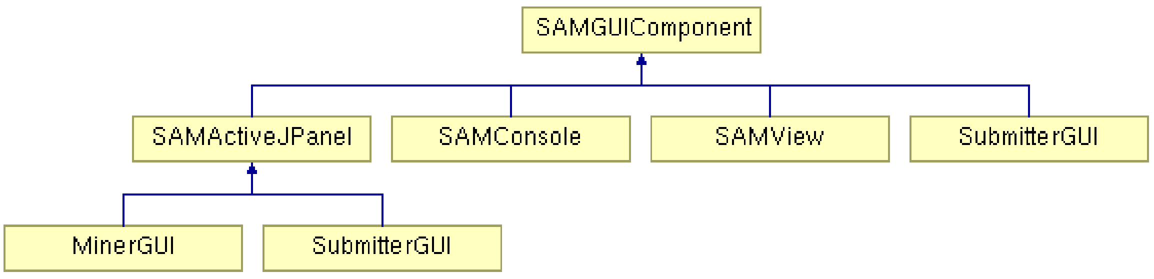


Figure 6.6 – Data Acquisition GUI Inheritance Hierarchy.

The components that have a status and change colour all inherit from SAMActiveJPanel class, so the code for this is only written once.

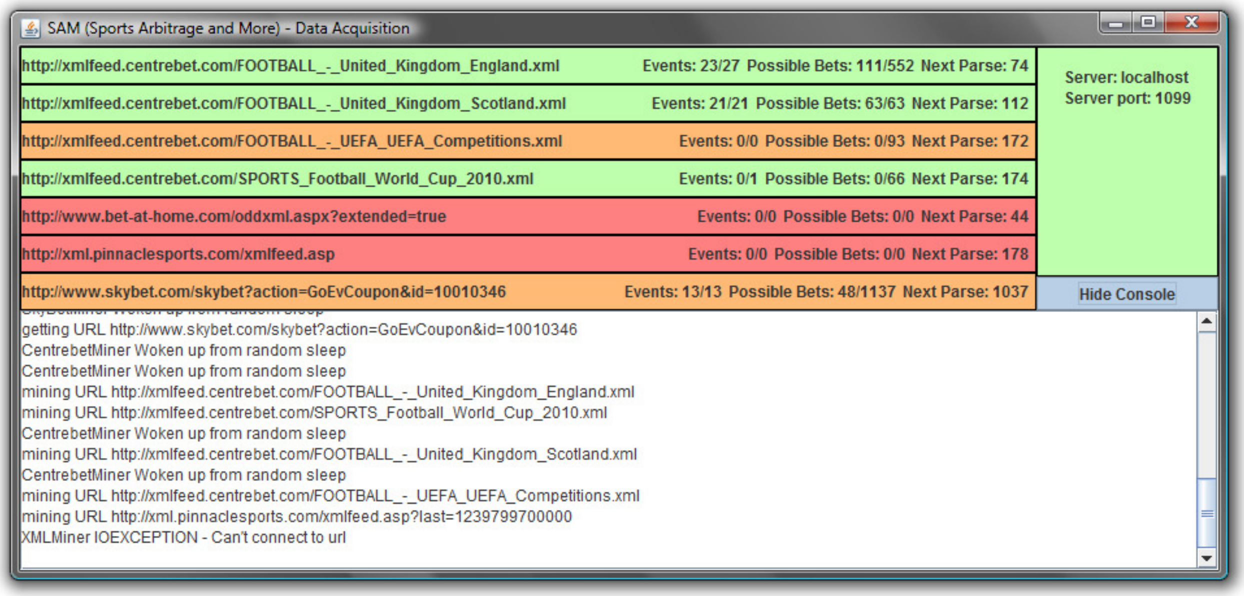
As the GUI components all implement the SAMGUIComponent interface their update methods are recursively called from the top level container, allowing them all to update themselves with a single call to the update method of the top level component. This is achieved with the following code (in SAMView).

```
        @Override
        public void update(){
updateRecursive(this.getContentPane().getComponents());
        }


        private void updateRecursive(Component[] comps){
                for (int i=0; i<comps.length; ++i){
                        if (comps[i] instanceof SAMGUIComponent)
((SAMGUIComponent)comps[i]).update();

```

```
                              else if (comps[i] instanceof Container)
updateRecursive(((Container)comps[i]).getComponents());
                      }
```

Figure 6.7 – Data Acquisition GUI Recursive Update code.

There is also a drop-down console which displays lots of progress information. If one of the miner components turned red (say) there would be a reason displayed here along with the build up events that may have leads to its failure. As this is drop-down it is easy to hide this detailed view and just use the simpler colour coded part of the display.



```
http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_England.xml     Events: 23/27  Possible Bets: 111/552  Next Parse: 74
http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_Scotland.xml    Events: 21/21  Possible Bets: 63/63  Next Parse: 112
http://xmlfeed.centrebet.com/FOOTBALL_-_UEFA_UEFA_Competitions.xml     Events: 0/0  Possible Bets: 0/93  Next Parse: 172
http://xmlfeed.centrebet.com/SPORTS_Football_World_Cup_2010.xml        Events: 0/1  Possible Bets: 0/66  Next Parse: 174
http://www.bet-at-home.com/oddxml.aspx?extended=true                   Events: 0/0  Possible Bets: 0/0  Next Parse: 44
http://xml.pinnaclesports.com/xmlfeed.asp                              Events: 0/0  Possible Bets: 0/0  Next Parse: 178
http://www.skybet.com/skybet?action=GoEvCoupon&id=10010346             Events: 13/13  Possible Bets: 48/1137  Next Parse: 1037

Server: localhost
Server port: 1099
```

```
getting URL http://www.skybet.com/skybet?action=GoEvCoupon&id=10010346
CentrebetMiner Woken up from random sleep
CentrebetMiner Woken up from random sleep
mining URL http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_England.xml
mining URL http://xmlfeed.centrebet.com/SPORTS_Football_World_Cup_2010.xml
CentrebetMiner Woken up from random sleep
mining URL http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_Scotland.xml
CentrebetMiner Woken up from random sleep
mining URL http://xmlfeed.centrebet.com/FOOTBALL_-_UEFA_UEFA_Competitions.xml
mining URL http://xml.pinnaclesports.com/xmlfeed.asp?last=1239799700000
XMLMiner IOEXCEPTION - Can't connect to url
```

Figure 6.8 – Data Acquisition Module GUI (console showing).

The configuration file has a section for system configuration as well as the original miner information section. This includes variables for the number parsing threads, proxy and server information as were in the original GUI specification.

Acquisition Control/Configuration File

Following on from the points raised above, this part of the module which was originally the Control in the MVC pattern, now reads the configuration file, validates its contents and sets up the Data Acquisition module to perform as the user requires. Following this it has no further function.

The configuration file (example at Appendix J) contains the following information:

- Connection
  - Server address.
  - Server port.
  - RMI interface name.
  - Proxy host.
  - Proxy port.

- Configuration
    - Number of mining threads.
    - Number of failures before a miner is reset.
    - Whether output is to log file or the server specified above.
- Miner Definitions
    - Miner class to instantiate.
    - URL to direct miner to.
    - The sport type of the URL (only required if sport specific url).
    - Period between mining attempts (in secs).
    - How many periods between full data refreshes are required (only if required).*

*(Some miners set the URL for their next mine as they go. This allows the user to specify how often the miner must return o the above URL.)

Miner Scheduler

The designed method for miner scheduling worked well and has not been changed. All miners starting from the same starting time, however, led to bursts of activity followed by large periods of inactivity. To try to even out the activity levels the user can define varying miner periods of operation (via the configuration file) and there is a random start to all of the miners that sets each one going at a (pseudo-) random point in their period.

Data Miners

Four data miners have been implemented. Three XML miners were initially created by making bookmaker specific handlers to work with a generic XML miner. An HTML miner followed and at this point it was appreciated what common features the two types of data retrieval had, so the XML handlers were re-written as miners in their own right. Both miner types now inherit from a common superclass and act as superclass to bookmaker specific subclasses. Both types follow an event-driven pattern that calls methods when "interesting" code is found by the parser.

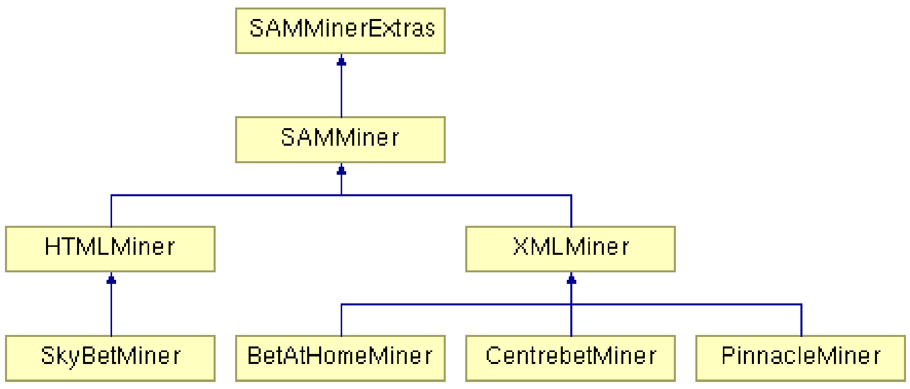


Figure 6.9 – Miner Inheritance Hierarchy.

In the case of the XML miners the inherited code parses the data and calls methods on the subclass when predefined interesting tags are found. This makes the bookmaker specific sub-classes hold minimal code and it is this project's opinion that it couldn't be much easier to create a new XML miner.

HTML miners are different as they parse the data themselves, using generic tools that generate the events that the miner must handle. This offers more freedom to tailor the miner to the bookmaker's website but it means more code is needed to create a miner. Part of this is due to HTML mining being about a whole website as apposed to just a page of HTML. In the one HTML miner that was created the required data was spread between a "master" page and many event specific pages who's URLs were found on the master page. Other websites distribute their information in slightly different ways and this kind of uncertainty means that there must be a less constricted framework to write the code for a new miner in. Unfortunately this means a higher degree of difficulty in writing a new HTML miner.

SAX Parser Problem

There were two utilities included in the standard java development kit (jdk) to help parse xml code; SAX and DOM. DOM creates a tree like structure in the memory and holds all of the data for programmers to use as they please, where as SAX navigates through the code returning information when it hits start tags, end tags and the characters in between. The SAX parser was chosen for use by this project as it is far more memory efficient, especially for large XML files.

The problem that the title of this section refers to is that the SAXParser Parse method has a tendency to trap the thread of execution if the network connection is lost before the method returns, certainly on Windows Vista 64bit anyway (the problem could not reproduce on Windows XP 32bit). Even interrupting the thread did not have any effect. In order to maintain the specified number of parsing threads and to not lose the miner that made the offending parse method call, a work-around was required.

The user can define the number of times that a miner can be asked to start and fail before the work-around is initiated. If this number is exceeded (whether due to failure or a too short period being set), the miner is deemed to have failed. In this case it is the mining thread that is trying to initialise the already mining miner that takes the action. It interrupts the trapped thread (just in case it does respond at some point) and also the thread that schedules that miner before creating a new instance of the miner, its scheduling thread and a new mining thread to replace the trapped and dispensed with objects. This effectively sets adrift the mining thread and miner, replacing them with identical copies.

This is a good plan but it relies on the user specifying that there is more than one mining thread as the mining threads "save" each other. Furthermore it assumes that all of the mining threads will not get trapped at the same time. To this end, a second mechanism was implemented. If the queue of miners ready for mining becomes more than the number of miners instantiated (the same miner in the list more than once) the whole system is deemed to have stopped responding. In this case it is one of the threads that schedules mining jobs that becomes the saving thread, resetting all miners, scheduler threads and miner threads. It then tells itself to (effectively) die as soon as everything is reset.

Data Submission

Data can be submitted either to the designated Data Storage module (via an RMI server) or to a log file in the working directory of the Data Acquisition module. Selection of an output target is done very easily in the configuration file.

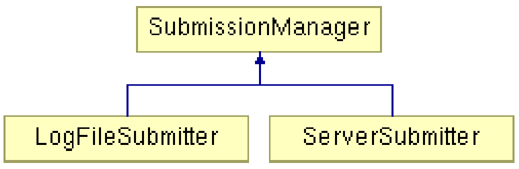


Figure 6.10 – Data Submission Inheritance Hierarchy.

Having the ability to output to a file allows a programmer to test new miners in an enclosed environment so that their performance can be easily evaluated and improved. A miner should only be allowed to submit to a live Data Storage module when it has been tested thoroughly and this offers a quick and easy local way to check the status of the data that the miner is creating.

Here the actual submission is done by a separate submission thread of execution, from a buffer that is fed by the numerous miners in the SubmissionManager class. As with other code in this project, this is written with concurrency issues in mind. The buffer also has a maximum size where (once reached) the oldest submission item is removed to make way for a more recent one. If the maximum buffer size is reached a message is output to the GUI Console. If this occurs there must've been a reason for it (like connection to Data Storage lost), which would have resulted in an error status and a change of colour to a GUI component. As the user will already be aware of a problem there is no other specific user notification (like a dialogue box for example) of buffer overload.

If the connection to the Data Storage module is lost efforts are made to re-establish the connection with every item in the buffer that this part of the program tries to submit.

Data View

The data view part of this project was not properly designed or implemented due to time constraints covered in the evaluation section of this report. Included below however is information on the work that was carried out.

Data Retrieval and Display

The mySQL code for the queries specified in the design section are listed in Appendix F. They can be used with the WAMP phpMyAdmin interface to find the data that would have been available in a Data View produced in this project, although using them is a little cumbersome. If a project in the future wanted to expand on this one, these queries would be very helpful.

Alias Management Interface

The original plan was for alias management to be done via the Data View interface. As there was no data view realised in this project and no time to implement an alternative complete solution other ways of meeting the requirements of the design have been found.

Three of the four requirements laid out in the design section can be achieved using the phpMyAdmin interface of the WAMP software. These are:

- View already inserted participant information. (View the participants table.)
- View already inserted alias information. (View the alias_list table.)
- Amend the participant id that aliases are related to. (Amend records in the alias_list table.)

The other requirement (Insert a participant and (processed) alias version of the participant's primary name) has been addressed individually. If it weren't addressed, there would be no way to enter participants into the database without trying to manually imitate the processing of an alias, which could introduce dangerous inconsistencies.

As this was not a part of the original design and could be seen as a temporary fix for this important lack of functionality, access to it is very discreet. A small interface has been written that is accessible from the GUI of the Data Storage module by clicking the tiny button in the bottom right hand corner (see below).

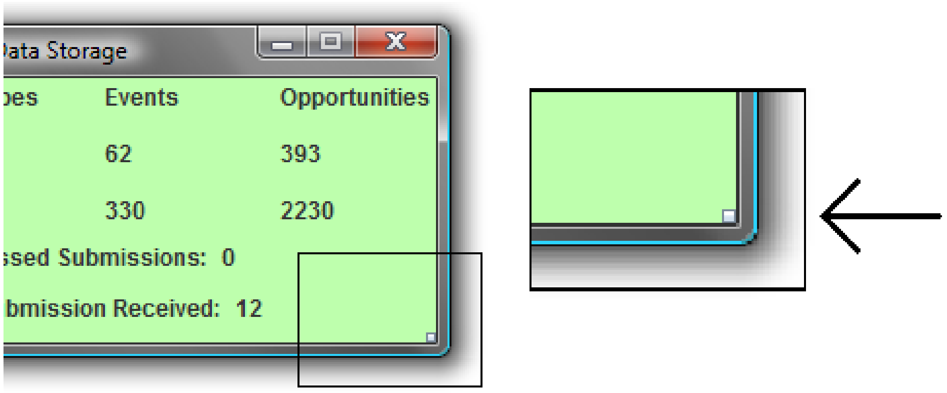


Figure 6.11 – Participant Inserter Interface Button.

It is a fairly crude interface and time certainly hasn't been spent making it look pretty, but it works. Functionally it is sound, as it checks the database for duplicate entries, asks the user for confirmation if the alias is already in the database and offers confirmation about what has been done. This is the only way to input new participants into the database and keep the database consistent. Even though minimal time has been spent on it's visual design, for this reason it had to be included.

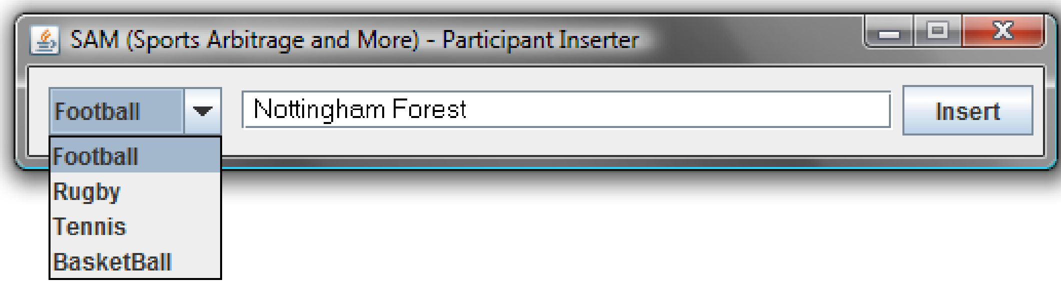


Figure 6.12 – Participant Inserter GUI.

7. Testing

Overview

The SAM system has not undergone the same kind of pre-conceived rigorous testing schedule that a highly interactive program would require. The reason for this is the highly limited method of input that is used on the system. Highly interactive user interfaces require testing schedules that model a high proportion of the many inputs that a user could possibly make, but SAM doesn't have interactive input. The predictability of the inputs makes SAM very easy to test by comparison.

The Data Storage module of SAM receives inputs from the Data Acquisition module and from the database in the WAMP software upon request. Both the data in the database and the Data Acquisition module are completely controlled by this project so data values are predictable. Even the errors that can be generated by the database access are predictable.

The Data Acquisition module has control over its inputs as they are the result of controlled mining actions, the only other "inputs" there can be are errors due to bad URLs etc. Every piece of data mined is validated to make sure the it has the correct characteristics (data type etc), so exceptions caused by errors here are predictable too.

In general, the testing technique used here was to test each individual part of the modules in isolation, the complete modules and then the whole system. This could be described as Unit Testing followed by Integration Testing for the separate modules then System Testing. All tests were initially carried out with test data that was injected into the necessary methods. The first system tests were carried out like this too before the first bookmaker's miner was created and a saved XML page was used repeatedly as "real" but not "live" test data. The next stage was "live" testing when the miner actually downloaded data, parsed it and fed it into the system. Data checks were then carried out to make sure that the data in the database was a true likeness of the online data.

This approach was taken to first show that the methods/objects performed as they were supposed to, then to incrementally include more and more of the module before the two modules were integrated. This methodical approach allowed the individual sections of code to be virtually bug free by the time they were used in conjunction with too many other sections, allowing ease of fault tracing. The bugs that were found after the unit tests were usually due to the integration of separately coded units.

It was decided quite early on that writing many test cases was not a productive use of time. As inputs were fairly "predictable" (as discussed earlier) it was far easier and probably more productive to use live data to test with, once there was a certain level of confidence in the quality of the code. This could be seen as automatic live testing as it only required a tester to look at the log file when and if a problem was found.

Some issues that could only be identified by testing in an operational environment were:

- Internet connection speed leading to maximum number of miners/minimum period of re-mine.
- Processing capacity limits.

- Concurrency issues that would not show without such an abundance of test data.
- Overloading of database – just how quickly does the unmanaged "past data" grow.

Unit Tests

Unit tests were undertaken during the implementation stage of the project, during and following the writing of objects and methods. Basic outlines of these were documented but have been moved to the appendix section (Appendix G) as they were a little tedious to read and they would have made this section a lot larger than it needed to be.

Module Tests

Data Storage

This module was tested in conjunction with its RMI interface with the use of a dummy miner program that submitted contrived submission objects designed to mimic complete real data. This program ran on the same computer as the Data Storage, then it was transplanted to another computer (within the same LAN). In both sets of tests the Data Storage module performed well, as expected, with new opportunities being inserted into the database at the right times and data being managed correctly. There is an example test schedule in Appendix H to show the type of testing that was undertaken here.

The participant inserter was a late addition to this module but has been tested with a good range of valid and invalid data. This includes data values that are technically "valid" but should not be inserted into the database due to repetition or ambiguity.

Data Acquisition

The testing of this module started with the injected data, moved on to real data stored locally and then on to live data from The Internet. The main aim was to test the basic framework rather than the individual data miner code, although in later tests a live mining was required. All outputs were sent to the log file and examined for consistency with the data that was used for input.

In all tests the required submission objects were created when they were supposed to be, correctly filled with the relevant data then successfully submitted or destroyed if they contained insufficient data to define anything useful. During some tests intermediate data was examined using output from the SubObjBuilder class (submission object builder).

System Tests

Testing for the system followed a similar progression to the module testing of the Data Acquisition module; the input data getting more "live" with every test and the outputs (in the database) being evaluated against the input data for consistency.

As the module tests didn't highlight any notable errors it was no surprise that when the system was ran as a whole the contrived data didn't expose any errors either. It was the following set of tests that exposed a few interesting issues.

This next test involved the repeated mining of the same xml file that was stored locally. Data miners build data objects using a SubObjBuilder (submission object builder) object, which was not being refreshed in between mining operations. This meant that the second time that the miner parsed the file it submitted the new and old data (exactly the same data) twice. As this was received by the Data Storage module at the same time (due to it working with the old 3-step submission process that submitted all bookie data at the same time) it meant that the receive timestamps were identical too. This meant that two identical sets of data were being inserted into the database, which had the same primary keys and therefore threw database exceptions for duplicate entries. It is interesting to note that this error may not have been picked up (at least not in the same way) with the new (superior) single step submission process.

Live mining using a single bookmaker's miner was the next testing hurdle that the system needed to get over and from a data perspective there were no additional major bugs to report. A loss of internet connection did however highlight the issue that was documented in the implementation section under the heading "Sax Parser Problem". Please read the "Continuity of Service" section below for further details on this.

The testing stage of running multiple live miners was the final stage of the system tests to be implemented. Following separate integration stage testing of the new miners, they were specified in the configuration file and all called to run "live" together for the first time. These tests highlighted the difference in mining speed of the XML miners (that visit a single URL) and an HTML miner that (that can visit tens or even hundreds). This exposed the vital need to set well thought out mining periods for HTML miners, as too short periods might either use all of the available network bandwidth or the module might reset the miner due to it classifying the miner as unresponsive.

Some errors that were found and fixed here were:

- All miners ran once at module initialisation due to the same list being used as the queue of waiting miners as was used during the instantiation of the miner objects. This was unsatisfactory as they were supposed to observe a random sleep period.
- It became evident that once a certain bet type with mutator had been entered into the database it was possible to enter another one without a mutator and it was regarded as the same type as checking only looked at types and should have been looking for the combination of bet type and mutator.
- Certain errors on the Data Storage module caused continuous repetitive notification to the user. This was changed so that the user was only notified once then not again until the error condition was rectified and then it reappeared again some time later.
- Initially all errors that notified the user by dialogue box had to wait for the user to respond before the thread could be freed to perform other tasks. This was found to be highly undesirable, especially when the error condition could have been fixed by later operations had the thread of operation been freed. The error reporting evolved to have the capability to highlight an error to the user and wait or highlight to the user and continue depending on the nature of the error.
- Certain data was being seemingly mixed up by the miners. It transpired that superclass variables that were only supposed to be a reference were being changed by a certain miner, so that all the other miners now operated from the changed (incorrect) references.

- The submitter part of the data acquisition GUI hardly ever showed up red (failed) even when not working as it would turn orange (processing) too quickly to see. The “processing” status was removed so that it showed only red (failed) or green (ok).
- Some HTML odds were attributed to the wrong bet types as each line of code was tested by the regular expressions (used for identifying data) only until one of them returned. The HTMLPageProcessor class now tests every line against every regular expression.
- One American miner’s event start times were all out by an hour after the change from GMT to BST. This was repaired in the miner in question but the issue of daylight saving hours needs to be considered for all new miners.
- Database access errors appeared after the data storage module was closed then open again whilst a Data Acquisition module was submitting. The closing operation now needs to acquire the database write lock prior to closing down to ensure that any data input or management is complete.

Data Checks

This involved the manual checking of database data against the data on the websites/in the XML files. As there was no Data View to facilitate this the WAMP software database access program phpMyAdmin was used with the following query:

```
SELECT          bookies.Bookie_Name,
                participants.Primary_name,
                participants2.Primary_name,
                events.Event_Start_Timestamp,
                bet_Styles.Bet_Description,
                bet_types.Bet_Style_Mutator,
                possible_bets.Odds
FROM  possible_bets
        INNER JOIN bet_types
                ON possible_bets.Bet_Type_ID = bet_types.Bet_Type_ID
        INNER JOIN events
                ON bet_types.Event_ID = events.Event_ID
        INNER JOIN participants
                ON events.Participant_ID_1 = participants.Participant_ID
        INNER JOIN participants participants2
                ON events.Participant_ID_2 = participants2.Participant_ID
        INNER JOIN bet_styles
                ON bet_styles.Bet_Style_ID = bet_types.Bet_Style_ID
        INNER JOIN bookies
                ON bookies.Bookie_ID = possible_bets.Bookie_ID
```

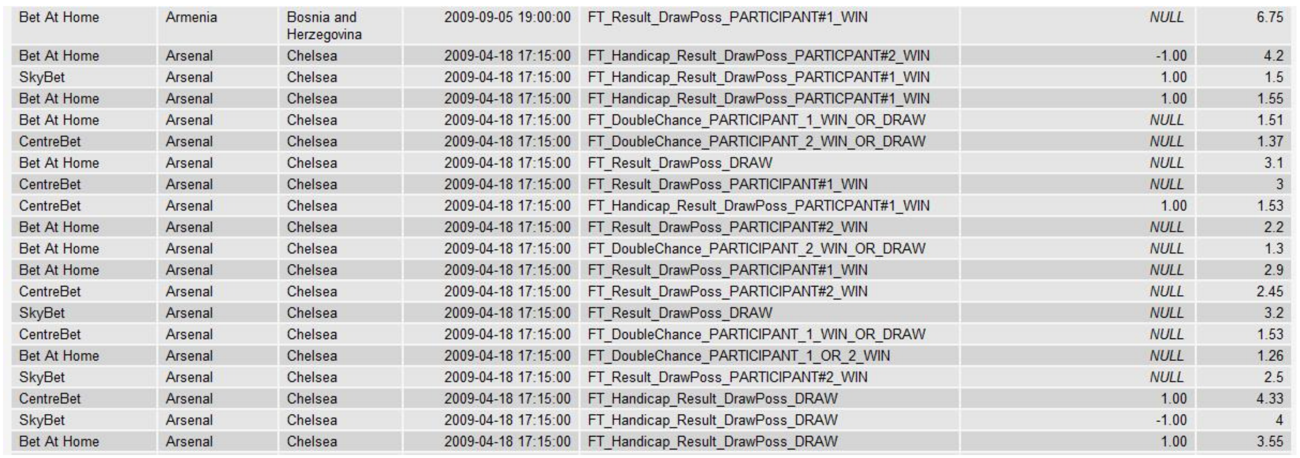
Figure 7.1 – Data Check SQL Query.

This query resulted in a page being displayed with the following data:

- Bookie’s name.
- Participant 1 name.
- Participant 2 name.

- Event start time and date.
- Bet style description.
- Bet style mutator.
- Odds.

From this it was easy to check a random selection (yet full coverage) of odds from the database against the odds on the bookmakers' websites to validate the integrity of the data over the whole of the SAM system.



Bet At Home	Armenia	Bosnia and Herzegovina	2009-09-05 19:00:00	FT_Result_DrawPoss_PARTICIPANT#1_WIN	*NULL*	6.75
Bet At Home	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Handicap_Result_DrawPoss_PARTICPANT#2_WIN	-1.00	4.2
SkyBet	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Handicap_Result_DrawPoss_PARTICPANT#1_WIN	1.00	1.5
Bet At Home	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Handicap_Result_DrawPoss_PARTICPANT#1_WIN	1.00	1.55
Bet At Home	Arsenal	Chelsea	2009-04-18 17:15:00	FT_DoubleChance_PARTICIPANT_1_WIN_OR_DRAW	*NULL*	1.51
CentreBet	Arsenal	Chelsea	2009-04-18 17:15:00	FT_DoubleChance_PARTICIPANT_2_WIN_OR_DRAW	*NULL*	1.37
Bet At Home	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Result_DrawPoss_DRAW	*NULL*	3.1
CentreBet	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Result_DrawPoss_PARTICIPANT#1_WIN	*NULL*	3
CentreBet	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Handicap_Result_DrawPoss_PARTICPANT#1_WIN	1.00	1.53
Bet At Home	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Result_DrawPoss_PARTICIPANT#2_WIN	*NULL*	2.2
Bet At Home	Arsenal	Chelsea	2009-04-18 17:15:00	FT_DoubleChance_PARTICIPANT_2_WIN_OR_DRAW	*NULL*	1.3
Bet At Home	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Result_DrawPoss_PARTICIPANT#1_WIN	*NULL*	2.9
CentreBet	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Result_DrawPoss_PARTICIPANT#2_WIN	*NULL*	2.45
SkyBet	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Result_DrawPoss_DRAW	*NULL*	3.2
CentreBet	Arsenal	Chelsea	2009-04-18 17:15:00	FT_DoubleChance_PARTICIPANT_1_WIN_OR_DRAW	*NULL*	1.53
Bet At Home	Arsenal	Chelsea	2009-04-18 17:15:00	FT_DoubleChance_PARTICIPANT_1_OR_2_WIN	*NULL*	1.26
SkyBet	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Result_DrawPoss_PARTICIPANT#2_WIN	*NULL*	2.5
CentreBet	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Handicap_Result_DrawPoss_DRAW	1.00	4.33
SkyBet	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Handicap_Result_DrawPoss_DRAW	-1.00	4
Bet At Home	Arsenal	Chelsea	2009-04-18 17:15:00	FT_Handicap_Result_DrawPoss_DRAW	1.00	3.55

Figure 7.2 – Example Data Check output.

Compatibility Tests

Most system tests were carried out on a Windows Vista 64bit dual processor machine but a second machine was also used in order to test compatibility of the software. This machine was a Windows XP 32bit single processor machine that ran noticeably slower than the Vista machine. The software did show a slight quirk here in that the Acquisition Module display took a while to show if there was no connection to the server. This was corrected by creating a separate thread specifically for trying to connect to the Data Storage server upon program startup. Once this is completed (having succeeded or failed) the status of the submission section of the GUI is updated and the thread has no further use.

Continuity of Service

The "SAX Parser Problem" highlighted to this project just how much of an emphasis there should be on maintaining the continuity of the working system. It prompted an increased effort towards maintaining data connections and re-establishing them once they had been severed.

Testing of what happened following the loss and subsequent regaining of the three connections in question (between the database, Data Storage module, Data Acquisition module and The Internet) was implemented by switching off components and physically severing links. This highlighted numerous issues (listed below) that were subsequently addressed and corrected before being re-tested to prove their working state.

- Following the interruption of an XML miner Boolean values were not being reset which resulted in future mining operations returning no data.

- There was no attempt to re-establish a connection to the database following disconnection.
- The initial purge of old data was not reattempted if it failed during Data Storage module startup.
- A submission that got interrupted during the 3-stage submission process effectively locked the associated bookie out of future submissions. The answer was to have a submission time limit that terminated the submitter's window of opportunity after 30 seconds. (The 3-stage submission process was later replaced as described in the implementation section, which made this fix surplus to requirement.)
- The submission thread's operating loop was initially designed poorly, so that it was possible for the thread to return and no more submissions to take place due to a thrown exception.
- Loss of connection between acquisition and storage modules had already been programmed to reconnect, but testing showed that there was data loss as a result. This resulted in the implementation of a buffer to limit data loss and allow for glitches in connectivity.

8. Evaluation

Project Management

The first few weeks of this project were spent refining the projects specification, investigating the subject of the project and evolving a database design that would be the centre of this project (and used by other projects). The time and effort put into this initial stage really paid off as it developed an enthusiasm for the project that helped drive it on. The fact that the database has changed very little from this initial state is testament to the research that was put in at this stage.

The creation of a Gantt chart at an early stage was a very useful tool. It may not have been referred to very often or even kept to particularly closely but it certainly helped to appreciate the work that was required over the whole project. It would be fair to say that the time plan underestimated the time needed for thorough testing of the system, which is apparently a common mistake in a lot of software products.

It was suggested at the beginning of this project that the aims may be a little adventurous, which turned out to be correct. This advice was taken on board at the planning stage though, and the important stages of the project programmed in first so that any overrun could be at the expense of less important work. As the design methodology chosen was a sequential version of Parallel Development it was the whole development of the Data View module that was sacrificed for the quality of the other two modules.

This project has had regular meetings with the customer and has taken heed of valuable advice that has helped to shape the final piece(s) of software.

Project Result

Both the Data Storage and Data Acquisition modules have been designed and implemented to a high standard and true to the Revised Specification included in the Description section of this report. This project has fulfilled a large proportion of the functional and non-functional requirements laid out in the specification for this project. The requirements that have not been achieved due to the projects failure to implement a Data View module have been addressed and suggestions have been made how to achieve the same outcomes by using alternative methods; by using the WAMP phpMyAdmin interface along with the SQL queries prepared in Appendix F.

It was a part of the design that modules should do their utmost to maintain working service whenever possible. In the associated parts of the Implementation section of this report it has been detailed what the modules do to reconnect when service is lost. It is the opinion of this project that the hopes and aims on this front have been very successful and that this is a major achievement in making the software viable for “real” use.

Project Evaluation

It was clear from the start that this project was about more than just producing a working piece of software. It was about a software product that would be the framework for expansions and enhancements that might one day make a profitable system (for somebody). This required a high level of coding standards, good code documentation (see Appendix I for an example of the docs available in the code submission), programmer guides that explained how to expand the software and code that was as easy to expand as possible.

It is important to note that most software submitted as part of a Computer Science (or Software Systems) degree will probably be shown and evaluated around submission deadline day, then will never be ran again. This software will have a life after deadline day as it will be ran often for long periods of time and probably before a mark is issued, so there will be ample opportunity to spot any frailties. This prospect of end use has been one of the motivations behind the successful drive for high quality and stable software.

There is no doubt that Sports Arbitrage and More (SAM) does work, as a peek at the database clearly shows. In fact, at this very moment (15:42 17/04/09) in the database there are:

- Over 1500 possible bets (sets of odds).
- Over 1000 bet types.
- Over 150 events.
- Over 750 opportunities.

The past data is gradually increasing too.

The best opportunity is a 0.9% arbitrage on the result of a Coca Cola League 2 match (15:00 18/04/09) between Bury and Macclesfield Town, with odds from Skybet and PinnacleSports. This is very good considering there are only four bookmakers mined for only nine styles of odds. With more bet styles mined and more miners running this project is very confident that profitable opportunities will be highlighted with a frequency that makes taking advantage of them a very profitable prospect.

Critique

Apart from the implementation of the Data View module as was hoped, with more time the project could have been improved in a number of ways. Below are a just a few suggestions for improvements.

- After the implementation of HTML miners and the subsequent re-writing of XML miners to conform to a neater event driven style it is evident that other parts of the code could have done with a make-over in the same way. The miner scheduler class has a lot going on and would benefit greatly from being separated into separate classes to help encapsulate the two different functions more clearly. At present the mining threads and the miner scheduling threads could be easily confused, so miner schedulers could be separated off and defined as a separate object that has a timer to fire its operation.
- The random starting of miners could have been designed to deliver more efficient mining if it were more cleverly implemented. Adhoc schedules could have been created that minimised the occurrence of multiple schedules running at once and periods of complete inactivity.
- There could have been a more consistent theme of designing to interface, to make it easier to replace certain functions of the code. Moreover, there could have been the developed re-usable stubs and drivers that could have been used during testing of the project and been kept for

future re-use by the project customer for if they wanted to develop the code in the future. The testing that was carried out was not re-useable.

- It would have been nice to develop more HTML miners. As three XML miners were written there was an evolution as commonalities were found between mined code, but only creating a single HTML miner did not allow the same opportunity for improvement. With more HTML miners would have come more generic HTML mining code in the superclass'.

9. Bibliography

Books

Jiawei Han, Micheline Kamber (2006). Data Mining, Second Edition: Concepts and Techniques. Morgan Kaufmann

Hornick, Mark F, (2007). Java data mining : strategy, standard, and practice : a practical guide for architecture, design, and implementation. Morgan Kaufmann.

Websites

1. www.sports-arbitrage.com
   Useful information on Sports Arbitrage. They supply an arbitrage calculator that I used at the very start of the project whilst conducting research.

2. http://sportsarbitragereview.co.uk/
   A money making scheme based on Sports Arbitrage (or the promise of).

3. www.arbitragepro.com
   Rival software.

4. www.arbhunters.co.uk/
   Subscriber service offering sports arbitrage.

5. http://divisionoflabour.com/archives/002495.php
   Argument for possibility of sports arbitrage.

6. http://economics.about.com/od/arbitrage/a/sports_arb.htm
   Argument for possibility of sports arbitrage but not with a subscriber service.

7. http://www.arbets.com/guide/sports-arbitrage-betting-starter-guide/
   Guide to sports arbitrage.

8. http://economics.about.com/b/2006/04/20/can-sports-arbitrage-software-make-you-easy-money-2.htm
   Information about “middles”.

Glossary

Arbitrage	- Process of taking advantage of a price differential between the same product on two or more markets in order to make a profit.
Middle	- Like an arbitrage but a proportion of an event's outcomes are covered by more than one bet and the bet placer is not guaranteed to make a profit on an individual "middle". The advantage of this kind of opportunity is that the "middled" outcome returns up to double the returns of the other outcomes. If the middled outcome comes up say one fifth of the time and the others lose the bet placer one tenth of their stake, over an average five bets the bet placer will win once and lose four times. Overall that's winning 80% and losing 40%, which is an overall win gain of 40%.
Sports Arbitrage	- An opportunity where bets can be made to cover all outcomes of an event that guarantees the bet placer an overall profit.

Appendices

A

System Request

System Request

Project:

Sports Arbitrage - Develop a real-time system that advises a user how to bet on all outcomes of a specific sporting event where a guaranteed profit can be made.

Aim:

Sports arbitrage takes advantage of a price differential between two or more betting markets that supply odds on a common event. In a competitive market, the bookmaker's profit margin is low (to make a competitive price). If there is a difference in opinion between bookmakers as to the likelihood of an outcome occurring an arbitrage may be found if odds are taken for the different outcomes from different bookmakers. Bets are then placed on all possible outcomes of an event in a way where overall profit is guaranteed.

The required system automates the identification of arbitrages, records their occurrence and notifies the user with the information required to take advantage of them.

Functional Requirements:

- Be able to retrieve data from up to 100 bookmakers/sources of odds.
- Record odds from numerous events and various sports.
- Be able to record the following sports fixture result odds:
    1. Home win.
    2. Draw.
    3. Away Win.
- Be able to recognize when odds from different bookmakers are comparable (i.e. predict the same outcome of the same event).
- Be able to identify arbitrages as soon as the data is available and highlight them to a user.
- Allow the user to view data in at least the following ways:-
    1. Arbitrages, sorted by profitability.
    2. Sporting events, sorted by start time/date.
    3. All odds from every bookmaker on a selected event, with highlighting of the best odds.
- Allows an "administrator" to amend participant alias data.
- Keep a history of all odds collected for future analysis. This should be able to be backed up by administrator actions.

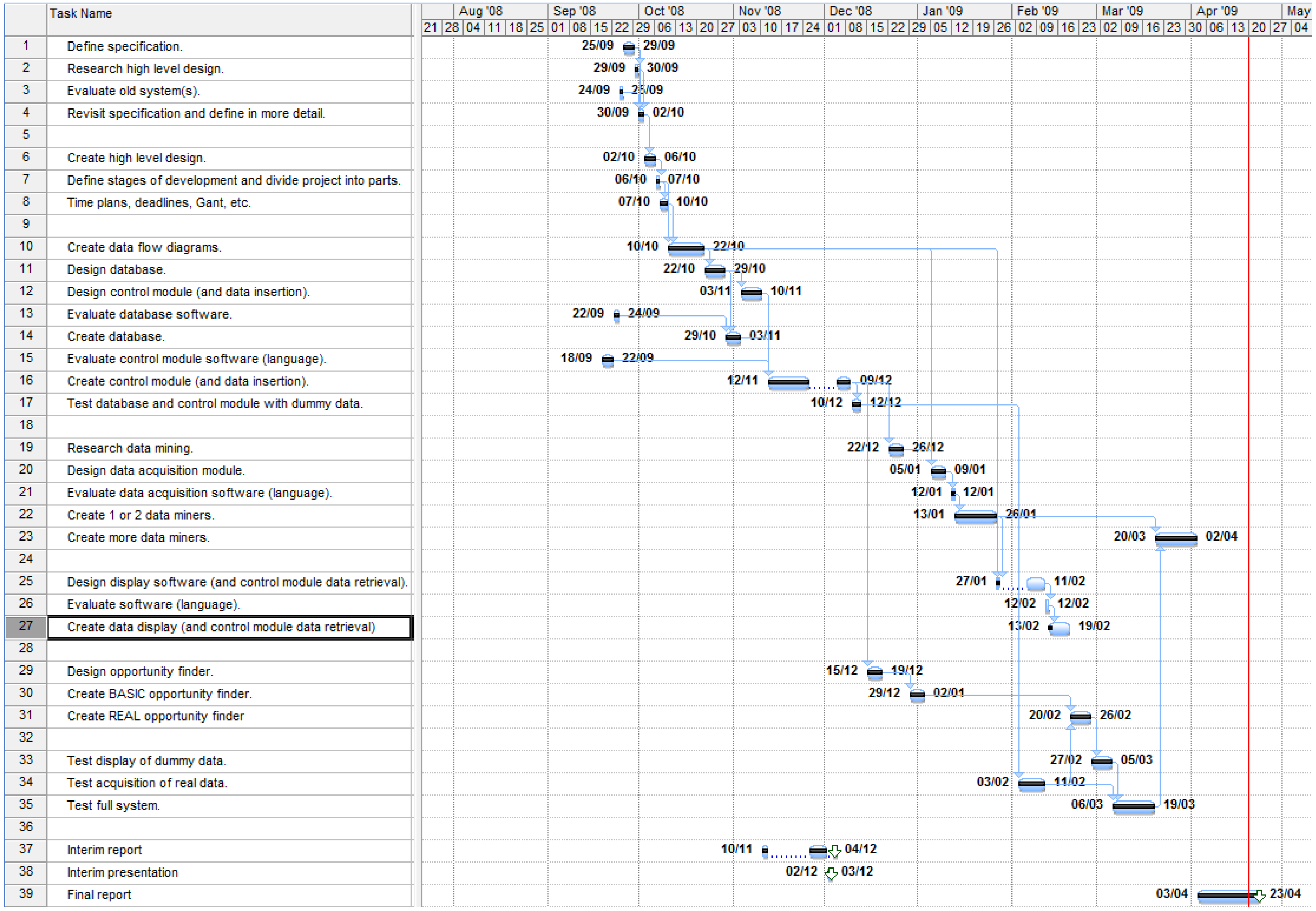
Non-functional Requirements:

- Due to the involvement of multiple project teams working towards similar goals, the customer would like the data storage and its interface(s) implemented as soon as possible. The customer can then let the other projects use these with a view to having multiple products that can be interfaced with the same "core" component.
- Should be implemented so that client's programming staff can maintain the supplied data miners (and parsers) and write more of their own.
- Should be well documented to facilitate the above requirement.
- Allow for implementation of data retrieval from xml and html sources.
- Should be possible to view data as a user from any computer (via the internet).
- System should be able to be installed on standard MS Windows/Vista based PCs.

Appendices

B

Gantt chart



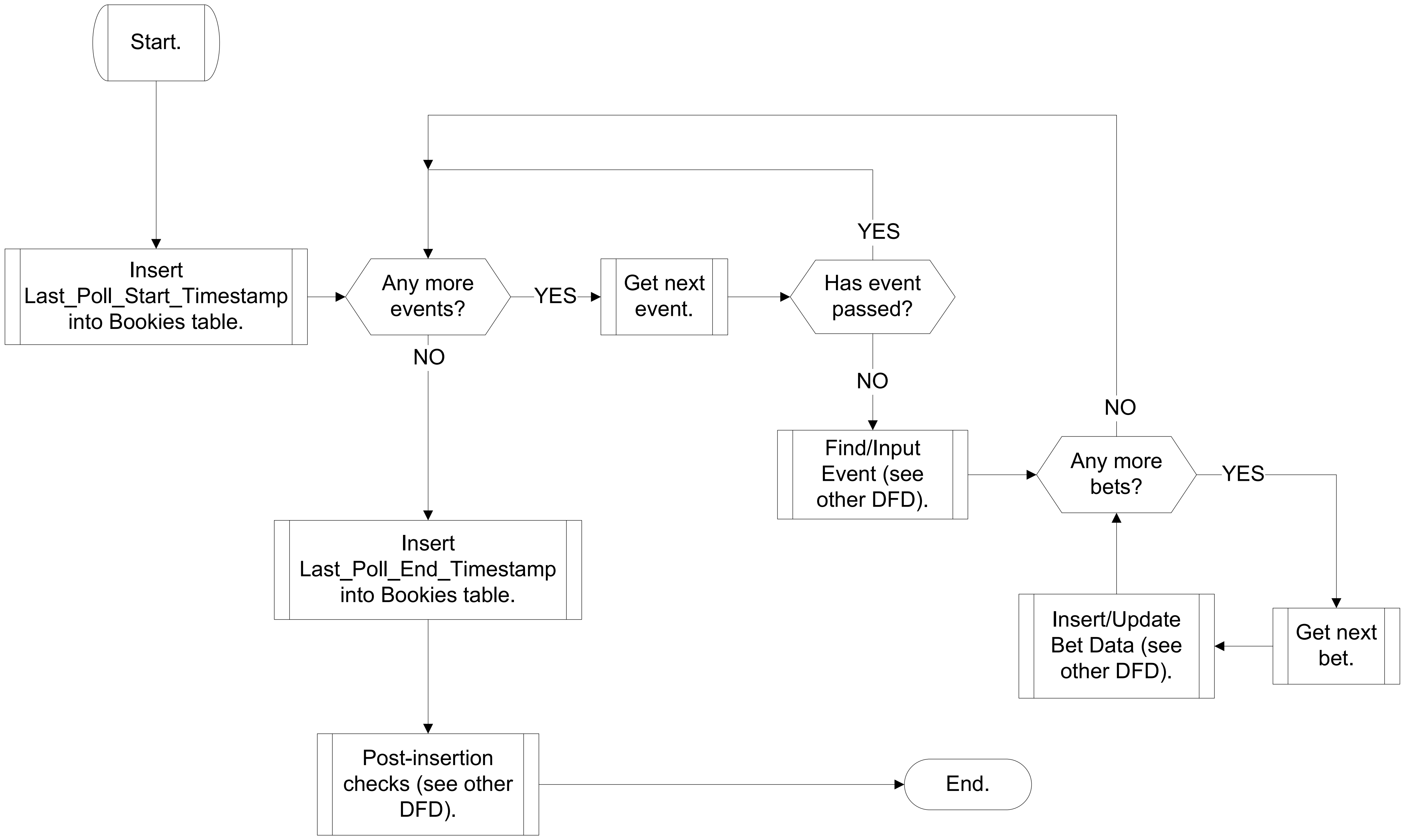
	Task Name
1	Define specification.
2	Research high level design.
3	Evaluate old system(s).
4	Revisit specification and define in more detail.
5	
6	Create high level design.
7	Define stages of development and divide project into parts.
8	Time plans, deadlines, Gant, etc.
9	
10	Create data flow diagrams.
11	Design database.
12	Design control module (and data insertion).
13	Evaluate database software.
14	Create database.
15	Evaluate control module software (language).
16	Create control module (and data insertion).
17	Test database and control module with dummy data.
18	
19	Research data mining.
20	Design data acquisition module.
21	Evaluate data acquisition software (language).
22	Create 1 or 2 data miners.
23	Create more data miners.
24	
25	Design display software (and control module data retrieval).
26	Evaluate software (language).
27	Create data display (and control module data retrieval)
28	
29	Design opportunity finder.
30	Create BASIC opportunity finder.
31	Create REAL opportunity finder
32	
33	Test display of dummy data.
34	Test acquisition of real data.
35	Test full system.
36	
37	Interim report
38	Interim presentation
39	Final report

Appendices

C

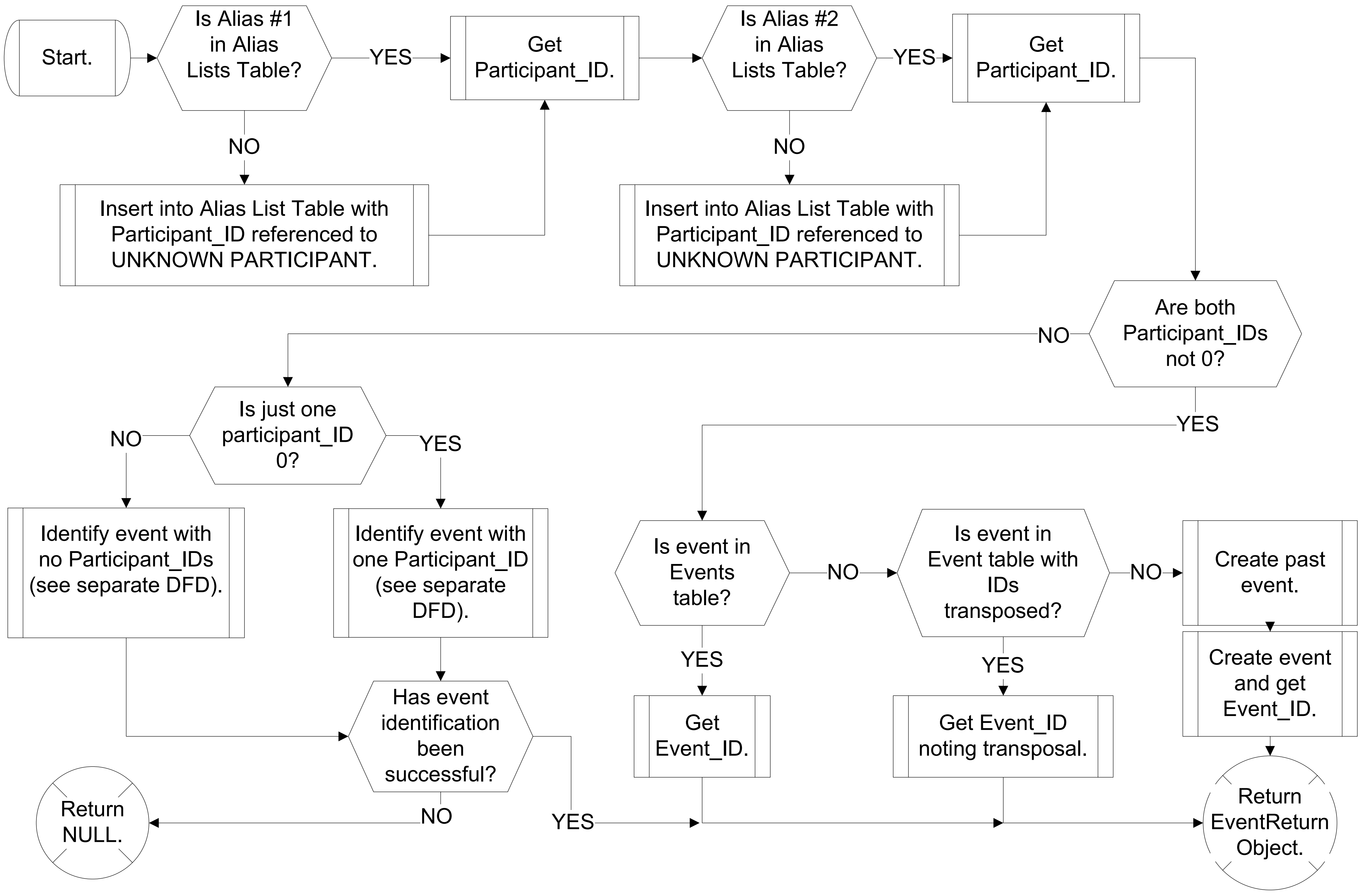
Input to database Data Flow Diagrams

Overview of operations on new data.



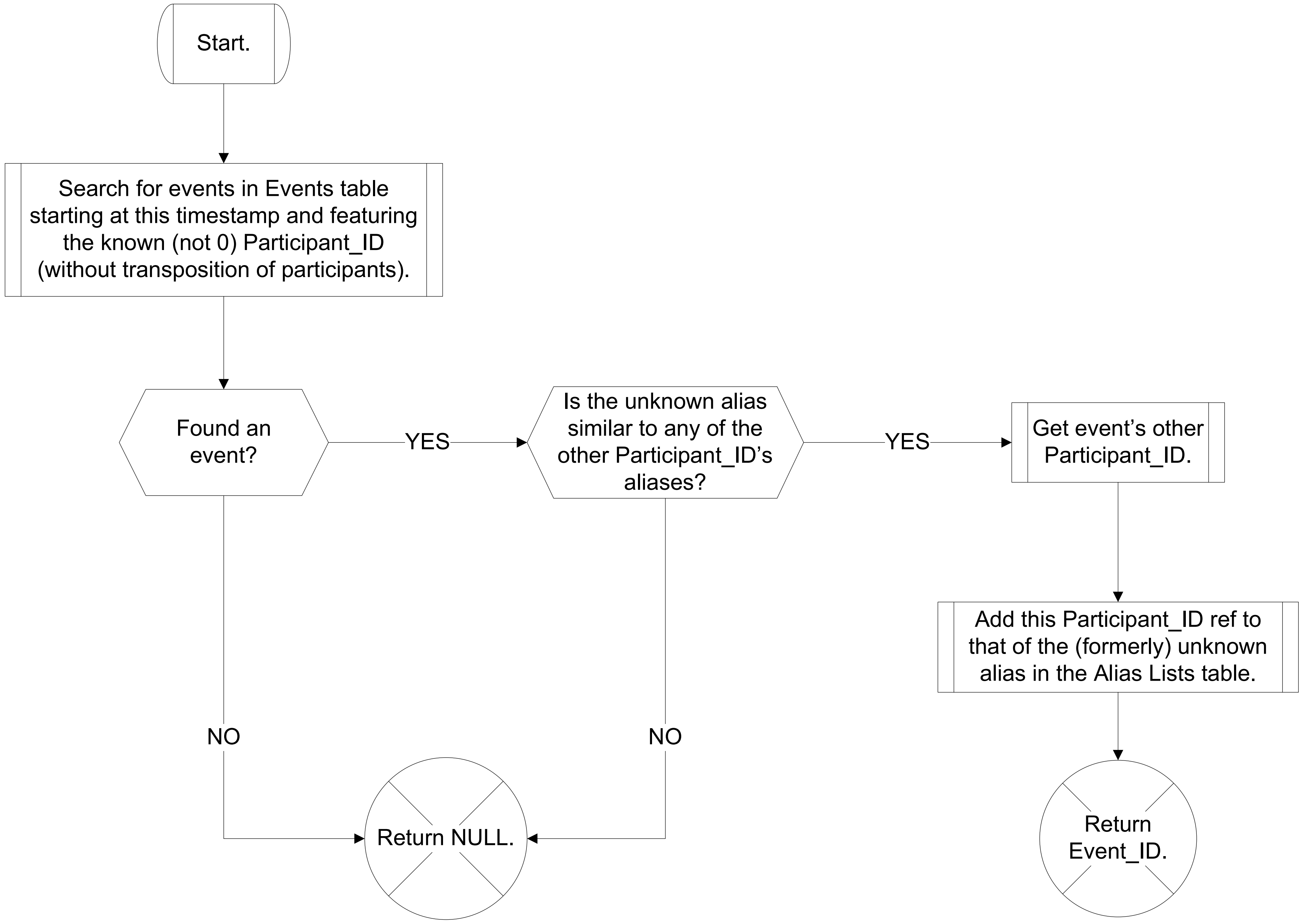
Start.
Insert Last_Poll_Start_Timestamp into Bookies table.
Any more events?
YES
NO
Get next event.
Has event passed?
YES
NO
Find/Input Event (see other DFD).
Any more bets?
YES
NO
Get next bet.
Insert/Update Bet Data (see other DFD).
Insert Last_Poll_End_Timestamp into Bookies table.
Post-insertion checks (see other DFD).
End.

Find/Input Event.

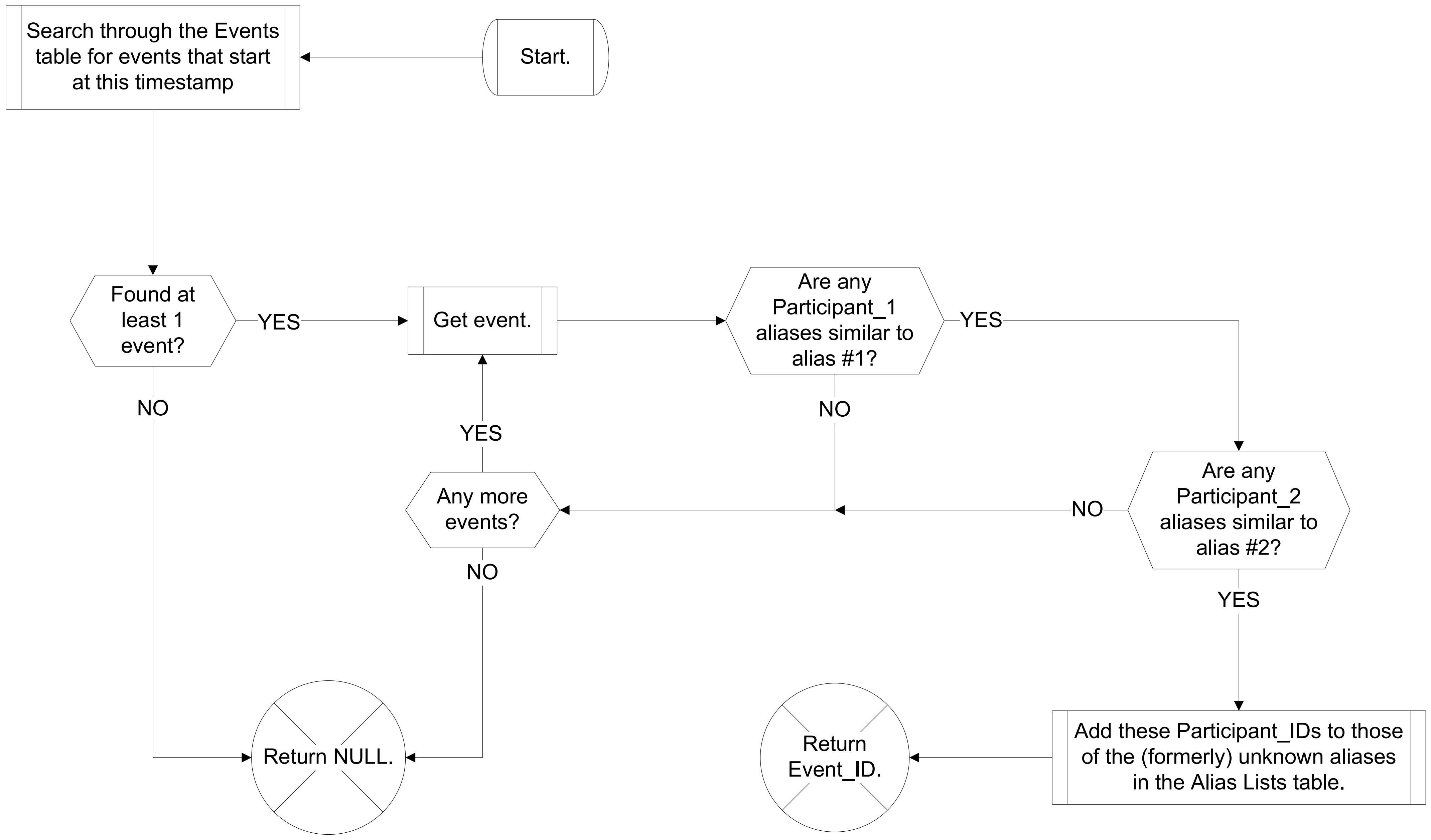


Start.
Is Alias #1 in Alias Lists Table?
YES
NO
Get Participant_ID.
Insert into Alias List Table with Participant_ID referenced to UNKNOWN PARTICIPANT.
Is Alias #2 in Alias Lists Table?
YES
NO
Get Participant_ID.
Insert into Alias List Table with Participant_ID referenced to UNKNOWN PARTICIPANT.
Are both Participant_IDs not 0?
NO
YES
Is just one participant_ID 0?
NO
YES
Identify event with no Participant_IDs (see separate DFD).
Identify event with one Participant_ID (see separate DFD).
Has event identification been successful?
NO
YES
Return NULL.
Is event in Events table?
NO
YES
Get Event_ID.
Is event in Event table with IDs transposed?
NO
YES
Get Event_ID noting transposal.
Create past event.
Create event and get Event_ID.
Return EventReturn Object.

Find Event Using One Participant_ID.

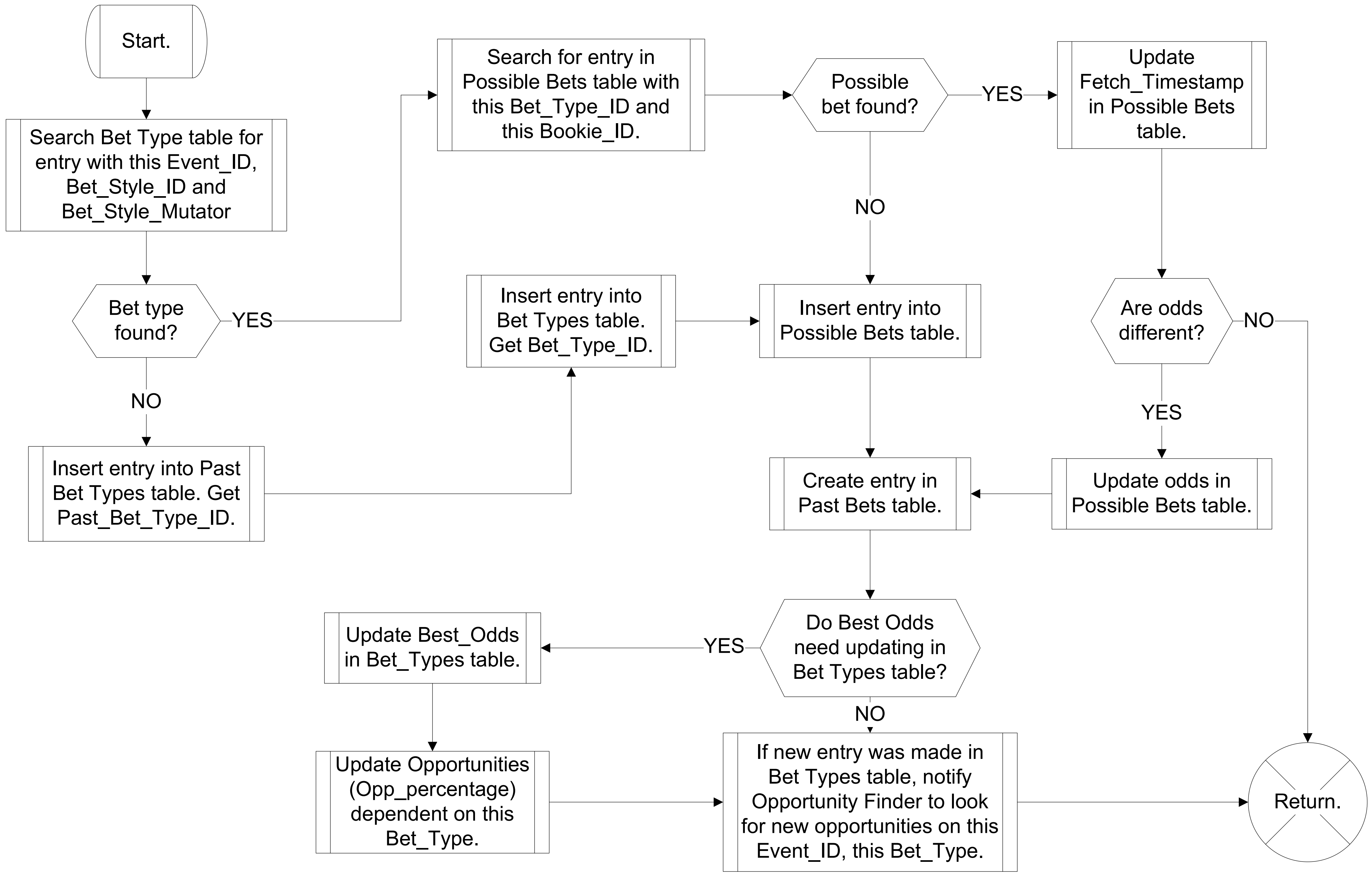

Start.
Search for events in Events table starting at this timestamp and featuring the known (not 0) Participant_ID (without transposition of participants).
Found an event?
YES
Is the unknown alias similar to any of the other Participant_ID's aliases?
YES
Get event's other Participant_ID.
Add this Participant_ID ref to that of the (formerly) unknown alias in the Alias Lists table.
Return Event_ID.
NO
NO
Return NULL.

Event Using No Participant_IDs.



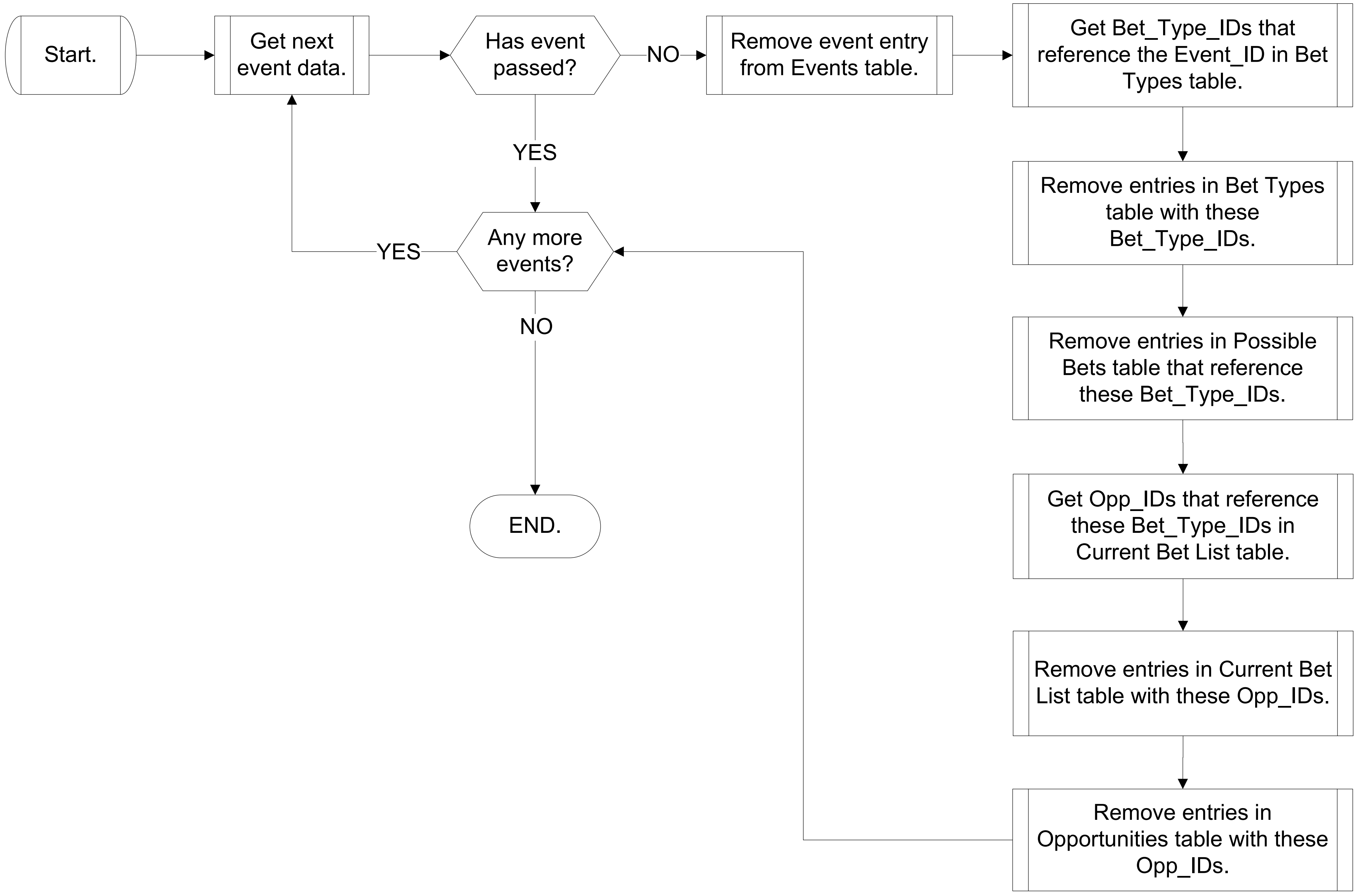
Search through the Events table for events that start at this timestamp
Start.
Found at least 1 event?
YES
NO
Get event.
Are any Participant_1 aliases similar to alias #1?
YES
NO
YES
Any more events?
NO
Are any Participant_2 aliases similar to alias #2?
NO
YES
Return NULL.
Return Event_ID.
Add these Participant_IDs to those of the (formerly) unknown aliases in the Alias Lists table.

Insert or Update Bet.

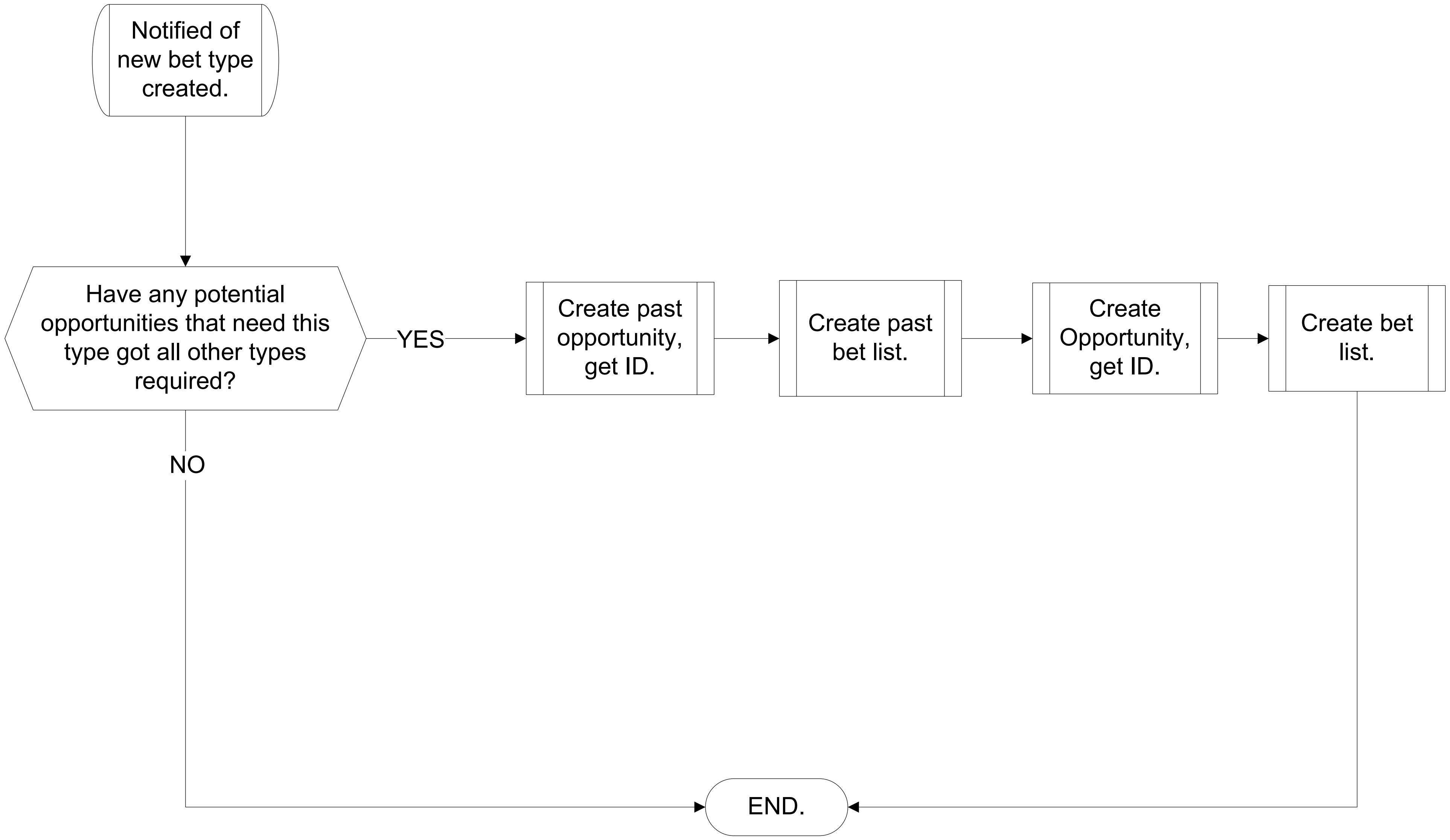


Start.
Search Bet Type table for entry with this Event_ID, Bet_Style_ID and Bet_Style_Mutator
Bet type found?
YES
NO
Insert entry into Past Bet Types table. Get Past_Bet_Type_ID.
Search for entry in Possible Bets table with this Bet_Type_ID and this Bookie_ID.
Insert entry into Bet Types table. Get Bet_Type_ID.
Possible bet found?
YES
NO
Insert entry into Possible Bets table.
Update Fetch_Timestamp in Possible Bets table.
Are odds different?
NO
YES
Update odds in Possible Bets table.
Create entry in Past Bets table.
Do Best Odds need updating in Bet Types table?
YES
NO
Update Best_Odds in Bet_Types table.
Update Opportunities (Opp_percentage) dependent on this Bet_Type.
If new entry was made in Bet Types table, notify Opportunity Finder to look for new opportunities on this Event_ID, this Bet_Type.
Return.

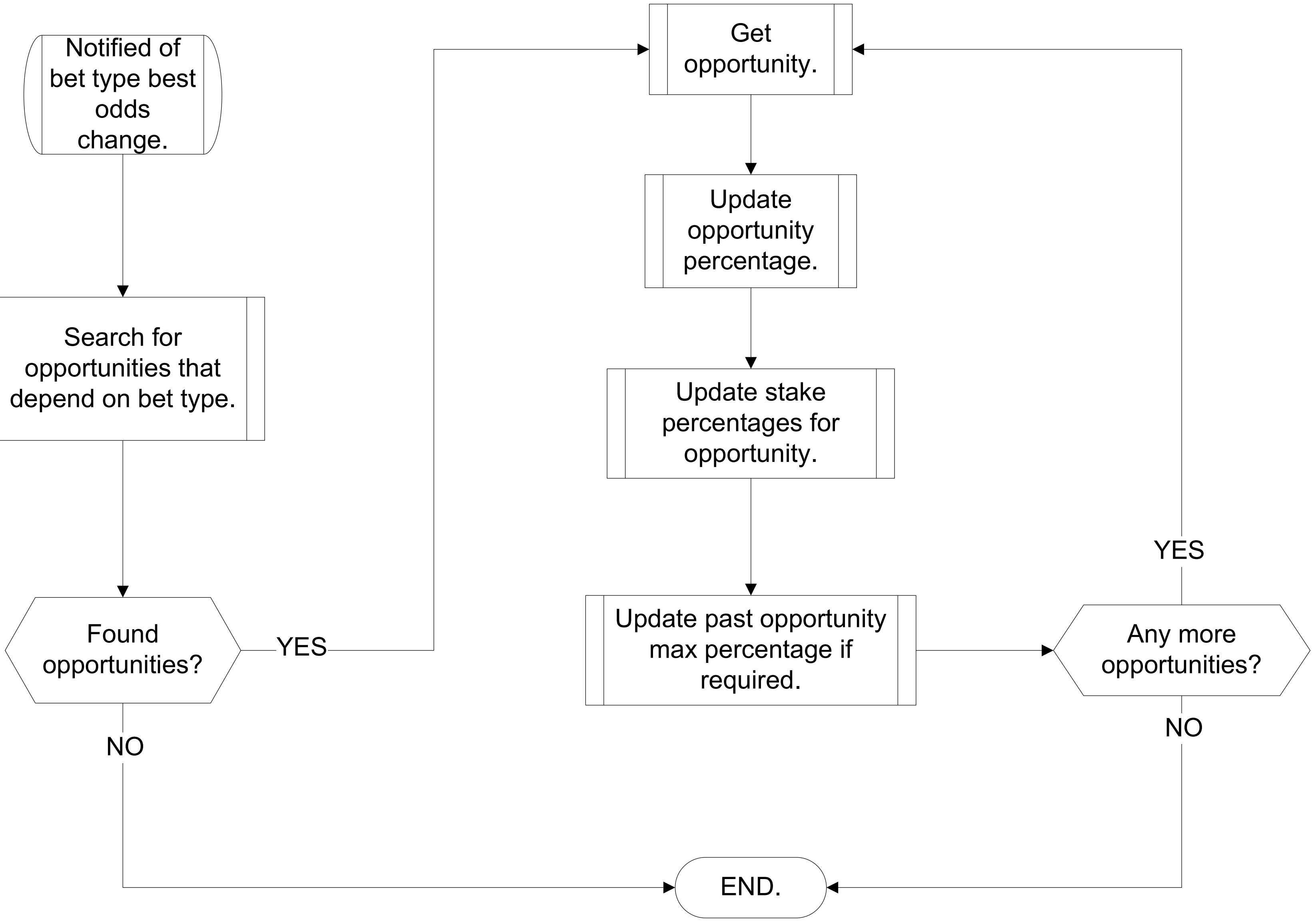
Current Data Management.


Start.
Get next event data.
Has event passed?
NO
Remove event entry from Events table.
Get Bet_Type_IDs that reference the Event_ID in Bet Types table.
Remove entries in Bet Types table with these Bet_Type_IDs.
Remove entries in Possible Bets table that reference these Bet_Type_IDs.
Get Opp_IDs that reference these Bet_Type_IDs in Current Bet List table.
Remove entries in Current Bet List table with these Opp_IDs.
Remove entries in Opportunities table with these Opp_IDs.
YES
Any more events?
YES
NO
END.

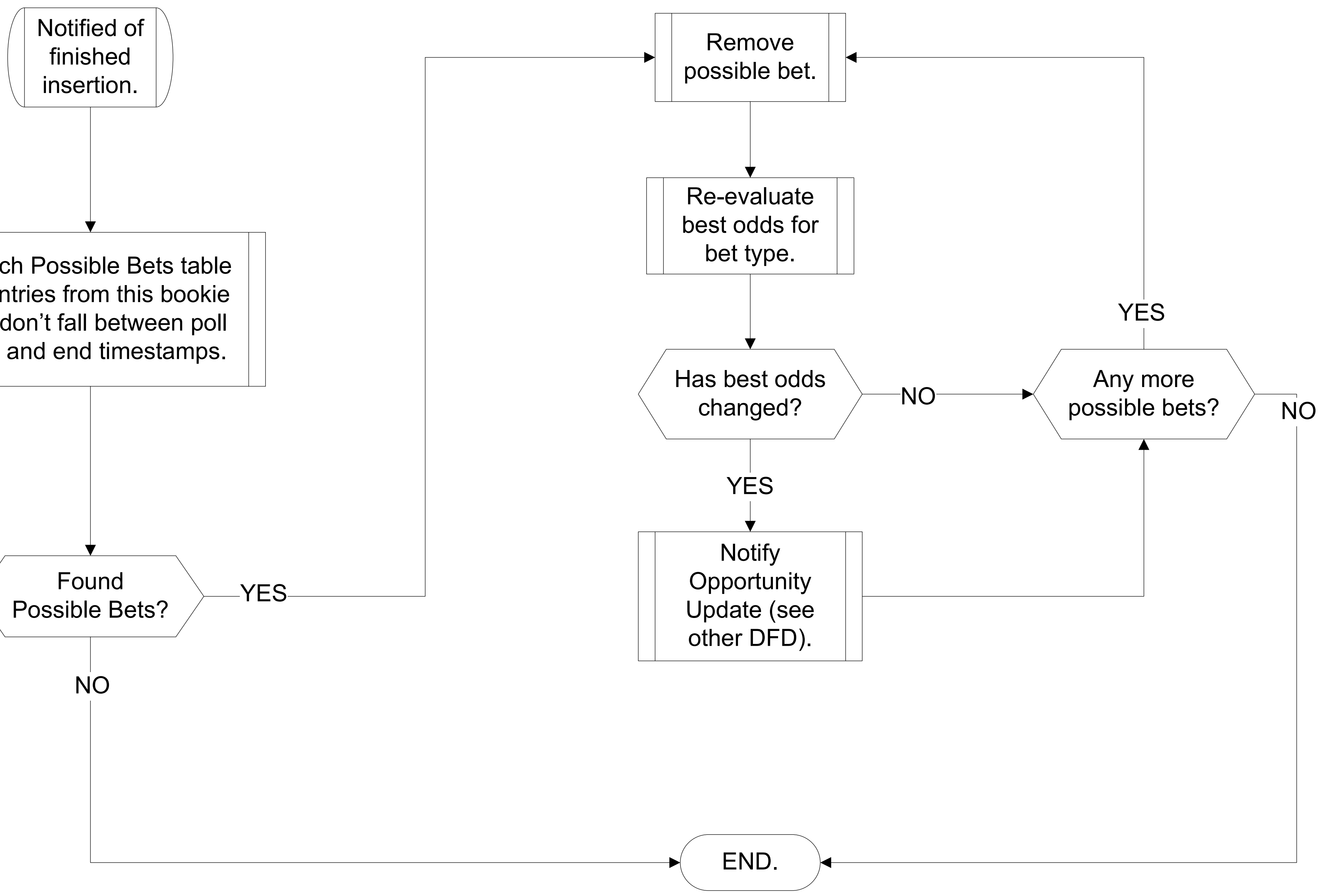
Opportunity Finder


Notified of new bet type created.
Have any potential opportunities that need this type got all other types required?
YES
Create past opportunity, get ID.
Create past bet list.
Create Opportunity, get ID.
Create bet list.
NO
END.

Opportunity Updater


Notified of bet type best odds change.
Search for opportunities that depend on bet type.
Found opportunities?
YES
NO
Get opportunity.
Update opportunity percentage.
Update stake percentages for opportunity.
Update past opportunity max percentage if required.
Any more opportunities?
YES
NO
END.

Post-Insertion Checks


Notified of finished insertion.
Search Possible Bets table for entries from this bookie that don't fall between poll start and end timestamps.
Found Possible Bets?
YES
NO
Remove possible bet.
Re-evaluate best odds for bet type.
Has best odds changed?
NO
YES
Notify Opportunity Update (see other DFD).
Any more possible bets?
YES
NO
END.

D

Database Tables

Current Market Information

Possible Bets		
Bet_Type_ID	PK	FK
Bookie_ID	PK	FK
Odds		
Fetch_Timestamp		
Bet_URL		
Bet_Caution_ID		FK

Bet Types		
Bet_Type_ID	PK	FK
Event_ID		FK
Bet_Style_ID	CK	FK
Bet_Style_Mutator		
Best_Odds		
Past_Bet_Type_ID		FK

Events		
Event_ID	PK	FK
Participant_ID_#1		FK
Participant_ID_#2	CK	FK
Event_Start_Timestamp		
Event_Description		

Bookies		
Bookie_ID	PK	FK
Bookie_Name		
Last_Poll_Start_Timestamp		
Last_Poll_End_Timestamp		
Bookie_URL		
Bookie_Percentage		

Participants		
Participant_ID	PK	FK
Primary_Name		
Sport_ID		FK

Alias Lists		
Alias (lower case only)	PK	
Participant_ID		FK

Current Opportunity Information

Opportunities		
Opp_ID	PK	FK
Opp_Percentage		
Opp_Type_ID		
Past_Opp_ID		FK

Opportunity Bet List		
Opp_ID	PK	FK
Bet_Type_ID	PK	FK
Stake_Percentage		

Manually Inserted Reference Information

Bookie Caution List		
Sport_ID	PK	FK
Bookie_ID	PK	FK
Bookie_Caution_ID		FK

Opportunity Types		
Opp_Type_ID	PK	FK
Opp_Description		
Profitable_Threshold		

Sport Types		
Sport_ID	PK	FK
Draw_Possible		
Sport_Description		

Bookie Cautions		
Bookie_Caution_ID	PK	FK
Bookie_Caution Description		

Bet Styles		
Bet_Style_ID	PK	FK
Bet_Description		

Bet Cautions		
Bet_Caution_ID	PK	FK
Bet_Caution_Description		

Past Information

Past Opportunities		
Past_Opp_ID	PK	FK
Opp_Type_ID		FK
Max_Percentage		

Past Bet List		
Past_Opp_ID	PK	FK
Past_Bet_Type_ID	PK	FK

Past Events		
Past_Event_ID	PK	FK
Participant_ID_#1		FK
Participant_ID_#2	CK	FK
Event_Start_Timestamp		
Event_Description		

Past Bets		
Past_Bet_Type_ID	PK	FK
Bookie_ID	PK	FK
Past_Fetch_Timestamp	PK	
Past_Odds		
Bet_Caution_ID		FK

Past Bet Types		
Past_Bet_Type_ID	PK	FK
Past_Event_ID		FK
Past_Bet_Style_ID	CK	FK
Past_Bet_Style_Mutator		

Appendices

E

Sample database data and descriptions of table columns

alias_list

Alias (Lower case alias of a participant.)	Participant_ID (Foreign Key. Participant_ID that this alias belongs to.)
arsenal	101
aston villa	102
blackburn rovers	103
arsenl	101
arsenal f.c.	101
aston v	102
aston villa f.c.	102

bet_cautions

Bet_Caution_ID (Easy to reference ID. Represents a reason to be cautious (or not) about a bet.)	Bet_Caution_Description (Description of the caution for output purposes.)
0	Low/No bookie profit from set of bets.

bet_styles

Bet_Style_ID (Easy to reference ID. Represents the betting style described below.)	Bet_Description (Description of the bet style (e.g. Result Participant #1 win). For output purposes.)
0	Result Participant #1 Win
1	Result Participant #2 Win
2	Result Draw
3	Asian Handicap Participant #1 Win
4	Asian Handicap Participant #2 Win
5	Handicap Participant #1 Win
6	Handicap Participant #2 Win
7	Handicap Draw

bet_types

Bet_Type_ID (Easy to reference ID representing the unique combination of Event_ID, Bet_Style_ID and Bet_Style Mutator. When creating entry leave as NULL and it will auto-increment.)	Event_ID (Easy to reference ID representing the event this bet type is on.)	Bet_Style_ID (Easy to reference ID. Represents the style of bet this is.)	Bet_Style_Mutator (This mutates the bet style by adding a variable (e.g. +0.5 Identifies an Asian Handicap participant #1 win as a +0.5). Can be NULL.)	Best_Odds (Best odds of all the possible bets of this type.)	Past_Bet_Type_ID (Easy to reference ID. Represents the place where the history of this bet type is recorded.)
1	1	0	*NULL*	1.4	4
2	1	1	*NULL*	11	5
3	1	2	*NULL*	5	6

bookies

Bookie_ID (Easy to reference ID.)	Bookie_Name (Name of the bookie (for output purposes only).)	Last_Poll_Start_Timestamp (Timestamp representing the start of the data miner's transmission)	Last_Poll_End_Timestamp (Timestamp representing the end of the data miner's transmission.)	Bookie_URL (String representing the URL of the bookie's website.)	Bookie_Percentage (Float representing the percentage of your winnings that the bookie takes (normally only for betting exchanges).)
0	William Hill	2008-10-27 00:00:00	2008-10-27 00:05:00	http://www.willhill.com	0
1	Ladbrokes	2008-10-27 00:00:00	2008-10-27 00:03:00	http://www.ladbrokes.com	0
2	Bet365	2008-10-27 00:00:00	2008-10-27 00:09:00	www.bet365.com	0
3	Betfair	0000-00-00 00:00:00	0000-00-00 00:00:00	http://www.betfair.com	5

bookie_cautions

Bookie_Caution_ID (Easy to reference ID.)	Bookie_Caution_Description (Description of the reason to be cautious (e.g. Football result taken at the end of extra time instead of 90 mins). Used during output.)
0	Results taken as after any extra time, not after 9...
1	Winner is decided even if match abandoned.

bookie_caution_list

Sport_ID (Easy to reference ID representing the sport this caution is concerned with.)	Bookie_ID (Easy to reference ID representing the bookie this caution is concerned with.)	Bet_Caution_ID (Caution ID that is relevent to this bookie and this sport.)
0	0	0
2	1	1

events

Event_ID (Easy to reference ID. When creating entry leave as NULL and it will auto-increment.)	Participant_ID_1 (Easy to reference ID. Foreign key to reference the "Home" participant.)	Participant_ID_2 (Easy to reference ID. Foreign key to reference the "Away" participant.)	Event_Start_Timestamp (Timestamp representing the time and date when the event starts.)	Event_Description (Description of the event (e.g. English premier league). Can be Null. Used during output.)
1	0	3	2008-11-12 19:30:00	Premier League
2	2	1	2008-11-12 19:30:00	Premier League

opportunities

Opp_ID (Easy to reference ID. When creating entry leave as NULL and it will auto-increment.)	Opp_Percentage (Float representing the profit margin of this opportunity.)	Opp_Type_ID (Easy to reference ID. Represents the type of opportunity.)	Past_Opp_ID (Easy to reference ID. Represents the place where the history of this opportuniyty is stored.)
1	-0.6	0	2

opportunity_bet_list

Opp_ID (Easy to reference ID.)	Bet_Type_ID (Easy to reference ID. **This table links an opportunity to the bet types that it depends on**.)	Stake_Percentage Float representing the percentage of a stake that the user should bet on this type.
1	0	30
1	1	40
1	2	30

opportunity_types

Opp_Type_ID (Easy to reference ID.)	Opp_Description (Description of the opportunity (e.g. Arbitrage). Used during output.)	Profitable_Threshold (Float representing the threshold at which the Opp_Percentage becomes "Interesting".)
0	Arbitrage	0
1	Middle	-5

participants

Participant_ID (Easy to reference ID representing an individual event participant. When creating entry leave as NULL and it will auto-increment.)	Primary_Name (A String used to identify this participant during output.)	Sport_ID (Easy to reference ID representing the sport/event type that this participant competes in.)
0	UNKNOWN PARTICIPANT 0	0
1	UNKNOWN PARTICIPANT 1	1
2	UNKNOWN PARTICIPANT 2	2
3	UNKNOWN PARTICIPANT 3	3
100	Arsenal	0
101	Aston Villa	0
102	Blackburn Rovers	0
103	Bolton Wanderers	0
104	Chelsea	0
105	Everton	0
106	Fulham	0
107	Hull City	0
108	Liverpool	0
109	Manchester City	0
110	Manchester United	0
111	Middlesbrough	0
112	Newcastle United	0
113	Portsmouth	0
114	Stoke City	0

Participant_ID (Easy to reference ID representing an individual event participant. When creating entry leave as NULL and it will auto-increment.)	Primary_Name (A String used to identify this participant during output.)	Sport_ID (Easy to reference ID representing the sport/event type that this participant competes in.)
115	Sunderland	0
116	Tottenham Hotspur	0
117	West Bromwich Albion	0
118	West Ham United	0
119	Wigan Athletic	0

past_bets

Past_Bet_Type_ID (Easy to reference ID representing the history of a possible bet.)	Bookie_ID (Easy to reference ID representing the bookie.)	Past_Fetch_Timestamp (Timestamp representing when these odds were retrieved.)	Past_Odds (Float representing the odds of this bet type, of this bookie, at this time.)	Bet_Caution_ID (Easy to reference ID representing a reason to be cautious of this bet.)
1	0	2008-10-01 00:00:00	1.5	*NULL*
1	0	2008-10-01 05:00:00	2	*NULL*
2	0	2008-10-01 00:00:00	2	*NULL*
2	0	2008-10-01 05:00:00	2	*NULL*
3	0	2008-10-01 00:00:00	10	*NULL*
3	0	2008-10-01 05:00:00	10	*NULL*
1	1	2008-10-01 00:00:00	1.4	*NULL*
1	1	2008-10-01 04:00:00	1.3	*NULL*
2	1	2008-10-01 00:00:00	2.2	*NULL*
2	1	2008-10-01 04:00:00	2.3	*NULL*
3	1	2008-10-01 00:00:00	12	*NULL*
3	1	2008-10-01 04:00:00	13	*NULL*
1	2	2008-10-01 00:00:00	1	*NULL*
1	2	2008-10-01 06:00:00	1.1	*NULL*
2	2	2008-10-01 00:00:00	1.5	*NULL*
2	2	2008-10-01 06:00:00	1.6	*NULL*

Past_Bet_Type_ID (Easy to reference ID representing the history of a possible bet.)	Bookie_ID (Easy to reference ID representing the bookie.)	Past_Fetch_Timestamp (Timestamp representing when these odds were retrieved.)	Past_Odds (Float representing the odds of this bet type, of this bookie, at this time.)	Bet_Caution_ID (Easy to reference ID representing a reason to be cautious of this bet.)
3	2	2008-10-01 00:00:00	17	*NULL*
3	2	2008-10-01 06:00:00	20	*NULL*
4	0	2008-10-27 00:00:00	1.33	*NULL*
4	1	2008-10-27 00:00:00	1.37	*NULL*
4	2	2008-10-27 00:00:00	1.4	*NULL*
5	0	2008-10-27 00:00:00	5	*NULL*
5	1	2008-10-27 00:00:00	4.8	*NULL*
5	2	2008-10-27 00:00:00	4	*NULL*
6	0	2008-10-27 00:00:00	10	*NULL*
6	1	2008-10-27 00:00:00	11	*NULL*
6	2	2008-10-27 00:00:00	10	*NULL*

past_bet_list

Past_Opp_ID (Easy to reference ID.)	Past_Bet_Type_ID (Easy to reference ID. **This table links a past opportunity to the past bet types that it depended on**.)
1	1
1	2
1	3
2	4
2	5
2	6

past_bet_types

Past_Bet_Type_ID (Easy to reference ID representing a past bet type. When creating entry leave as NULL and it will auto-increment.)	P_Event_ID (Easy to reference ID representing past event.)	Bet_Style_ID (Easy to reference ID representing the style of the past bet.)	Past_Bet_Style_Mutator (This is the past version of the Bet_Style_Mutator.)
1	1	0	*NULL*
2	1	1	*NULL*
3	1	2	*NULL*
4	3	0	*NULL*
5	3	1	*NULL*
6	3	2	*NULL*

past_events

Past_Event_ID (Easy to reference ID representing a past event. When creating entry leave as NULL and it will auto-increment.)	Participant_ID_1 (Easy to reference ID. Foreign key to reference the "Home" participant.)	Participant_ID_2 (Easy to reference ID. Foreign key to reference the "Away" participant.)	Past_Event_Description (Description of the event for output purposes. Can be NULL.)	Event_Start_Timestamp (Timestamp representing the time and date when this event was due to start.)
3	0	3	Premier League	2008-11-12 19:30:00
4	2	1	Premier League	2008-11-12 19:30:00
1	3	1	FA Cup	2008-10-06 00:00:00
2	0	2	League Cup	2008-10-04 00:00:00

past_opportunities

Past_Opp_ID (Easy to reference ID representing a past opportunity. When creating entry leave as NULL and it will auto-increment.)	Opp_Type_ID (Easy to reference ID. Represents the type of opportunity.)	Max_Percentage (Float representing the best percentage that this opportunity returned.)
1	0	1.55
2	0	-0.6

possible_bets

Bet_Type_ID (Easy to reference ID. Represents the type of bet.)	Bookie_ID (Easy to reference ID. Represents the identity of the bookie.)	Odds (Float representing the odds offered.)	Fetch_Timestamp (Timestamp representing the time and date when these odds were collected.)	Bet_URL (String representing the URL to get straight to this bet with this bookie. Can be NULL.)	Bet_Caution_ID (Easy to reference ID. Represents a reson to be cautious about this bet.)
1	0	1.33	2008-10-27 00:00:00	*NULL*	*NULL*
1	1	1.37	2008-10-27 00:00:00	*NULL*	*NULL*
1	2	1.4	2008-10-27 00:00:00	*NULL*	*NULL*
2	0	5	2008-10-27 00:00:00	*NULL*	*NULL*
2	1	4.8	2008-10-27 00:00:00	*NULL*	*NULL*
2	2	4	2008-10-27 00:00:00	*NULL*	*NULL*
3	0	10	2008-10-27 00:00:00	*NULL*	*NULL*
3	1	11	2008-10-27 00:00:00	*NULL*	*NULL*
3	2	10	2008-10-27 00:00:00	*NULL*	*NULL*

sport_types

Sport_ID (Easy to reference ID. Represents a sport/event type.)	Draw_Possible (Boolean representing whether a draw is possible (1) or impossible (0) in this sport.)	Sport_Description (String describing this sport/event type.)
0	1	Football
1	1	Rugby
2	0	Tennis
3	1	BasketBall

Appendices

F

Data View SQL queries

Data View SQL Queries

Comments are in square brackets e.g. [Integer corresponding to opportunity type].

View all opportunities, sorted by profitability, for a single opportunity type:

```
SELECT      Opp_Type_ID,
            Opp_Percentage
FROM        opportunities
WHERE       Opp_Type_ID = [Integer corresponding to an opportunity type]
ORDER BY    Opp_Percentage DESC
```

View all profitable opportunities, sorted by profitability:

```
SELECT      Opp_ID,
            Opp_Percentage,
            Opp_Description
FROM        opportunities
INNER JOIN  opportunity_types
ON          opportunities.Opp_Type_ID = opportunity_types.Opp_Type_ID
WHERE       Opp_Percentage >= Profitable_Threshold
ORDER BY    Opp_Percentage - Profitable_Threshold DESC
```

View all profitable opportunities, sorted by profitability, for a single opportunity type:

```
SELECT      Opp_ID,
            Opp_Percentage,
            Opp_Description
FROM        opportunities
INNER JOIN  opportunity_types
ON          opportunities.Opp_Type_ID = opportunity_types.Opp_Type_ID
WHERE       Opp_Percentage >= Profitable_Threshold
AND         opportunities.Opp_Type_ID = [Integer corresponding to an opportunity type]
ORDER BY    Opp_Percentage - Profitable_Threshold DESC
```

View all sporting events, sorted by start time/date:

```
SELECT      Event_ID,
            Participants.Primary_Name,
            Participants2.Primary_Name,
            Event_Start_Timestamp,
            Event_Description,
            Sport_Description
FROM        events
INNER JOIN  participants
ON          events.Participant_ID_1 = participants.Participant_ID
INNER JOIN  participants participants2
ON          events.Participant_ID_2 = participants2.Participant_ID
INNER JOIN  sport_types
ON          sport_types.Sport_ID = participants.Sport_ID
ORDER BY    Event_Start_Timestamp
```

View all sporting events, sorted by start time/date, for a single sport type:

```
SELECT      Event_ID,
            Participants.Primary_Name,
            Participants2.Primary_Name,
            Event_Start_Timestamp,
            Event_Description
FROM        events
INNER JOIN  participants
ON          events.Participant_ID_1 = participants.Participant_ID
INNER JOIN  participants participants2
ON          events.Participant_ID_2 = participants2.Participant_ID
WHERE       participants.Sport_ID = [Integer corresponding to a sport type]
ORDER BY    Event_Start_Timestamp
```

View all sporting events, sorted by first participant name, for a single sport type:

```
    SELECT      Event_ID,
                Participants.Primary_Name,
                Participants2.Primary_Name,
                Event_Start_Timestamp,
                Event_Description
    FROM        events
    INNER JOIN  participants
    ON          events.Participant_ID_1 = participants.Participant_ID
    INNER JOIN  participants participants2
    ON          events.Participant_ID_2 = participants2.Participant_ID
    WHERE       participants.Sport_ID = [Integer corresponding to a sport type]
    ORDER BY    Participants.Primary_Name
```

View all sporting events, sorted by second participant name, for a single sport type:

```
    SELECT      Event_ID,
                Participants.Primary_Name,
                Participants2.Primary_Name,
                Event_Start_Timestamp,
                Event_Description
    FROM        events
    INNER JOIN  participants
    ON          events.Participant_ID_1 = participants.Participant_ID
    INNER JOIN  participants participants2
    ON          events.Participant_ID_2 = participants2.Participant_ID
    WHERE       participants.Sport_ID = [Integer corresponding to a sport type]
    ORDER BY    Participants2.Primary_Name
```

View all sporting events, sorted by event description, for a single sport type:

```
    SELECT      Event_ID,
                Participants.Primary_Name,
                Participants2.Primary_Name,
                Event_Start_Timestamp,
                Event_Description
    FROM        events
    INNER JOIN  participants
    ON          events.Participant_ID_1 = participants.Participant_ID
    INNER JOIN  participants participants2
    ON          events.Participant_ID_2 = participants2.Participant_ID
    WHERE       participants.Sport_ID = [Integer corresponding to a sport type]
    ORDER BY    Event_Description
```

View all opportunities on a single event, sorted by profitability:

```
SELECT          Event_ID,
                Opp_Description,
                Profitable_Threshold,
                Opp_Percentage
FROM
(SELECT         DISTINCT opportunities.Opp_ID,
                bet_types.Event_ID,
                Opp_Description,
                Profitable_Threshold,
                Opp_Percentage
FROM            opportunities
INNER JOIN      opportunity_bet_list
ON              opportunities.Opp_ID = opportunity_bet_list.Opp_ID
INNER JOIN      bet_types
ON              bet_types.Bet_Type_ID = opportunity_bet_list.Bet_Type_ID
INNER JOIN      events
ON              events.Event_ID = bet_types.Event_ID
INNER JOIN      opportunity_types
ON              opportunities.Opp_Type_ID = opportunity_types.Opp_Type_ID
WHERE           events.Event_ID = [Integer corresponding to an event id])
AS              temp_table
ORDER BY        Opp_Percentage - Profitable_Threshold DESC
```

View all possible bets on a single opportunity, grouped by bet type, ordered by odds:

WARNING: The odds displayed here are "processed" to take into account any cut in winnings that the bookmaker takes. The odds shown here may not be the odds that were collected but the comparable equivalent according to the value in bookies.Bookie_Percentage.

```
SELECT          opportunities.Opp_ID,
                Bet_Description,
                Bet_Style_Mutator,
                Odds,
                Bookie_Name,
                Bet_URL,
                Bet_Caution_Description
FROM            possible_bets
INNER JOIN      bet_types
ON              bet_types.Bet_Type_ID = possible_bets.Bet_Type_ID
INNER JOIN      opportunity_bet_list
ON              opportunity_bet_list.bet_Type_ID = bet_types.bet_Type_ID
INNER JOIN      opportunities
ON              opportunities.Opp_ID = opportunity_bet_list.Opp_ID
INNER JOIN      bet_styles
ON              bet_styles.Bet_Style_ID = bet_types.Bet_Style_ID
INNER JOIN      bookies
ON              bookies.Bookie_ID = possible_bets.Bookie_ID
LEFT JOIN       bet_cautions
ON              bet_cautions.Bet_Caution_ID = possible_bets.Bet_Caution_ID
WHERE           opportunities.Opp_ID = [Integer corresponding to an opportunity id]
```

```
ORDER BY        opportunity_bet_list.bet_Type_ID, possible_bets.Odds
```

View all possible bets on a single event, grouped by bet type, ordered by odds:

WARNING: The odds displayed here are “processed” to take into account any cut in winnings that the bookmaker takes. The odds shown here may not be the odds that were collected but the comparable equivalent according to the value in bookies.Bookie_Percentage.

```
SELECT          Bet_Description,
                Bet_Style_Mutator,
                Odds,
                Bookie_Name,
                Bet_URL,
                Bet_Caution_Description
FROM            possible_bets
INNER JOIN      bet_types
ON              bet_types.Bet_Type_ID = possible_bets.Bet_Type_ID
INNER JOIN      bet_styles
ON              bet_styles.Bet_Style_ID = bet_types.Bet_Style_ID
INNER JOIN      bookies
ON              bookies.Bookie_ID = possible_bets.Bookie_ID
LEFT JOIN       bet_cautions
ON              bet_cautions.Bet_Caution_ID = possible_bets.Bet_Caution_ID
INNER JOIN      events
ON              events.Event_ID = bet_types.Event_ID
WHERE           events.Event_ID = [Integer corresponding to an event id]
ORDER BY        bet_types.Bet_Type_ID, possible_bets.Odds
```

View the possible bets and stake percentages that a user needs to take advantage of a single opportunity (odds in original retrieved incomparable state, not processed comparable state):

```
SELECT          Bet_Description,
                Bet_Style_Mutator,
                ( (Odds -1) / ( ( 100 - Bookie_Percentage ) /100 ) ) +1 AS Odds,
                Stake_Percentage,
                Fetch_Timestamp,
                Bet_Caution_Description,
                Bet_URL,
                Bookie_Name,
                Bookie_URL
FROM
(SELECT         Bookie_Name,
                Bookie_URL,
                Fetch_Timestamp,
                Bet_URL,
                Bet_Style_Mutator,
                Stake_Percentage,
                Bet_Description,
                Bet_Caution_Description,
                Odds,
                Best_Odds,
                Bookie_Percentage
FROM        possible_bets
INNER JOIN  bet_types
ON          bet_types.Bet_Type_ID = possible_bets.Bet_Type_ID
INNER JOIN  opportunity_bet_list
ON          opportunity_bet_list.bet_Type_ID = bet_types.bet_Type_ID
INNER JOIN  opportunities
ON          opportunities.Opp_ID = opportunity_bet_list.Opp_ID
INNER JOIN  bet_styles
ON          bet_styles.Bet_Style_ID = bet_types.Bet_Style_ID
INNER JOIN  bookies
ON          bookies.Bookie_ID = possible_bets.Bookie_ID
LEFT JOIN   bet_cautions
ON          bet_cautions.Bet_Caution_ID = possible_bets.Bet_Caution_ID
INNER JOIN  events
ON          events.Event_ID = bet_types.Event_ID
INNER JOIN  participants
ON          events.Participant_ID_1 = participants.Participant_ID
INNER JOIN  participants participants2
ON          events.Participant_ID_2 = participants2.Participant_ID
INNER JOIN  sport_types
ON          sport_types.Sport_ID = participants.Sport_ID
WHERE       opportunities.Opp_ID = [Integer corresponding to an opportunity id]
AND         Odds = Best_Odds)
AS          temp_table
```

Appendices

G

Unit tests

Unit Tests

Under the headings of the following sections of code are a short description of the approach taken and problems found with that section of the program.

Data Storage: Database/Database Control

Description:

The database and database control were tested as the control methods and associated database queries were written. There was a main method set up in the class and each of the public methods were called individually, with valid and invalid data. The results of these were taken straight from the database (via the phpMyAdmin interface of the WAMP software), or written to the console in the IDE.

Comments:

There were issues identified with the written mySQL syntax as well as the java implementation of the written queries, most of which were easy to identify and to put right due to the small section of code being tested at any one time. Valid inputs returned as they were supposed to and invalid inputs threw exceptions where they were supposed to.

The main issue raised was the ability to insert a Null value into the database, which wasn't as easy as was first thought. Because the queries are written as string objects a null value either appeared as the string "null" in the table or threw an exception. Once these values were replaced with the string "IS NULL" this problem was resolved.

Data Storage: StringMatching

Description:

As this section of the code merely processes string objects, then checks them against each other and returns a Boolean it was very simple to test. The public methods were called from a main method and using static values that tested the processing of the strings (to get rid of unwanted characters) and the matching of them at the same time.

Comments:

This is a small but important piece of code that mainly uses java library methods so there wasn't much that could go wrong. Tests came back positive so there was no need to debug.

Data Storage: Event Identification

Description:

Tests were carried out with the use of the already tested string matching section of code. Some participant data was planted into the database so that there was an initial state to the system. Initially the purpose of testing was to make sure that the code reacted correctly to a variety of inputs; some that were meant to be accepted and some that weren't. Some of the data was suppose to be rejected on the first attempt but (following other queries) were later meant to be accepted.

Comments:

Initial tests tested the code's selection of identification method by using input data that was meant to make the code identify events using both, either, or no participant ids by using aliases that would and would not be recognised. The event start time/date was also varied. The methods called just printed out their method names and the arguments that they were called with to the screen, and all of them functioned as expected.

The second stage here was to implement the methods and allow the insertion of events, the correct identification of events and the rejection of event data due to being unable to identify both participants. Some tests were run that correctly failed to relate to a similar event due to unknown aliases, which were subsequently able to relate to it due to the increased alias list that a participant had built up during the tests in between (a demonstration that as a participant's alias list grows it will "learn" to accept more and more correct aliases). Again the code functioned as expected, as was evident from the state of the database and the returned values that were printed to the IDE console.

The first attempt at some of these tests failed due to an inconsistent database state. As the database was being manually manipulated to mimic a real state, errors in the consistency of this state (the requirement for events to have a related past event) caused errors in the program. This was a completely correct response by the program and the tests were later performed a second time with the correct database state.

Data Storage: Possible Bet Entry/Update

Description:

These tests were similar in style to the tests in the above section. Initial tests just tested the ability of the code to identify whether a bet should be inserted or updated where as later tests actually tested the ability to do this.

Comments:

Initially there were many exceptions thrown by this section of the code due to assuming valid values of passed in variables and returned variables from the database. Many extra validation points were inserted into the code to make it more robust. Some of these may be considered to be over-kill as it some of the problems were due to was injected database inconsistencies, but as it is a requirement that some data be inserted into the database manually there will always be mistakes made. It is better that these mistakes are recognised and dealt with properly in a way that highlights the issue to the user than in an ambiguous and difficult to debug way that would otherwise happen.

Data Storage: Current Market Information Manager

Description:

Tests on this section were limited as this code only really implements regular calls to methods of the database control that had already been tested. It does also catch some exceptions though so correct handling of this required testing too.

Results:

The database control methods had already been shown to work so the testing here was focussed on whether the regular calling was working correctly, which it was. The caught exceptions also contacted the centralised error reporting successfully. These were both proven again by screen output.

Data Storage: Opportunities

Description:

Opportunities were tested using static variable values as their job is to process data.

Comments:

The simple operation of these made it very easy to spot and diagnose inconsistencies.

Data Storage: Opportunity Finder

Description:

Testing this code involved database access so it was done in conjunction with the database control module. Static variables were injected and output statements were generated in the IDE output console to keep track of the process. The final database status was evaluated against expected values. The aim of the tests was to examine whether the section of code recognised which opportunities it should create when, and that it created them in the database at the correct time.

Comments:

These tests were difficult to implement as they required the database to start in a specific state. As the tests changed the database the database needed to be set up before every test, which took a lot of time and attention to detail. As a result many tests failed due to user set up mistakes. The section of code worked well though with only minor changes due to issues like only single opportunity identification when multiple opportunities were applicable.

Data Storage: Opportunity Updater

Description:

Again, the testing of this code involved database access so it was done in conjunction with the database control module. After the opportunity finder had inserted an opportunity into the database all that was required was a manual change to one of the maximum percentages of one of the opportunities bet types. The opportunity updater section of code could then be called with the bet type changed as an argument. Again, the status of the database was then checked for the results.

Comments:

All went well.

Data Storage: Centralised Error Reporting

Description:

The basic operation of this section of code was tested here, that is the ability to write the error to the log file and display dialogue boxes to the user etc.

Comments:

These two simple functions operated well, as expected, as they are common operations.

Data Acquisition: Acquisition Control/Configuration File

Description:

The main aim of this section’s tests were to ensure accurate the reading in of the configuration file, the validity checks on the read in data were ample and correct creation of the miners.

The reading in of the file was easy to test as it just involved a sample configuration file and writing the read lines out to the IDE output console.

Testing of the validation system was achieved by changing the configuration file to have a range states that included valid and sometimes invalid values. Code was again inserted to output any validation errors to the IDE output console.

Creation of the miners was tested by telling the miners to output their state upon creation.

Comments:

As the configuration file was xml a lot of the techniques used for parsing this data were developed whilst writing the xml miner code, so there were no major problems with this.

Validating the data extracted from the xml file is a difficult task as it could contain all sorts of invalid data. The only real tests possible were on the data type (e.g. making sure that what should be an integer value is an integer), the inclusion of a value for necessary fields, the validity of the file specified and the validity of the XML in that file. All of the configuration file errors mentioned have been tested and all display dialogue boxes to the user that describe the problem at the correct time. They then terminate the program as it cannot function without this initial startup information.

As invalid miner data results in the miner not being instantiated, testing the miner creation involved changing configuration file values and seeing the outputs in the IDE console that correspond to successful miner creation.

Data Acquisition: Miner Scheduler

Description:

Tests for the Miner Scheduler merely output to the IDE when the miners were selected for their next mining operation and when the mining threads actually called the mining method on the mining thread. This was an extended test over a number of hours so timestamps were output to keep track of events and so that a tester was not required to be present for the duration.

Comments:

This was just a test of time keeping and continuity that was easy to carry out and was successful at proving that the scheduling did what it was meant to.

Data Acquisition: Miners

Description:

Initially individual miners were tested by having output statements in the submission object builder code. Every time an object was supposed to be complete and fit for submission to the data storage module they would print out all of the data. Later on when log file submission was available this form of output was used instead (using the data submitter code too).

Comments:

Miners are quite complex and at least one of the sources of data changed their style whilst the miner was being written, resulting in some major changes to the code. Even when the source didn't change, testing was used throughout the miner writing process to ensure that the data was being collected correctly. As a result of testing being done during the writing of the code, there were no major problems with miner code.

Data Acquisition: Data Submitter

Description:

The data submitter, like many other parts of SAM, uses producer-consumer style management of the items being passed around, so the testing of the queue was a simple case of writing output statements to display queue status information to the screen. The log file output method was also simple to test using dummy, statically generated submission objects. Server submission to the data storage module was tested by the creation of a dummy miner, the use of which is described in the Data Storage module test.

Comments:

Initial log file tests failed due to the need for a file to be already in the set location in order to write to. Code was later written to check for the directory and file and create them if they were not found, which made a lot more sense than outputting errors just because the file was not there.

Appendices

H

Sample test schedule

Test Description:
Data Storage module tests using static data and the OD_FullTimeHandicap_Result_DrawPos opportunity definition.

Start Conditions:
Current market informatin management should be switched off to allow the Data Storage module to be switched off and on without deletion of old data in the database.
There should be bookies in the bookies table with bookie_ID = 0 and 1, both with bookie_Percentage=0.
The bet_styles, sport_types and opportunity_types tables should correspond to the constants in the SAMConstants class.
There should be entries in the participants table for Manchester United and Aston Villa (both sport type 0) and lower case entries in the alias_list table to correspond.
The following tables should be completely empty: bet_types, events, opportunities, opportunity_bet_list, past_bets, past_bet_list, past_bet_types, past_events, past_opportunities, possible_bets.

Test Data:

Test no.	Bookie_ID	Event info				Bet info		
		Part 1 alias	**Part 2 alias**	**Event_Start_Timestamp**	**Sport_ID**	**Bet_Style_ID**	**Bet_Style_Mutator**	**Odds**
1	0	Manchester United	Aston Villa	01/07/2009 15:00	0	6	-1	3.1
2	0	Man U FC	A Villa FC	01/07/2009 15:00	0	7	1	3
3	0	Man U	A Villa	02/07/2009 15:00	0	8	-1	3
4				Same as test 2				
5	1	Man United	Aston V	01/07/2009 15:00	0	6	-1	3.3
6	0	Man U	A Villa	02/07/2009 15:00	0	8	-1	2.9
7	1	Man United	Aston V	01/07/2009 15:00	0	8	-1	2.8

Test no.	Expected Outcome	Outcome Achieved
1	The Last_Poll_Start_Timestamp is updated in the bookies table.	Yes
	A past event is entered into the past_events table.	Yes
	A corresponding event is entered into the events table.	Yes
	A past bet type is entered into the past_bet_types table.	Yes
	A bet type is entered into the bet_types table.	Yes
	An entry is made in the possible_bets table.	Yes
	An entry is made in the past_bets table.	Yes
	The best_odds field in the bet_types table is updated.	Yes
	The Last_Poll_End_Timestamp is updated in the bookies table.	Yes
2	The Last_Poll_Start_Timestamp is updated in the bookies table.	Yes
	The Last_Poll_End_Timestamp is updated in the bookies table.	Yes
	Comment: Nothing else - the event is not recognosed and neither are the aliases.	Yes
3	Comment: The event is recognised as the same event from test 1.	Yes
	The Last_Poll_Start_Timestamp is updated in the bookies table.	Yes
	An entry is made in the alias_list table for "man u" with the same ID as "manchester united".	Yes
	An entry is made in the alias_list table for "a villa" with the same ID as "aston villa".	Yes
	A past bet type is entered into the past_bet_types table.	Yes
	A bet type is entered into the bet_types table.	Yes
	An entry is made in the possible_bets table.	Yes
	An entry is made in the past_bets table.	Yes
	The best_odds field in the bet_types table is updated.	Yes
	The Last_Poll_End_Timestamp is updated in the bookies table.	Yes
4	Comment: The event is now recognised as the same event from tests 1 and 3.	Yes
	The Last_Poll_Start_Timestamp is updated in the bookies table.	Yes
	An entry is made in the alias_list table for "man u" with the same ID as "manchester united".	Yes
	An entry is made in the alias_list table for "a villa" with the same ID as "aston villa".	Yes
	A past bet type is entered into the past_bet_types table.	Yes
	A bet type is entered into the bet_types table.	Yes
	An entry is made in the possible_bets table.	Yes
	An entry is made in the past_bets table.	Yes
	The best_odds field in the bet_types table is updated.	Yes
	A entry is made in the past_opportunities table.	Yes
	Three entries are made in the past_bets_list table to link the data from test 1 & 3 along with this test to the entry in the past_opportunities table.	Yes
	A entry is made in the opportunities table.	Yes
	Three entries are made in the opportunity_bet_list table to link the data from test 1 & 3 along with this test to the entry in the opportunities table.	Yes
	The Last_Poll_End_Timestamp is updated in the bookies table.	Yes

Test no.	Expected Outcome	Outcome Achieved
5	The Last_Poll_Start_Timestamp is updated in the bookies table.	Yes
	An entry is made in the alias_list table for "man united" with the same ID as "manchester united".	Yes
	An entry is made in the alias_list table for "aston v" with the same ID as "aston villa".	Yes
	An entry is made in the possible_bets table.	Yes
	An entry is made in the past_bets table.	Yes
	The best_odds field in the bet_types table is updated.	Yes
	The Opp_Percentage field in the corresponding opportunities table entry is updated.	Yes
	The Stake_Percentage fields in the corresponding opportunity_bet_list table entries is updated.	Yes
	The Max_Percentage field in the corresponding past_opportunities table entry is updated.	Yes
	The Last_Poll_End_Timestamp is updated in the bookies table.	Yes
6	The Last_Poll_Start_Timestamp is updated in the bookies table.	Yes
	The possible_bets table entry is updated.	Yes
	An entry is made in the past_bets table.	Yes
	The best_odds field in the bet_types table is updated.	Yes
	The Opp_Percentage field in the corresponding opportunities table entry is updated.	Yes
	The Stake_Percentage fields in the corresponding opportunity_bet_list table entries is updated.	Yes
	The Max_Percentage field in the corresponding past_opportunities table entry is NOT updated.	Yes
	The Last_Poll_End_Timestamp is updated in the bookies table.	Yes
7	The Last_Poll_Start_Timestamp is updated in the bookies table.	Yes
	An entry is made in the possible_bets table.	Yes
	An entry is made in the past_bets table.	Yes
	The best_odds field in the bet_types table is NOT updated.	Yes
	The Opp_Percentage field in the corresponding opportunities table entry is NOT updated.	Yes
	The Stake_Percentage fields in the corresponding opportunity_bet_list table entries is NOT updated.	Yes
	The Max_Percentage field in the corresponding past_opportunities table entry is NOT updated.	Yes
	The Last_Poll_End_Timestamp is updated in the bookies table.	Yes

Appendices

I

JavaDocs of Standard Objects and Constants

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Class Documentation

BetSubmissionObject Class Reference

BetSubmissionObjectBetSubmissionObjectInherits java::io::Serializable.Public Member Functions

- **BetSubmissionObject** (int betStyleID_in, Float betStyleMutator_in, float decimalOdds_in, String betURL_in, Integer betCautionID_in)
- **BetSubmissionObject** (int betStyleID_in, Float betStyleMutator_in, float decimalOdds_in)
- int **getBetStyleID** ()
- void **setBetStyleID** (int styleID)
- Float **getBetStyleMutator** ()
- float **getDecimalOdds** ()
- void **setDecimalOdds** (float odds)
- String **getBetURL** ()
- Integer **getBetCautionID** ()
- void **setBetCautionID** (Integer caution)
- String **toString** ()

Detailed Description

Author:

Jon Stuteley

Version:

1.00 2008/11/5

Object for holding the information required for submitting a possible bet.

Definition at line 13 of file BetSubmissionObject.java.

Constructor & Destructor Documentation

BetSubmissionObject.BetSubmissionObject (int *betStyleID_in*, Float *betStyleMutator_in*, float *decimalOdds_in*, String *betURL_in*, Integer *betCautionID_in*)

Creates a **BetSubmissionObject** the arguments passed in

Parameters:

betStyleID_in An int value representing the Bet_Style_ID.
betStyleMutator_in An java.lang.Float representing the Bet_Style_Mutator.
decimalOdds_in An float value representing the Decimal_Odds.
betURL_in An java.lang.String representing the Bet_URL.
betCautionID_in An java.lang.Integer representing the Bet_Caution_ID.

Definition at line 33 of file BetSubmissionObject.java.

BetSubmissionObject.BetSubmissionObject (int *betStyleID_in*, Float *betStyleMutator_in*, float *decimalOdds_in*)

Creates an **BetSubmissionObject** the arguments passed in and sets betURL and betCautionID to null

Parameters:

betStyleID_in An int value representing the Bet_Style_ID.
betStyleMutator_in An java.lang.Float representing the Bet_Style_Mutator.

decimalOdds_in An float value representing the Decimal_Odds.

Definition at line 57 of file BetSubmissionObject.java.

Member Function Documentation

Integer BetSubmissionObject.getBetCautionID ()

Returns the betCautionID.

Returns:

A java.lang.Integer representing the Bet_Caution_ID. (Can be null)

Definition at line 125 of file BetSubmissionObject.java.

int BetSubmissionObject.getBetStyleID ()

Returns the betStyleID

Returns:

An int value representing the Bet_Style_ID.

Definition at line 69 of file BetSubmissionObject.java.

Float BetSubmissionObject.getBetStyleMutator ()

Returns the betStyleMutator

Returns:

A java.lang.Float representing the Bet_Style_Mutator.

Definition at line 89 of file BetSubmissionObject.java.

String BetSubmissionObject.getBetURL ()

Returns the betURL

Returns:

A java.lang.String representing the Bet_URL. (Can be null.)

Definition at line 116 of file BetSubmissionObject.java.

float BetSubmissionObject.getDecimalOdds ()

Returns the decimalOdds

Returns:

A float value representing the Decimal_Odds.

Definition at line 98 of file BetSubmissionObject.java.

void BetSubmissionObject.setBetCautionID (Integer *caution*)

Sets the bet caution ID value.

Parameters:

caution A java.lang.Integer representing a **SAMConstants** value for bet cautions.

Definition at line 135 of file BetSubmissionObject.java.

void BetSubmissionObject.setBetStyleID (int *styleID*)

sets the bet style id

Parameters:

styleID An int value representing a bet style id. Usually a **SAMConstants** value.

Definition at line 79 of file BetSubmissionObject.java.

void BetSubmissionObject.setDecimalOdds (float *odds*)

Sets the decimalOdds

Parameters:

odds A float value representing the Decimal_Odds.

Definition at line 107 of file BetSubmissionObject.java.

String BetSubmissionObject.toString ()

Returns:

A java.lang.String representation of this object.

Definition at line 158 of file BetSubmissionObject.java.

The documentation for this class was generated from the following file:

- C:/Users/Jon/Desktop/UNI/SPRING09/Dissertation/Dissertation/Java/SAM/Objects/src/BetSubmissionObject.java

DataStorageInterface Interface Reference

DataStorageInterfaceDataStorageInterfaceInherits java::rmi::Remote.Public Member Functions

- void **submitData** (Vector< **SubmissionObject** > soV) throws RemoteException

---

Detailed Description

Author:

jon stuteley

Version:

1.00 2008/12/18

This interface defines the way that clients interact with SAM. They should call setBookiePollStart, submitData, then setBookiePollEnd when finished. If they do not finish before the built in cut-off (currently 30 secs) the system cuts them off automatically to stop a lost connection restricting the system for too long.

Definition at line 17 of file DataStorageInterface.java.

Member Function Documentation

void DataStorageInterface.submitData (Vector< SubmissionObject > *soV*) throws RemoteException

submits data to the database.

Parameters:

soV java.util.Vector of **SubmissionObject** objects used to submit data.

The documentation for this interface was generated from the following file:

- C:/Users/Jon/Desktop/UNI/SPRING09/Dissertation/Dissertation/Java/SAM/Objects/src/DataStorageInterface.java

EventSubmissionObject Class Reference

EventSubmissionObjectEventSubmissionObjectInherits java::io::Serializable.Public Member Functions

- **EventSubmissionObject** (String alias_1_in, String alias_2_in, Timestamp eventStartTimestamp_in, String eventDescription_in, int sportID_in)
- **EventSubmissionObject** (String alias_1_in, String alias_2_in, Timestamp eventStartTimestamp_in, int sportID_in)
- String **getAlias** (int num)
- void **setAlias** (int num, String alias_in)
- Timestamp **getTimestamp** ()
- String **getDescription** ()
- int **getSportID** ()
- String **toString** ()

Detailed Description

Author:

Jon Stuteley

Version:

1.00 2008/11/5

Object for holding the information required for submitting an event

Definition at line 15 of file EventSubmissionObject.java.

Constructor & Destructor Documentation

EventSubmissionObject.EventSubmissionObject (String *alias_1_in*, String *alias_2_in*, Timestamp *eventStartTimestamp_in*, String *eventDescription_in*, int *sportID_in*)

Creates an **EventSubmissionObject** with the arguments passed in.

Parameters:

alias_1_in A java.lang.String representing the possible alias of a Participant_ID_1.
alias_2_in A java.lang.String representing the possible alias of a Participant_ID_2.
eventStartTimestamp_in A java.sql.Timestamp representing the Event_Start_Timestamp.
eventDescription_in A java.lang.String representing the Event_Description. (Can be null)
sportID_in A int value representing the participant's Sport_ID.

Definition at line 32 of file EventSubmissionObject.java.

EventSubmissionObject.EventSubmissionObject (String *alias_1_in*, String *alias_2_in*, Timestamp *eventStartTimestamp_in*, int *sportID_in*)

Creates an **EventSubmissionObject** with the arguments passed in and eventDescription = null

Parameters:

alias_1_in A java.lang.String representing the possible alias of a Participant_ID_1.
alias_2_in A java.lang.String representing the possible alias of a Participant_ID_2.
eventStartTimestamp_in A java.sql.Timestamp representing the Event_Start_Timestamp.
sportID_in A int value representing the participant's Sport_ID.

Definition at line 51 of file EventSubmissionObject.java.

Member Function Documentation

String EventSubmissionObject.getAlias (int *num*)

Returns the alias referenced by num

Parameters:

num An int value set to "0" to get alias 1 or "1" to get alias 2.

Returns:

A java.lang.String representing the possible alias of a Participant_ID_1 or Participant_ID_2.

Definition at line 63 of file EventSubmissionObject.java.

String EventSubmissionObject.getDescription ()

Returns a String representing description (other info) of the event

Returns:

A java.lang.String representing the Event_Description. (Can be null)

Definition at line 94 of file EventSubmissionObject.java.

int EventSubmissionObject.getSportID ()

Returns the Sport_ID associated with this event (participants).

Returns:

An int value representing Sport_ID.

Definition at line 103 of file EventSubmissionObject.java.

Timestamp EventSubmissionObject.getTimestamp ()

Returns a Timestamp representing the start of the event.

Returns:

A java.sql.Timestamp representing the Event_Start_Timestamp.

Definition at line 85 of file EventSubmissionObject.java.

void EventSubmissionObject.setAlias (int *num*, String *alias_in*)

Sets alias num to alias

Parameters:

num An int value set to "0" to set alias 1 or 1 to set alias 2.
alias_in A java.lang.String representing an alias.

Definition at line 75 of file EventSubmissionObject.java.

String EventSubmissionObject.toString ()

Returns:

A java.lang.String representing this object.

Definition at line 114 of file EventSubmissionObject.java.

The documentation for this class was generated from the following file:

- C:/Users/Jon/Desktop/UNI/SPRING09/Dissertation/Dissertation/Java/SAM/Objects/src/EventSubmissionObject.ja va

SAMConstants Class Reference

SAMConstantsSAMConstantsStatic Public Attributes

- static final int **FT_Result_DrawPoss_PARTICIPANT_1_WIN** = 0
- static final int **FT_Result_DrawPoss_PARTICIPANT_2_WIN** = 1
- static final int **FT_Result_DrawPoss_DRAW** = 2
- static final int **FT_AsianHandicap_PARTICIPANT_1_WIN** = 4
- static final int **FT_AsianHandicap_PARTICIPANT_2_WIN** = 5
- static final int **FT_Handicap_PARTICIPANT_1_WIN** = 6
- static final int **FT_Handicap_PARTICIPANT_2_WIN** = 7
- static final int **FT_Handicap_DRAW** = 8
- static final int **FT_DoubleChance_PARTICIPANT_1_WIN_OR_DRAW** = 10
- static final int **FT_DoubleChance_PARTICIPANT_2_WIN_OR_DRAW** = 11
- static final int **FT_DoubleChance_PARTICIPANT_1_OR_2_WIN** = 12
- static final int **SPORT_TYPE_FOOTBALL** = 0
- static final int **SPORT_TYPE_RUGBY_UNION** = 1
- static final int **SPORT_TYPE_RUGBY_LEAGUE** = 2
- static final int **SPORT_TYPE_TENNIS** = 3
- static final int **SPORT_TYPE_BASKETBALL** = 4
- static final int **ARBITRAGE** = 0
- static final int **MIDDLE_2** = 1
- static final int **MIDDLE_5** = 2
- static final int **MIDDLE_20** = 3
- static final int **LOW_NO_PROFIT** = 0
- static final int **EVENT_TRANSPOSED** = 1

Detailed Description

Author:

jon Stuteley

Version:

1.00 2008/11/17

A collection of Constants used to easily reference data in SAM.

Definition at line 11 of file SAMConstants.java.

Member Data Documentation

final int SAMConstants.ARBITRAGE = 0 `[static]`

Describes the Opp_Type_ID variable associated with an "arbitrage"

Definition at line 78 of file SAMConstants.java.

final int SAMConstants.EVENT_TRANSPOSED = 1 `[static]`

Describes the bet_caution_ID variable associated with "Event was transposed"

Definition at line 97 of file SAMConstants.java.

final int SAMConstants.FT_AsianHandicap_PARTICIPANT_1_WIN = 4 `[static]`

Describes the bet associated with an Asian Handicap participant #1 win

Definition at line 23 of file SAMConstants.java.

final int SAMConstants.FT_AsianHandicap_PARTICIPANT_2_WIN = 5 `[static]`

Describes the bet associated with an Asian Handicap participant #2 win

Definition at line 26 of file SAMConstants.java.

final int SAMConstants.FT_DoubleChance_PARTICIPANT_1_OR_2_WIN = 12 `[static]`

Describes the bet associated with a Double Chance participant 1 Win or participant 2 Win

Definition at line 44 of file SAMConstants.java.

final int SAMConstants.FT_DoubleChance_PARTICIPANT_1_WIN_OR_DRAW = 10 `[static]`

Describes the bet associated with a Double Chance participant 1 Win or draw

Definition at line 38 of file SAMConstants.java.

final int SAMConstants.FT_DoubleChance_PARTICIPANT_2_WIN_OR_DRAW = 11 `[static]`

Describes the bet associated with a Double Chance participant 2 Win or draw

Definition at line 41 of file SAMConstants.java.

final int SAMConstants.FT_Handicap_DRAW = 8 `[static]`

Describes the bet associated with a Handicap draw

Definition at line 35 of file SAMConstants.java.

final int SAMConstants.FT_Handicap_PARTICIPANT_1_WIN = 6 `[static]`

Describes the bet associated with a Handicap participant #1 win

Definition at line 29 of file SAMConstants.java.

final int SAMConstants.FT_Handicap_PARTICIPANT_2_WIN = 7 `[static]`

Describes the bet associated with a Handicap participant #2 win

Definition at line 32 of file SAMConstants.java.

final int SAMConstants.FT_Result_DrawPoss_DRAW = 2 `[static]`

Describes the bet associated with an event resulting in a draw (tie)

Definition at line 20 of file SAMConstants.java.

final int SAMConstants.FT_Result_DrawPoss_PARTICIPANT_1_WIN = 0 `[static]`

Describes the bet associated with an event resulting in a participant #1 win

Definition at line 14 of file SAMConstants.java.

final int SAMConstants.FT_Result_DrawPoss_PARTICIPANT_2_WIN = 1 `[static]`

Describes the bet associated with an event resulting in a participant #2 win

Definition at line 17 of file SAMConstants.java.

final int SAMConstants.LOW_NO_PROFIT = 0 `[static]`

Describes the bet_caution_ID variable associated with"Low/No bookie profit"

Definition at line 94 of file SAMConstants.java.

final int SAMConstants.MIDDLE_2 = 1 `[static]`

Describes the Opp_Type_ID variable associated with an "middle" with profitable threshold 2%

Definition at line 81 of file SAMConstants.java.

final int SAMConstants.MIDDLE_20 = 3 `[static]`

Describes the Opp_Type_ID variable associated with an "middle" with profitable threshold 20%

Definition at line 87 of file SAMConstants.java.

final int SAMConstants.MIDDLE_5 = 2 `[static]`

Describes the Opp_Type_ID variable associated with an "middle" with profitable threshold 5%

Definition at line 84 of file SAMConstants.java.

final int SAMConstants.SPORT_TYPE_BASKETBALL = 4 `[static]`

Describes the Sport_Type associated with BasketBall

Definition at line 64 of file SAMConstants.java.

final int SAMConstants.SPORT_TYPE_FOOTBALL = 0 `[static]`

Describes the Sport_Type associated with Football (Soccer)

Definition at line 52 of file SAMConstants.java.

final int SAMConstants.SPORT_TYPE_RUGBY_LEAGUE = 2 `[static]`

Describes the Sport_Type associated with Rugby League

Definition at line 58 of file SAMConstants.java.

final int SAMConstants.SPORT_TYPE_RUGBY_UNION = 1 `[static]`

Describes the Sport_Type associated with Rugby Union

Definition at line 55 of file SAMConstants.java.

final int SAMConstants.SPORT_TYPE_TENNIS = 3 `[static]`

Describes the Sport_Type associated with Tennis

Definition at line 61 of file SAMConstants.java.

The documentation for this class was generated from the following file:

- C:/Users/Jon/Desktop/UNI/SPRING09/Dissertation/Dissertation/Java/SAM/Objects/src/SAMConstants.java

SubmissionObject Class Reference

SubmissionObjectSubmissionObjectInherits java::io::Serializable.Public Member Functions

- **SubmissionObject** (**EventSubmissionObject** event, Vector< **BetSubmissionObject** > betV, int bookieID)
- **EventSubmissionObject getEventSO** ()
- Vector< **BetSubmissionObject** > **getBetSOV** ()
- int **getBetSize** ()
- int **getBookieID** ()
- String **toString** ()

Detailed Description

Author:
jon stuteley

Version:
1.00 2008/11/14

Object for holding an **EventSubmissionObject** and a java.util.Vector of **BetSubmissionObject** objects in a single package.

Definition at line 13 of file SubmissionObject.java.

Constructor & Destructor Documentation

SubmissionObject.SubmissionObject (EventSubmissionObject *event*, Vector< BetSubmissionObject > *betV*, int *bookieID*)

Creates a **SubmissionObject** using the given argumants.

Parameters:

event An **EventSubmissionObject** representing all the data needed to locate/insert an event.
betV A java.util.Vector of **BetSubmissionObject** objects representing all the bets to be submitted on the event.
bookieID An int value representing the id of the bookie, corresponding to the value in the bookies table of the SAM database.

Definition at line 27 of file SubmissionObject.java.

Member Function Documentation

int SubmissionObject.getBetSize ()

Returns:

An int representing the number of bets in this object.

Definition at line 56 of file SubmissionObject.java.

Vector<BetSubmissionObject> SubmissionObject.getBetSOV ()

Returns the contained java.util.Vector of **BetSubmissionObject** objects

Returns:

A java.util.Vector of **BetSubmissionObject** objects representing all the bets to be submitted on the event.

Definition at line 49 of file SubmissionObject.java.

int SubmissionObject.getBookieID ()

Returns:

An int representing the id of the bookie that this data is from.

Definition at line 64 of file SubmissionObject.java.

EventSubmissionObject SubmissionObject.getEventSO ()

Returns the contained **EventSubmissionObject**.

Returns:

An **EventSubmissionObject** representing all the data needed to locate/insert an event.

Definition at line 38 of file SubmissionObject.java.

String SubmissionObject.toString ()

Returns:

A java.lang.String representation of this object.

Definition at line 73 of file SubmissionObject.java.

The documentation for this class was generated from the following file:

C:/Users/Jon/Desktop/UNI/SPRING09/Dissertation/Dissertation/Java/SAM/Objects/src/

J

Example Data Acquisition module configuration file

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <!--
     This section contains an example configuration file with comments in square brackets[].
     <Definitions>
     [root tag]

             <System>
             [marks the section associated with non-miner information]

                     <Connection>
                     [marks the section relating to networking]

                             <ServerAddress>localhost</ServerAddress>
                             [The location of a Data Storage module server. Not essential.
                                     If missing then a log file submitter is created whatever is in the OutputToLog tag.]

                             <ServerPort>1099</ServerPort>
                             [The port of a Data Storage module server. Not essential. If missing then a
                                     log file submitter is created whatever is in the OutputToLog tag.]

                             <RMIInterfaceName>DataStorageInterface</RMIInterfaceName>
                             [This is the name that is bound to the Data Aquisition module and registered with
                                     the RMI registry. Essential.]

                             <ProxyHost>wwwcache-20.cs.nott.ac.uk</ProxyHost>
                             [Sets a proxy host for the network connection. Not Essential.]

                             <ProxyPort>3128</ProxyPort>
                             [Sets a proxy port for the network connection. Not Essential.]

                     </Connection>

                     <Configure>
                     [marks the section relating to system variable settings]

                             <MiningThreadCount>3</MiningThreadCount>
                             [Sets the number of threads of execution that are used to mine with.  Essential.]

                             <FailuresBeforeMinerReset>3</FailuresBeforeMinerReset>
                             [Sets the number of times a miner is asked to mine and fails before the miner is reset.  Essential.]

                             <OutputToLog>false</OutputToLog>
                             [Specifies output to log.  Any value but "true" means false. Automatically
                                     true if ServerAddress or ServePort are missing. Not Essential.]

                     </Configure>
```

```
        </System>

        <MinerDef>
        [marks the start of a miner definition. Multiple tags allowed, one for each miner required.]

                <ProcessingClass>CentrebetMiner</ProcessingClass>
                [The name of the class that will be used to mine]

                <URL>http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_England.xml</URL>
                [the url to mine.  Be careful of non-alphanumeric characters. ";" needs to be "&amp"?]

                <SportType>0</SportType>
                [If this is a sport specific url and the miner requires a sport type specified do it here.
                        Not Essential (for all miners but is for CentrebetMiner)]

                <Period>200</Period>
                [period between starting mining operations.  If this is too short it may lead to miner being reset.]

                <FullRefreshOnPeriod>10</FullRefreshOnPeriod>
                [Some miners change the url they mine each time. This value tells them to start from the url
                        in this file every x mines. Pinnacle for example.]

        </MinerDef>
  -->
- <Definitions>
  - <System>
    - <Connection>
        <ServerAddress>localhost</ServerAddress>
        <ServerPort>1099</ServerPort>
        <RMIInterfaceName>DataStorageInterface</RMIInterfaceName>
        <!-- <ProxyHost>wwwcache-20.cs.nott.ac.uk</ProxyHost>  -->
        <!-- <ProxyPort>3128</ProxyPort>  -->
      </Connection>
    - <Configure>
        <MiningThreadCount>3</MiningThreadCount>
        <FailuresBeforeMinerReset>3</FailuresBeforeMinerReset>
        <OutputToLog>false</OutputToLog>
      </Configure>
    </System>
  - <MinerDef>
      <ProcessingClass>CentrebetMiner</ProcessingClass>
      <URL>http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_England.xml</URL>
    <SportType>0</SportType>
    <Period>200</Period>
    <FullRefreshOnPeriod />
  </MinerDef>
- <MinerDef>
    <ProcessingClass>CentrebetMiner</ProcessingClass>
    <URL>http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_Scotland.xml</URL>
    <SportType>0</SportType>
    <Period>200</Period>
    <FullRefreshOnPeriod />
  </MinerDef>
- <MinerDef>
    <ProcessingClass>BetAtHomeMiner</ProcessingClass>
    <URL>http://www.bet-at-home.com/oddxml.aspx?extended=true</URL>
    <SportType />
    <Period>240</Period>
    <FullRefreshOnPeriod />
  </MinerDef>
- <MinerDef>
    <ProcessingClass>PinnacleMiner</ProcessingClass>
    <URL>http://xml.pinnaclesports.com/xmlfeed.asp</URL>
    <SportType />
    <Period>180</Period>
    <FullRefreshOnPeriod>10</FullRefreshOnPeriod>
  </MinerDef>
- <MinerDef>
    <ProcessingClass>SkyBetMiner</ProcessingClass>
    <URL>http://www.skybet.com/skybet?action=GoEvCoupon&id=10010346</URL>
    <SportType>0</SportType>
    <Period>1200</Period>
    <FullRefreshOnPeriod />
  </MinerDef>
</Definitions>
```

Appendices

K

SAM Installation Manual

SAM Installation Manual

Introduction

This installation manual covers the installation of all SAM (Sports Arbitrage and More) components. The Preparation instructions must be followed before installation of the Data Storage module or the Data Acquisition module can be started.

Preparation

These instructions assume that the target computer for installation has the below minimum system requirements and has the below software installed prior to starting the installation procedures.

Recommended System Requirements

Microsoft Windows Vista or XP.
1GHz Processor.
512K RAM.
200Mb HD Space (for operating data).
Broadband network connection.

Software

The software listed here was used to develop SAM. Other IDEs and JDKs are not guaranteed to be compatible.

IDE:	JCreater (LE)
JDK:	Version 1.6.0_11 *

Any mainstream html browser.

*It is recommended that the 32bit versions of all software (apart from operating system) is used even if the installation computer has 64 bit architecture, to ensure compatibility.

The installation software is bundled in a folder named SAM, which contains all of the files required to start using SAM.

Step-by-Step

1. Transfer the SAM folder to the preferred location on the installation system's hard disk.

Data Storage Module Installation and Configuration

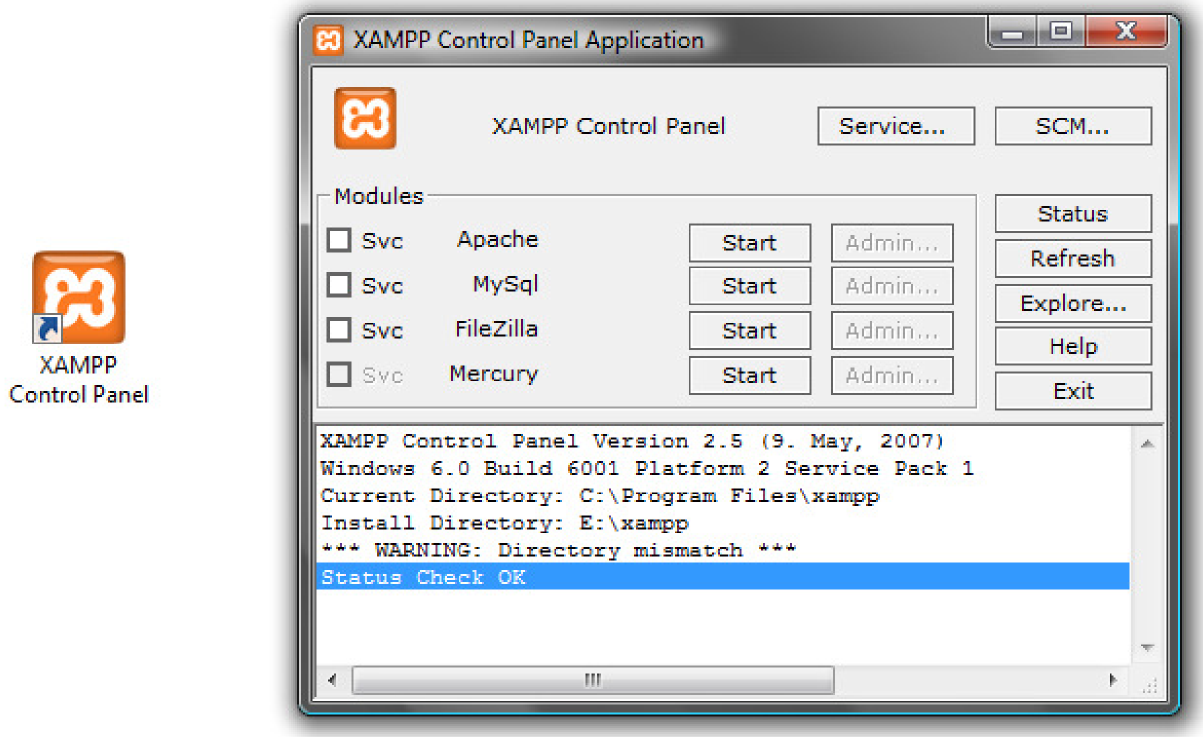
Step-by-Step

Installation of XAMPP Software.

1. An installation file for XAMPP v1.7.0 is included in the "Extras" folder (xampp-win32-1.7.0-installer.exe). This should be ran and the on-screen instructions followed until successful installation is complete.

XAMPP Setup

2. Start XAMPP using the start menu or desktop icon (left of figure 1.1). This will show the XAMPP Control Panel (right of figure 1.1).



```
XAMPP Control Panel Version 2.5 (9. May, 2007)
Windows 6.0 Build 6001 Platform 2 Service Pack 1
Current Directory: C:\Program Files\xampp
Install Directory: E:\xampp
*** WARNING: Directory mismatch ***
Status Check OK
```

Figure 1.1 – XAMPP Control Panel.

3. Press the “Start” buttons for MySQL and Apache followed by the “Admin…” button for Apache. The “Start” button text changes to “Stop” and “Running” messages appear to the left of them.
4. At this point there may be warning messages generated by firewall software (if installed). Allow these operations.
5. The system’s default web browser should have opened with a language select page showing. Please select the preferred language.
6. The next page shown has a menu bar on the left hand side (figure 1.2). Select “phpMyAdmin” from the “Tools” menu.

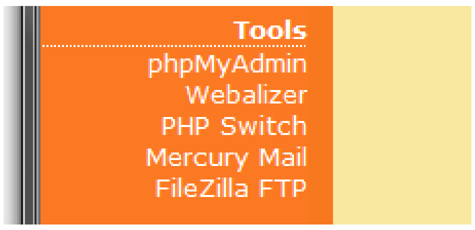
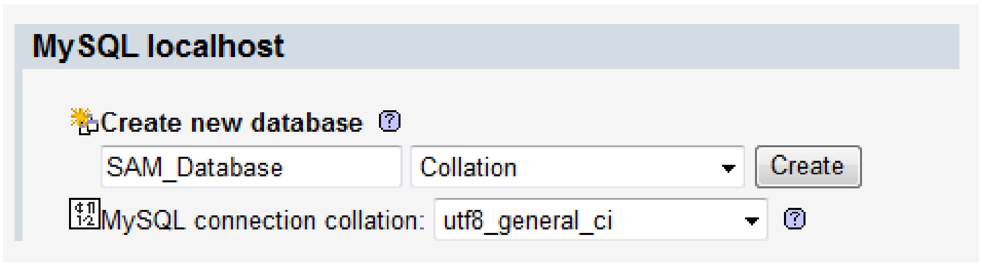


Figure 1.2 – XAMPP Tools Menu.

7. Showing now is the phpMyAdmin interface that can be used to view and manipulate the database manually. Create a new database called “SAM_Database” using the area in the middle of the screen (figure 1.3). Leave the other options as they are and press the “Create” button to create a new empty database.


MySQL localhost

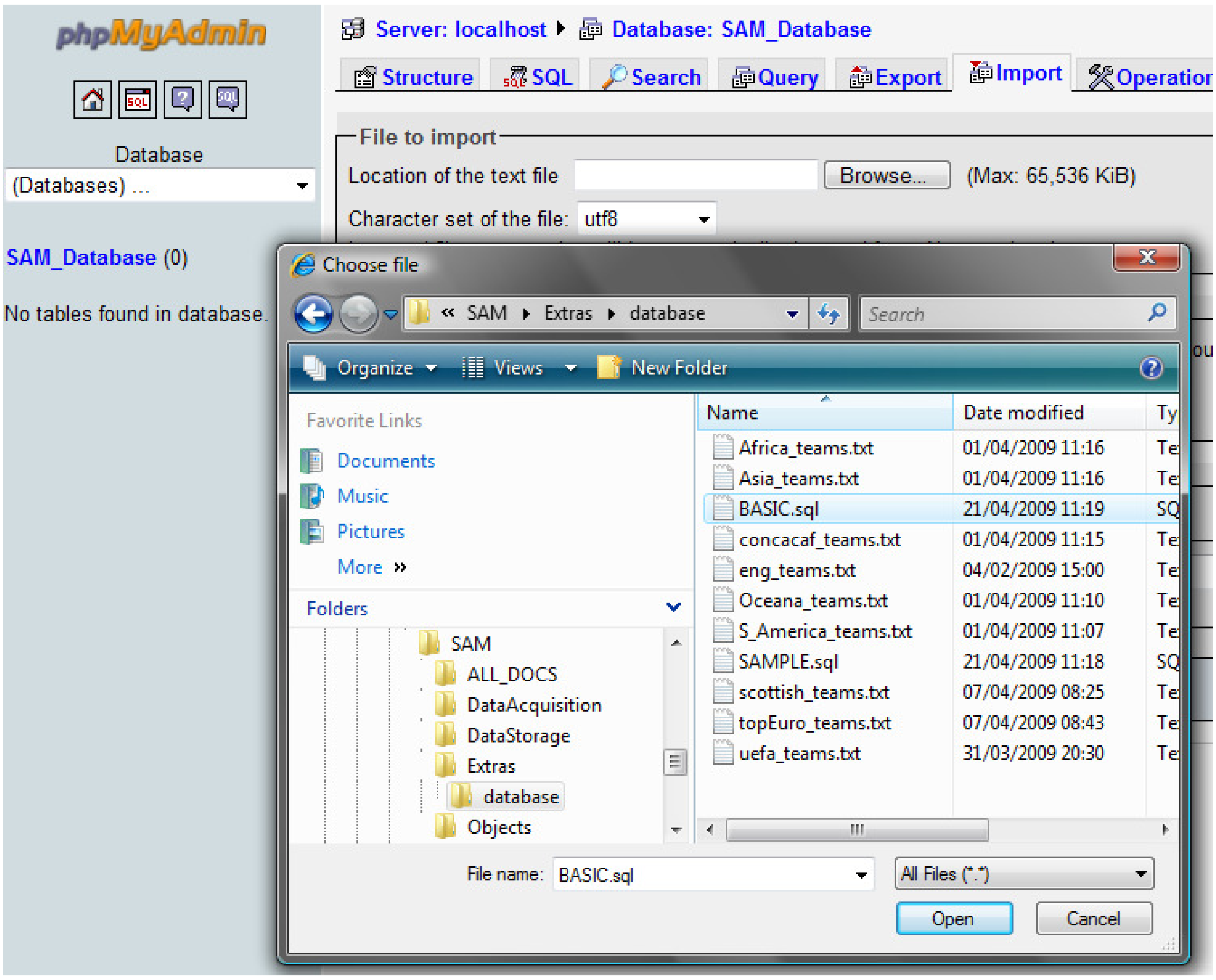
Create new database

SAM_Database | Collation | Create

MySQL connection collation: utf8_general_ci

Figure 1.3 – Create Database

8. Next we need to import the basic database configuration from a supplied file, so carry out the following steps (see figure 1.4):
   a. Select the "Import" tab towards the top of the screen.
   b. Select the "Browse" button.
   c. The file containing the standard SAM database information is "SAM\Extras\database\BASIC.sql". Navigate here, select the file and select "Open" on the file chooser. Next select "Go" on the browser page.



File to import

Location of the text file | Browse... | (Max: 65,536 KiB)

Character set of the file: utf8

Name	Date modified	Ty
Africa_teams.txt	01/04/2009 11:16	Te
Asia_teams.txt	01/04/2009 11:16	Te
BASIC.sql	21/04/2009 11:19	SQ
concacaf_teams.txt	01/04/2009 11:15	Te
eng_teams.txt	04/02/2009 15:00	Te
Oceana_teams.txt	01/04/2009 11:10	Te
S_America_teams.txt	01/04/2009 11:07	Te
SAMPLE.sql	21/04/2009 11:18	SQ
scottish_teams.txt	07/04/2009 08:25	Te
topEuro_teams.txt	07/04/2009 08:43	Te
uefa_teams.txt	31/03/2009 20:30	Te

File name: BASIC.sql | All Files (*.*)

Figure 1.4 – Database Import.

9. If successful, a list of database tables will appear on the left hand side of the page, under "SAM_Database" that should have the number "19" in brackets beside it.

The new database has the default access account with username : "root" and a blank password (no characters).

JDBC (Java Database Connectivity) Driver Installation

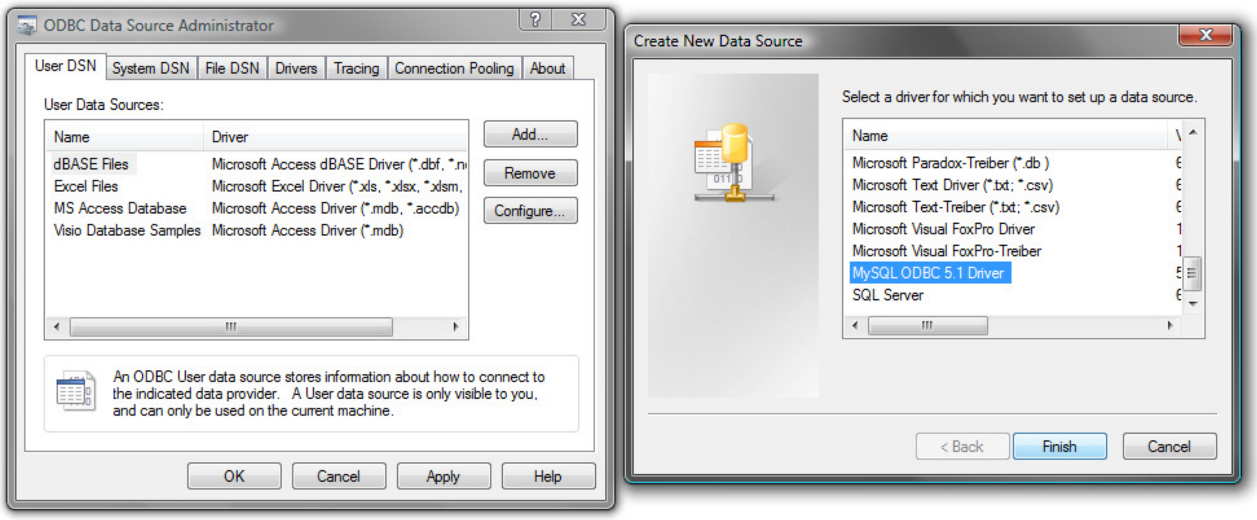
10. The installation file for this driver is "mysql-connector-odbc-5.1.5-win32.msi" it can be ran by double clicking on the file in a file chooser window (as the database file choice could have been made above). This will install the necessary software for a SAM Data Storage module to access the database that has just been created, once it has registered in the next step.

Database ODBC (Open Database Connectivity) Registration

11. For 32bit operating system users there is a shortcut to the registration application in the Control Panel/Administrative Tools folder named "Data Sources (ODBC)".

12. 64bit operating system users may find it a little harder to locate the 32bit version of this program as the one in the location mentioned above is the 64bit version , which is not compatible with the other 32bit software used. On Windows Vista 64bit the program file is located here: "C:\Windows\SysWOW64\odbcad32.exe". Once found it is advisable to create a shortcut to this in the Administrative Tools folder mentioned above.

13. Once the ODBC program has been started, an interface will appear on the screen that allows registration of a database. Select the "System DSN" (to register for access by all computer users) or the "User DSN" (access by this user only) tab and hit the "Add…" button to select the JDBC driver that was just installed. Select "MySQL ODBC 5.1 Driver" and press the "Finish" button to complete driver selection (figure 1.5).



User Data Sources:

Name	Driver
dBASE Files	Microsoft Access dBASE Driver (*.dbf, *.n
Excel Files	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm,
MS Access Database	Microsoft Access Driver (*.mdb, *.accdb)
Visio Database Samples	Microsoft Access Driver (*.mdb)

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

Select a driver for which you want to set up a data source.

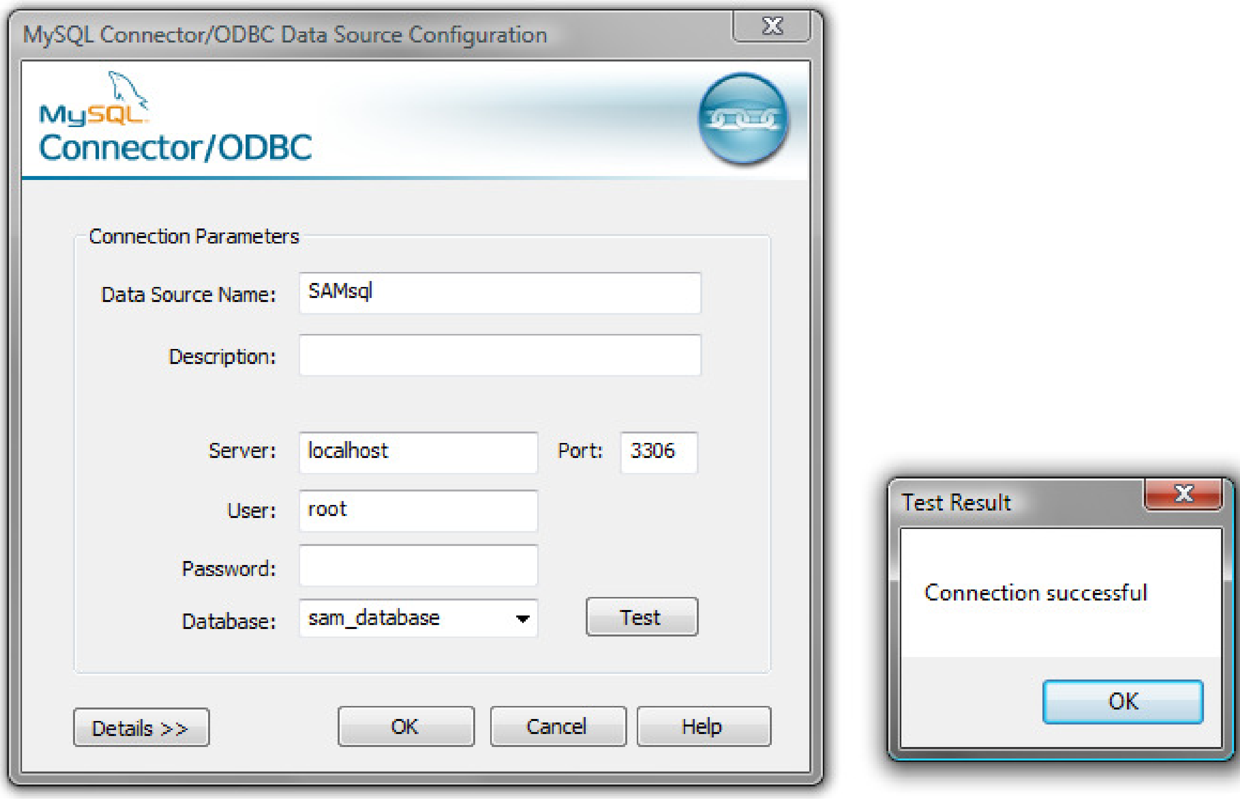
Name
Microsoft Paradox-Treiber (*.db)
Microsoft Text Driver (*.txt; *.csv)
Microsoft Text-Treiber (*.txt; *.csv)
Microsoft Visual FoxPro Driver
Microsoft Visual FoxPro-Treiber
MySQL ODBC 5.1 Driver
SQL Server

Figure 1.5 – ODBC Driver Selection.

14. Configure the new connection (as shown in figure 1.6) with the following data:

Data Source Name:	SAMsql
Server:	localhost
Port:	3306 (default)
User:	root
Password:	<Left Blank>

Assuming that XAMPP is still running from earlier steps, test the connection to the database by pressing the “Test” button. The result should be successful as depicted in figure 1.6.



MySQL Connector/ODBC Data Source Configuration

Connection Parameters

Data Source Name:	SAMsql
Description:	
Server:	localhost
Port:	3306
User:	root
Password:	
Database:	sam_database

Test Result

Connection successful

Figure 1.6 – ODBC Connection Configuration.

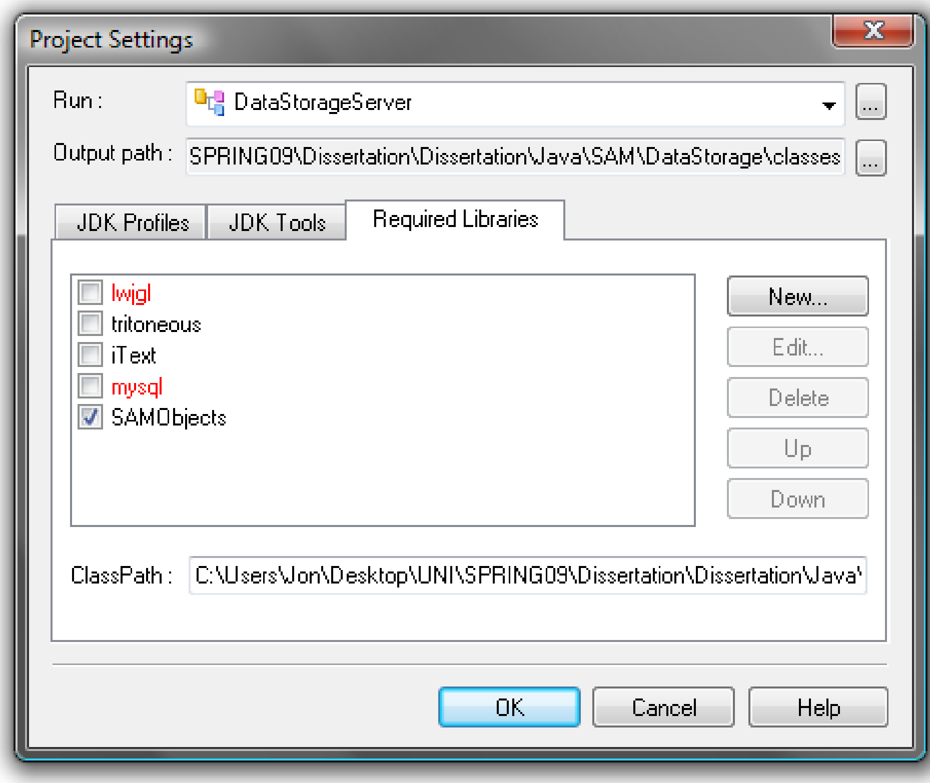
15. Clicking “OK” finishes this database registration section of the Data Storage Module installation. The ODBC interface can be closed.

XAMPP can be left running if running the SAM Data Storage Module is intended.

Closing XAMPP involves pressing the “Stop” buttons on the control panel next to “Apache” and “MySQL” (where the “Start” buttons were in figure 1.1) followed by the “Exit” button.

SAM Data Storage Module Setup (JCreator)

16. The Data storage module needs its references configured to work in the new environment. To do this first open up the project file in the supplied JCreator workspace by opening SAM\DataStorage\DataStorage.jcw in JCreator. Next, select the “Project Settings” option on the “Project” drop-down menu (on the JCreator menu bar). The display should look like figure 1.7.


Project Settings

Run : DataStorageServer

Output path : SPRING09\Dissertation\Dissertation\Java\SAM\DataStorage\classes

- [ ] lwjgl
- [ ] tritoneous
- [ ] iText
- [ ] mysql
- [x] SAMObjects

ClassPath : C:\Users\Jon\Desktop\UNI\SPRING09\Dissertation\Dissertation\Java\

Figure 1.7 – Data Storage Setup.

17. First of all make sure that it says "DataStorageServer" in the "Run" box at the top of the window. If it doesn't then try selecting it from the drop-down menu. If it is not an available selection then:
    a. Press the "Cancel" button.
    b. Make sure the "Package View" is visible on the left hand side of JCreator (if not press ctrl + alt + p).
    c. Expand the folders and double click the "DataStorageServer" class to open the file.
    d. Go back into "Project Settings" and "DataStorageServer" should be in the "Run" box.

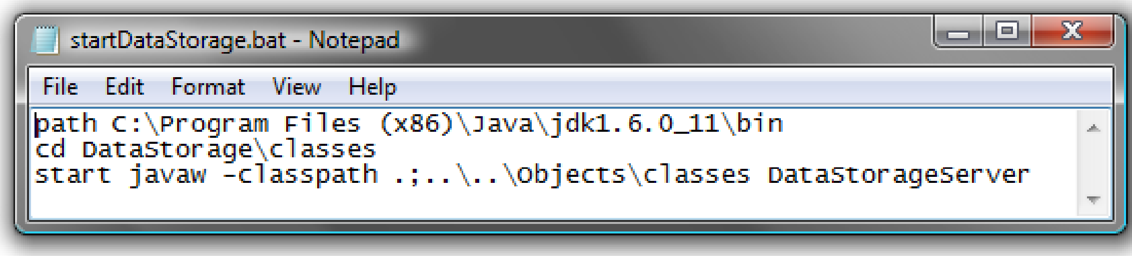
18. Select the "Required Libraries" tab. The "SAMObjects" entry should be BLACK to signify that it references a location that is valid. If the entry is RED it needs referencing to the folder containing the SAM Objects; SAM\Objects\classes. This is achieved by:
    a. Selecting the SAMObjects entry.
    b. Pressing the "Edit..." button on the right.
    c. Selecting the "Add" button then "Add Path".
    d. Selecting the "SAM\Objects\classes" folder.
    e. Selecting and deleting the old "RED" path.
    f. Clicking "OK" to get back to the Project Settings screen (figure 1.7).

    Now, making sure that the SAMObjects checkbox is ticked and the text is BLACK, the Project Settings "OK" button can be pressed to exit this window.

19. The project can now be re-compiled by pressing F7 to "Build Project".

SAM Data Storage Module Setup (Command prompt)

20. The batch file entitled "startDataStorage.bat" in the SAM folder can be used to run the Data Storage module without the aid of an IDE, but configuration of the jdk folder reference is require. To do this (see figure 1.8):
    a. Open SAM\startDataStorage.bat in a simple text editor.
    b. Change the path on the first line to reflect the "bin" folder of the jdk
    c. Save the file.



```
path C:\Program Files (x86)\Java\jdk1.6.0_11\bin
cd DataStorage\classes
start javaw -classpath .;..\..\Objects\classes DataStorageServer
```

Figure 1.8 – Batch File.

RMI Registry Batch File Configuration

21. This step is almost identical to step 20, only using a the file "startRMIReg.bat". Change the reference to the jdk "bin" folder in the same way as in this step.

This Installation of the SAM Data Storage module is now complete. Please refer to the User Manual for information about how to start the module running.

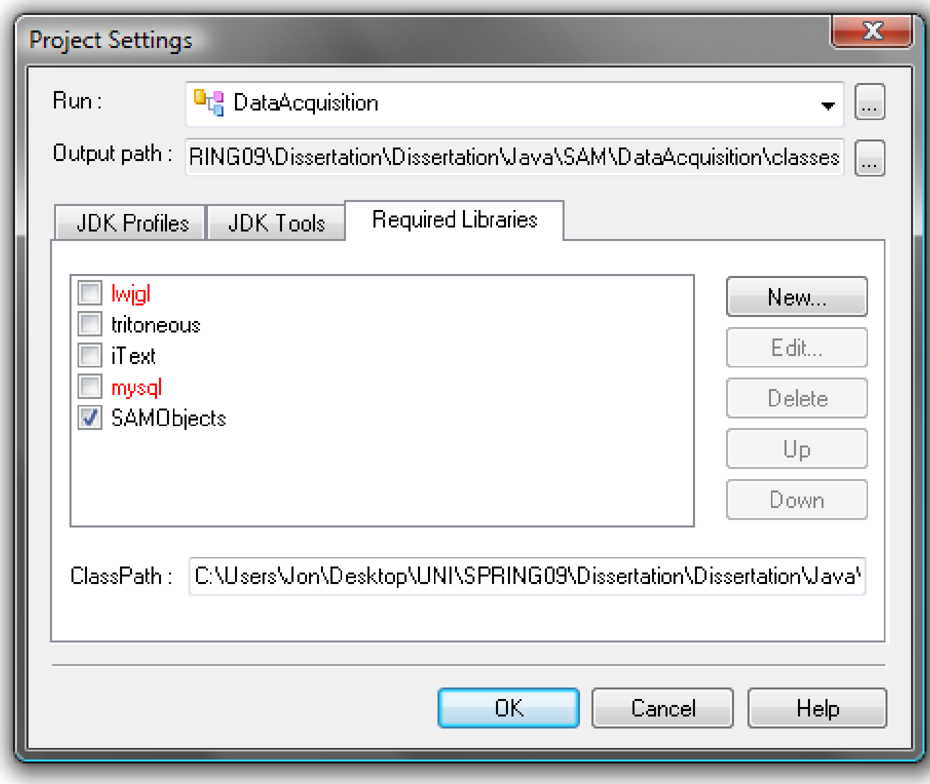
Data Acquisition Module Installation and Configuration

This procedure is exactly the same as the "SAM Data Storage Module Setup" section above but with a few changes to names. It has been re-written here with those changes for completeness.

Step-by-Step

SAM Data Acquisition Module Setup (JCreator)

1. The Data Acquisition module needs its references configured to work in the new environment. To do this first open up the project file in the supplied JCreator workspace by opening SAM\DataAcquisition\DataAcquisition.jcw in JCreator. Next select the "Project Settings" option on the "Project" drop-down menu (on the JCreator menu bar). The display should look like figure 1.9.



Run : DataAcquisition

Output path : RING09\Dissertation\Dissertation\Java\SAM\DataAcquisition\classes

- [ ] lwjgl
- [ ] tritoneous
- [ ] iText
- [ ] mysql
- [x] SAMObjects

ClassPath : C:\Users\Jon\Desktop\UNI\SPRING09\Dissertation\Dissertation\Java\

Figure 1.9 – Data Acquisition Setup.

2. First of all make sure that it says “DataAcquisition” in the “Run” box at the top of the window. If it doesn’t then try selecting it from the drop-down menu. If it is not an available selection then:
   a. Press the “Cancel” button.
   b. Make sure the “Package View” is visible on the left hand side of JCreator (if not press ctrl + alt + p).
   c. Expand the folders and double click the “DataAcquisition” class to open the file.
   d. Go back into “Project Settings” and “DataAcquisition” should be in the “Run” box.

3. Select the “Required Libraries” tab. The “SAMObjects” entry should be BLACK to signify that it references a location that is valid. If the entry is RED it needs referencing to the folder containing the SAM Objects; SAM\Objects\classes. This is achieved by:
   a. Selecting the SAMObjects entry.
   b. Pressing the “Edit…” button on the right.
   c. Selecting the “Add” button then “Add Path”.
   d. Selecting the “SAM\Objects\classes” folder.
   e. Selecting and deleting the old “RED” path.
   f. Clicking “OK” to get back to the Project Settings screen (figure 1.9).

   Now, making sure that the SAMObjects checkbox is ticked and the text is BLACK, the Project Settings “OK” button can be pressed to exit this window.

4. The project can now be re-compiled by pressing F7 to “Build Project”.

SAM Data Acquisition Module Setup (Command Prompt)

5. This step is almost identical to step 20 of the Data Storage module setup, only using a the file “startDataAcquisition.bat”. Change the reference to the jdk “bin” folder in the same way as in this step.

Appendices

L

SAM User Manual

SAM User Manual

Introduction

SAM (Sports Arbitrage and More) is a collection of programs that identifies ways in which a user can make profit by betting on all outcomes of a sporting event with multiple bookmakers. A Data Acquisition module retrieves event and bet information from the internet using bookmaker specific miners and sends them to the Data Storage module where the data is analysed, grouped together and evaluated. This data can then be viewed by a user, along with stake percentage advice, and taken advantage of if it is so wished.

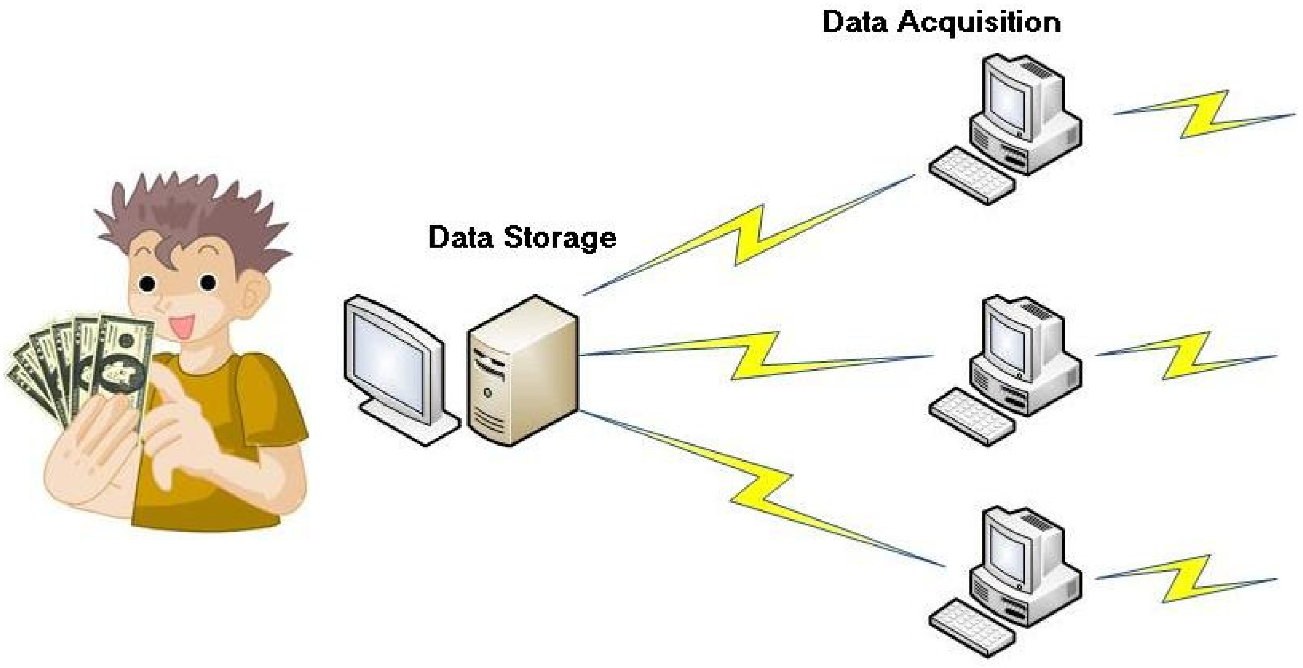


Figure 1.1 – SAM Overview

This user manual covers the use of all of SAM (Sports Arbitrage and More) components. Installation must have taken place in accordance with the SAM Installation Manual in order for this user manual to have full relevance.

The "user" of this software is assumed to be of a programmer class as the software is designed to be expanded. There are separate manuals designed to assist in the creation of new opportunity definitions (Appendix M) and new data miners (Appendix N) so this User Manual is aimed simply at running the software and explaining the functions accessible to the user whilst it is running.

Storage Module Guide

The Data Storage module of SAM comes pre-configured in every aspect of its operation. There are no configuration parameters that can be set by a user for the general use of this software. Once running, this module is completely autonomous. The only interactions with the user come when there are infrequent error messages that are produced for a variety of reasons.

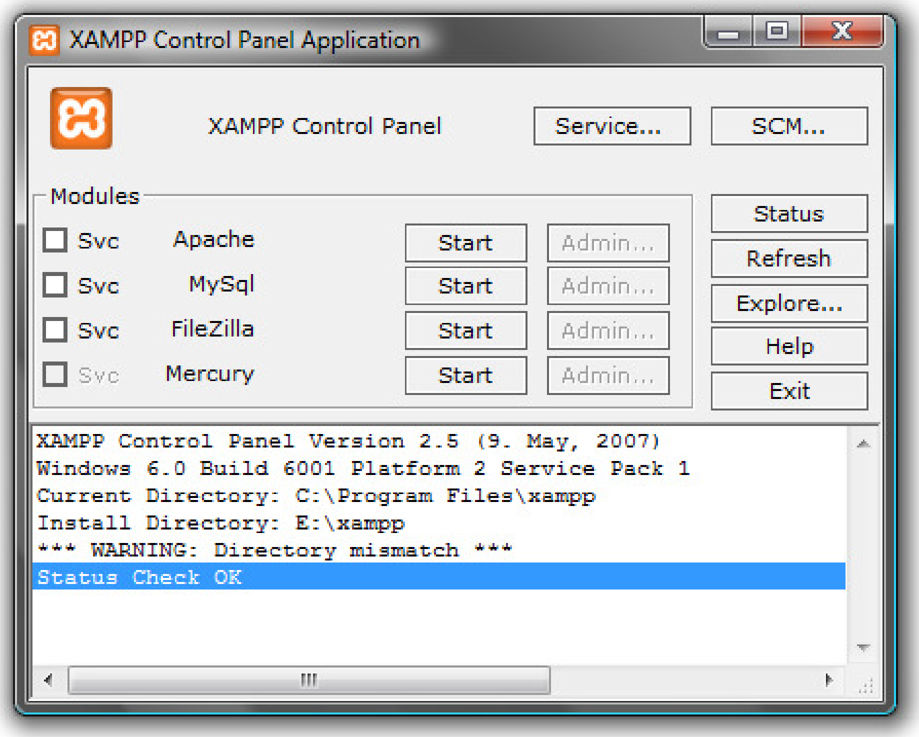
A late addition to this module was the inclusion of the facility to input new participants to the database. The use of this function will be fully explained.

Getting Started

Start XAMPP

XAMPP supplies the mySQL database that the Data Storage module uses. To start it:

1. Start XAMPP using the start menu or desktop icon (left of figure 1.1). This will show the XAMPP Control Panel (right of figure 1.1).



```
XAMPP Control Panel Version 2.5 (9. May, 2007)
Windows 6.0 Build 6001 Platform 2 Service Pack 1
Current Directory: C:\Program Files\xampp
Install Directory: E:\xampp
*** WARNING: Directory mismatch ***
Status Check OK
```

Figure 1.2 – XAMPP Control Panel.

2. Press the “Start” buttons for MySQL and for Apache followed by the “Admin...” button for Apache. The “Start” button text changes to “Stop” and “Running” messages appear to the left of them.

Start RMI Registry

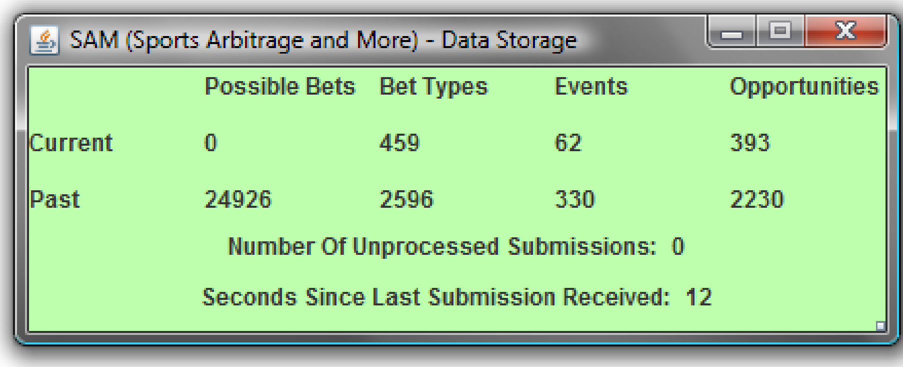
The RMI registry allows the Data Acquisition modules to connect to the Data Storage module.

3. Double click on the batch file named “startRMIReg.bat” in the SAM folder (figure 1.2). This will open a command prompt window (DOS prompt style) that should show no errors and no prompt after the “rmiregistry” command. This is now running and able to direct incoming connections to the (soon to be) running Data Storage module.

Start the Java Data Storage Module (JCreator)

This is an alternative to the command prompt version of this step below, so they should not both be carried out.

4. Open up the Data Storage module project file in the supplied JCreator workspace by opening SAM\DataStorage\DataStorage.jcw in JCreator. Run the project by pressing the F5 key and the GUI should pop up on the screen (figure 1.3). JCreator can now be minimized.


SAM (Sports Arbitrage and More) - Data Storage

	Possible Bets	Bet Types	Events	Opportunities
Current	0	459	62	393
Past	24926	2596	330	2230

Number Of Unprocessed Submissions: 0

Seconds Since Last Submission Received: 12

Figure 1.3 – Data Storage GUI.

Start the Java Data Storage Module (Command Prompt)

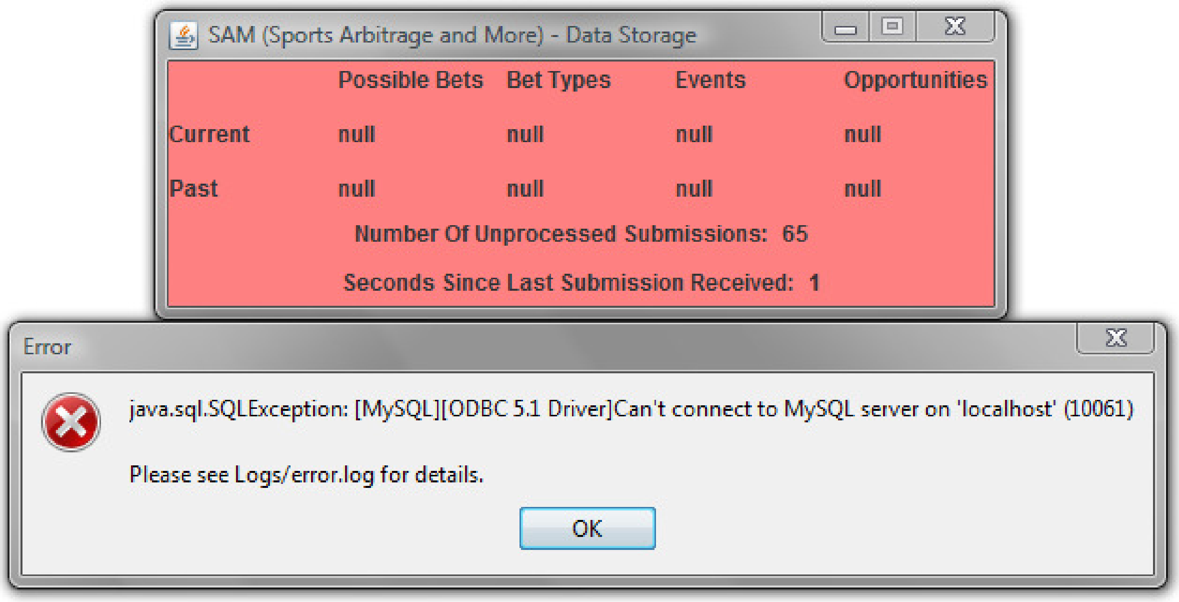
This is an alternative to the JCreator version of this step above, so they should not both be carried out.

4. Simply double click/run the batch file in the SAM folder entitled “startDataStorage.bat”.

General Use

Once the Data Storage module is running it sits on the desktop causing little fuss. The statistics visible in figure 1.3 are updated to show the current state of the database, as are the two values below. Unprocessed submissions refers to the size of the buffer accepting data that is submitted. This will increase at busy times and reduce back to zero at quieter times. The seconds since last submission value is reset to zero every time a submission is received.

If any errors occur, like loss of connection to the database, the display turns red and a dialogue box is displayed to notify the user, giving them as much information as possible (figure 1.4). The same information is written to a log file. If the user is away from the system and the connection is re-established, the dialogue box remains to inform the user that the error happened, but the GUI returns to its green, working condition and the processing of submissions is allowed to continue.


SAM (Sports Arbitrage and More) - Data Storage

	Possible Bets	Bet Types	Events	Opportunities
Current	null	null	null	null
Past	null	null	null	null

Number Of Unprocessed Submissions: 65

Seconds Since Last Submission Received: 1

Error

```
java.sql.SQLException: [MySQL][ODBC 5.1 Driver]Can't connect to MySQL server on 'localhost' (10061)

Please see Logs/error.log for details.
```

Figure 1.4 – Data Storage Database Connection Error.

Maintenance

Database Size Management

As data in the past data section of the database is never deleted, without management this will grow large and affect the efficiency of the database. When the past data statistics displayed on the GUI reach high levels the user must manually delete data from the database (this is the way the customer specified data management). This manual interference can be dangerous as it can cause inconsistencies in data and errors to occur if care isn't taken.

To do this safely the user must ensure that the Data Storage module is NOT running and that the database is not being used by any other source before manipulating data directly. The user must access the phpMyAdmin interface supplied by the XAMPP software (details of how to do this are in the Installation Manual).

It is advised that the best way to ensure post maintenance consistency is to empty ALL of the database tables EXCEPT the following:

1. alias_list
2. bet_cautions
3. bet_styles
4. bookies
5. bookie_cautions
6. bookie_caution_list
7. opportunity_types
8. sport_types

Assigning An Alias To A Participant

This is achieved using direct database manipulation as above, using the phpMyAdmin interface. Again, it is advised that no other database access is taking place.

1. Find the Participant_ID of the participant that the alias is to be assigned to by browsing the participants table of the database.
2. Locate the alias in the alias_list table of the database that is to be assigned.
3. Edit the record of the alias and update the Participant_ID field to the required value.

Insert A New Participant

This could be very dangerous if implemented using the same techniques as above, as alias strings are processed by the Data Storage module before they are entered into the database. Direct insertion may allow values that could cause serious consistency issues for SAM. To this end, an interface has been bolted onto the Data Storage module GUI to do just this job.

With the Data Storage module running, click the tiny button in the bottom right hand corner of the GUI (figure 1.5). The result is the appearance of another window specifically for participant insertion (figure 1.6).

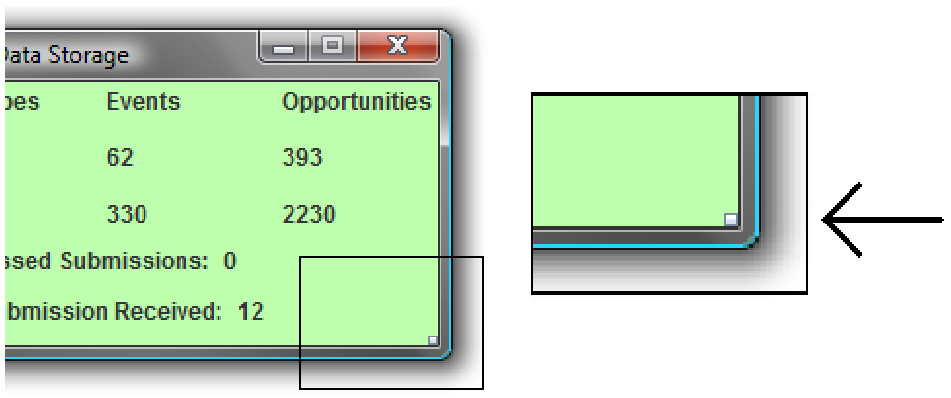


Figure 1.5 – Participant Insertion Button

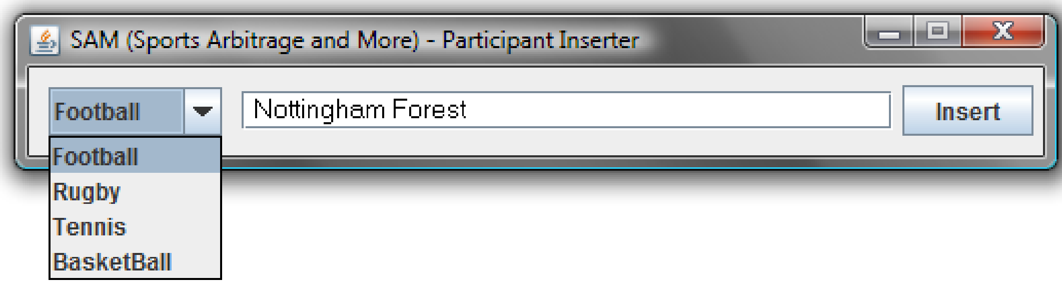


Figure 1.6 - Participant Insertion Window

From here insertion of new participants is possible. The user must select a sport type and insert the primary name for the participant then press the “Insert” button (or press the “Return” key). Duplicate primary names are not accepted. If the insertion of a name would result in the use of an alias that is already in the database, the user is prompted for confirmation to take the alias from its current participant. Following insertion the new participant identification number is displayed to the user.

Data Acquisition Module Guide

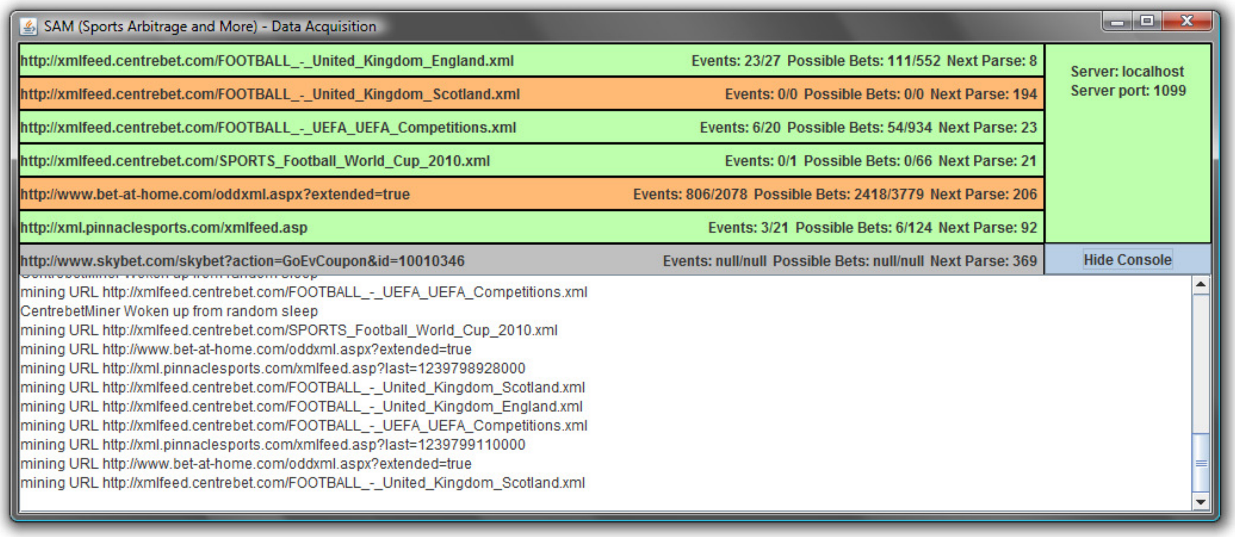
The Data Acquisition module has no interaction once running but has an XML configuration file (SAM\DataAcquisition\classes\config\SAMconfig.xml) that defines its operation and performance. It can be viewed in any simple text viewer or using any popular web browser. The file (also sown in Appendix J) is also annotated to describe the function of each line of the code, so this should be studied prior to running the Data Acquisition module.

Getting Started

Start the Java Data Acquisition Module (JCreator)

This is an alternative to the command prompt version of this step below, so they should not both be carried out.

1. Open up the Data Acquisition module project file in the supplied JCreator workspace by opening SAM\DataAcquisition\DataAcquisition.jcw in JCreator. Run the project by pressing the F5 key and the GUI should pop up on the screen (figure 1.7). JCreator can now be minimized.



SAM (Sports Arbitrage and More) - Data Acquisition

```
http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_England.xml        Events: 23/27  Possible Bets: 111/552  Next Parse: 8
http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_Scotland.xml       Events: 0/0  Possible Bets: 0/0  Next Parse: 194
http://xmlfeed.centrebet.com/FOOTBALL_-_UEFA_UEFA_Competitions.xml        Events: 6/20  Possible Bets: 54/934  Next Parse: 23
http://xmlfeed.centrebet.com/SPORTS_Football_World_Cup_2010.xml           Events: 0/1  Possible Bets: 0/66  Next Parse: 21
http://www.bet-at-home.com/oddxml.aspx?extended=true                      Events: 806/2078  Possible Bets: 2418/3779  Next Parse: 206
http://xml.pinnaclesports.com/xmlfeed.asp                                 Events: 3/21  Possible Bets: 6/124  Next Parse: 92
http://www.skybet.com/skybet?action=GoEvCoupon&id=10010346                Events: null/null  Possible Bets: null/null  Next Parse: 369
```

```
Server: localhost
Server port: 1099
```

```
mining URL http://xmlfeed.centrebet.com/FOOTBALL_-_UEFA_UEFA_Competitions.xml
CentrebetMiner Woken up from random sleep
mining URL http://xmlfeed.centrebet.com/SPORTS_Football_World_Cup_2010.xml
mining URL http://www.bet-at-home.com/oddxml.aspx?extended=true
mining URL http://xml.pinnaclesports.com/xmlfeed.asp?last=1239798928000
mining URL http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_Scotland.xml
mining URL http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_England.xml
mining URL http://xmlfeed.centrebet.com/FOOTBALL_-_UEFA_UEFA_Competitions.xml
mining URL http://xml.pinnaclesports.com/xmlfeed.asp?last=1239799110000
mining URL http://www.bet-at-home.com/oddxml.aspx?extended=true
mining URL http://xmlfeed.centrebet.com/FOOTBALL_-_United_Kingdom_Scotland.xml
```

Figure 1.7 – Data Acquisition GUI.

Start the Java Data Acquisition Module (Command Prompt)

This is an alternative to the JCreator version of this step above, so they should not both be carried out.

4. Simply double click/run the batch file in the SAM folder entitled “startDataAcquisition.bat”.

General Use

The Data Acquisition GUI is composed of miner and submitter visualisations and a console for outputting a running commentary of progress. The console can be hidden at the touch of a button to reduce the GUI down to a simple interface that displays the status all of the module’s components in a quick and easy to appreciate way. The console displays mining, submitter and error information so that a user can see the full history of what has happened since startup.

There are separate miner visualizations for all of the miners specified in the configuration file and a single one for the submitter. They are designed so that a single glance reassures the user that everything is working fine by colour coding the components in a traffic light style with green for “good”, red for “bad”, orange for “operating” (the submitter does not turn orange) and grey for “unknown”.

Each miner visualization displays statistics from the last/current mining operation so that a judgement about its efficiency can be made. This can show when a miner is not functioning correctly or the format of the mined data has changed so the miner is no longer working properly even though there are no errors generated.

Viewing data

Unfortunately there is no custom viewing interface for the data at present, so the phpMyAdmin interface is used to fulfil this function. Tables can be viewed very easily using this software and sql queries can be created and ran (with the use of search tools etc.) to get any view that the user wants. Appendix F of the report that this manual is a part of contains numerous useful queries for this purpose, allowing a user to see event, opportunity or betting data in a number of different ways.

Appendices

M

SAM Programmer Manual – Opportunity Definitions

SAM Programmer Manual - Opportunity Definitions

This document describes how to add a new Opportunity Definition to the Data Storage module of the SAM system.

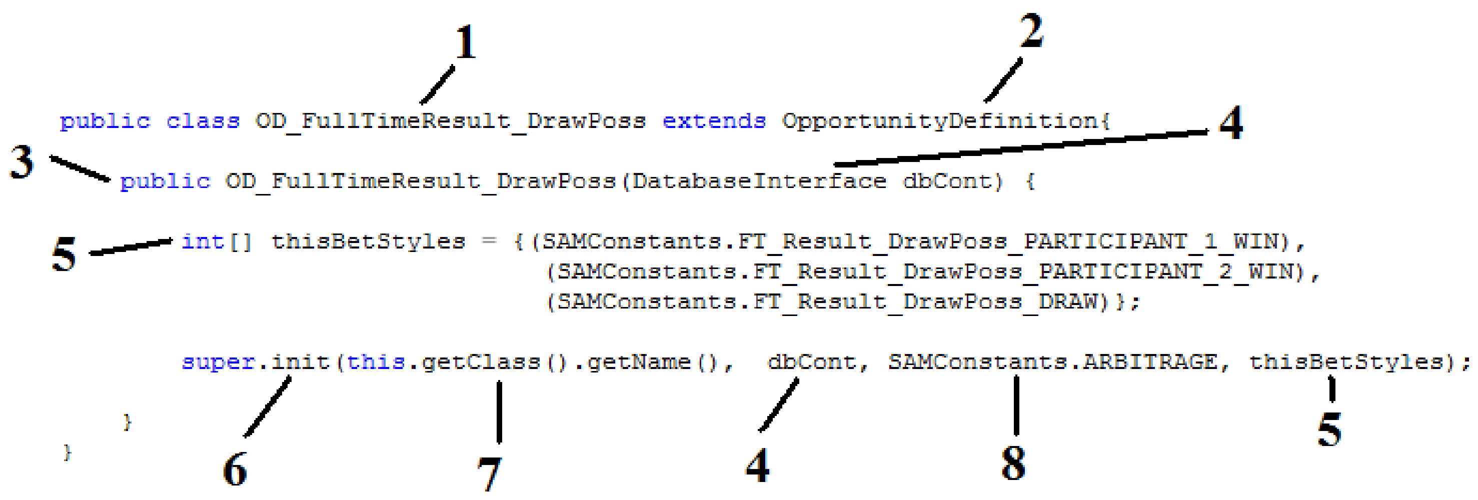
Creating a new opportunity description can be very easy, depending on the opportunity characteristics. All that is required is to write an object class that inherits from the OpportunityDefinition class. Then, every time a new bet type is registered in the database the system will check to see if the new bet type can be used to create an opportunity of this kind. There is no need to register the class anywhere, just make sure that when it is compiled the .class file resides in the working directory that all other class files are in.

The only way to tell which opportunity definitions have been found is by checking the standard output on the IDE when the Data Storage module is run. Once it has been confirmed that the program has recognised a new definition once, it can be assumed that it will keep on doing it.

Writing an Opportunity Definition Object Class

The OpportuityDefinition class is very comprehensive and most of its methods are reusable by most new definitions. It is this class that has the public methods that will be called and the subclass (in almost all cases) will merely specialise it by providing values for it operation.

Below is an example of a simple opportunity definition defining the arbitrage created by the odds for a home winning, an away win and a draw in an event.



```
public class OD_FullTimeResult_DrawPoss extends OpportunityDefinition{
    public OD_FullTimeResult_DrawPoss(DatabaseInterface dbCont) {
        int[] thisBetStyles = {(SAMConstants.FT_Result_DrawPoss_PARTICIPANT_1_WIN),
                               (SAMConstants.FT_Result_DrawPoss_PARTICIPANT_2_WIN),
                               (SAMConstants.FT_Result_DrawPoss_DRAW)};

        super.init(this.getClass().getName(),  dbCont, SAMConstants.ARBITRAGE, thisBetStyles);
    }
}
```

1.1 A simple Opportunity

1. The name of the class starts with “OD_” and it describes the opportunity succinctly. The opportunity will only be recognised if its name starts with these characters.
2. All new opportunity classes must extend OpportunityDefinition.
3. There is only a constructor here, no methods, as will be explained below.
4. All constructors must take a DBInterface as argument. This is later passed to the superclass.

5. An array of betStyles that make up this arbitrage. The bet styles are defined in the SAMConstants class.
6. A call to the superclass with the specific data and database interface for this opportunity.
7. A description of this opportunity (the class name is used for consistency).
8. A description of the type of opportunity (arbitrage, middle, 20% middle). This defined in the SAMConstants class.

The OpportunityDefinition class has provided two protected methods for this subclass that could have been overriden if the opportunity was more complex. These methods are described below.

```
protected Vector<BetSaM> getOtherSaMs(BetSaM betSaM)
```

This method returns a vector of BetSaMs (bet styles and mutators) that the caller would also need in the database in order to create one of these opportunity objects. It does not need to be overriden in this case as there are no mutators used with the bet styles. It only requires overwriting if mutators are used in this opportunity.

```
protected void calculateOpportunity(int oppID, Vector<Float> bestOdds) throws Exception
```

This method is called after the opportunity has been inserted into the database. It calculates the return percentage of this opportunity as well as the stake percentages that a user would need in order to take advantage of this opportunity. The default implementation of the method assumes that all of the bet types that make up the opportunity have an equal weighting, which is true for a classic arbitrage. This may not be the case for complicated middles (say), if the opportunity was made out of odds (say) for:

- The home team scoring 1 goal or more.
- The away team scoring 1 goal or more.
- No goals being scored in the match.

Here you may expect the no goals possible bet to have less of a chance of occurring, so may want to assign it less "weight". If this were the case, this method would need to be overriden. The following information will be useful in this case.

oppID -	is the Opp_ID of the opportunity (int).
bestOdds -	is a Vector of Float objects representing the best odds of the bet types that make up the opportunity (ordered by the bet types' betStyleID).

The new method would need to end with the following call:

```
updateDB(oppID, oppPercentage, stakePercentage);
```

Where:	oppID -	is the Opp_ID of the opportunity (int).
	OppPercentage -	is the calculated return percentage of this opportunity (float).
	StakePercentage -	is an array of stake percentages associated with the bet types in this opportunity, ordered by their betStyleID (float).

Appendices

N

SAM Programmer Manual – Data Miners

SAM Programmer Manual - Data Miners

This document describes how to add a new Data Miner to the Data Acquisition module of the SAM system. It is very difficult to write a complete guide as to how to create new miners as so much depends on the format or style of the data that is being parsed. With the information in this document, the source code of the already implemented miners and the associated code documentation, a skilled programmer should be able to write miners for their chosen bookmaker's data however it is laid out.

Overview

There are two kinds of miner that can be written; one for XML data that must inherit from the XMLMiner class and one for HTML data that must inherit from the HTMLMiner class. Both of these superclasses inherit from a superclass called SAMMiner, which itself inherits from SAMMinerExtras. These two classes (SAMMiner and SAMMinerExtras) can be considered to be the same class for functional purposes, with the only reason for their separation being that together their size was unmanageable.

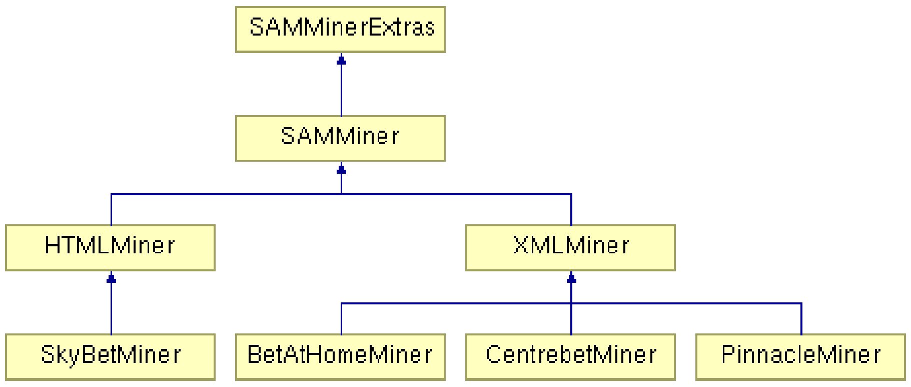


Figure 1.1 – Miner Inheritance Hierarchy.

Useful Objects/Methods

There are many superclass methods defined which are used by the current miners. It is essential that these current miners are studied, and the superclass documentation read, in order get the full use out of inherited code and the supporting objects.

getSOB();

This method retrieves the SubObjBuilder (submission object builder) from the superclass.

SubObjBuilder

This object class (a submission object builder) allows a miner to slowly build the necessary submission objects by giving it event and bet data as and when the miner finds it. This reduces the requirement for lots of local variables. There is one created in a superclass and a reference to it can be is returned by calling the above method.

There is a Boolean (debug) flag in this class that allows the output (to the IDE console) of all of the accumulated data prior to the attempt to create a submission object. This can be used to de-bug a new miner as output (to a submitter log file for example) is only possible if the SubmissionObject has been correctly built. If it hasn't then the SubmissionObject is thrown away and it will not be output.

reorderMarketBetStyles(int market, int pos1, int pos2);

Often a market is defined then the odds are given without a description of what the individual odds are for. The superclasses therefore allow the selection of a market that puts the expected bet types (and mutators) into a buffer(s) that can then be accessed as the odds are given.

If the odds for a market come in a different order to the "standard" order described in SAMMinerExtras (MARKET_BET_STYLES), a re-ordering will be required. Match Results for instance are expected in the order home win, draw, away win. If this is incorrect then a call to this superclass method will be required, passing in the required arguments with reference to the MARKET_BET_STYLES constants in SAMMinerExtras.

HTMLPageProcessor

This is a class that is designed to be used by an html miner object, and does a lot of the hard work for it too. It takes as arguments:

1. A url String (where to fetch the html from).
2. An array of regular expressions to locate the important bits of data.
3. An array of integers that tell it how much to chop off the FRONT of a string returned by a regular expression. (Called startChaff in SkyBetMiner.)
4. An array of integers that tell it how much to chop off the BACK of a string returned by a regular expression. (Called endChaff in SkyBetMiner.)
5. A java.awt.event.ActionListener (the HTML miners implement this interface) that will be notified when some useful data is found.

setNextURLAddition(String string)

This method allows a miner to change the URL that will be mined the next time that the miner mines.

Please see the associated code documentation for more information.

Constructors

For both types of miner the constructor that will be written can (currently) only take arguments if:

- The url string assigned to this miner is sport specific. In this case there may be no indication on the page what sport the data refers to so it must be defined prior to parsing.
- The miner only mines the "full" version of the odds every "x" times it mines, and this "x" is the argument passed in. This is demonstrated in the PinnacleMiner class.

The first thing that the new miner's constructor must do is call the superclass constructor with the three arguments (first two only for HTML) listed below, which are usually set manually as constants in the new object class' local variables.

bookieID -	An int representing the bookieID that matches the one in the database "bookies" table.
marketStrings -	A String array containing the strings used in the XML data to represent the required markets. The currently defined required markets (in SAMMinerExtras - MARKET_BET_STYLES) are Match Result, Handicap Result and Double Chance.
tags -	A String array of the "used" tags in the xml file; the ones that hold important information and that require further action when one is found.

Finally, consideration as to whether a call to reorderMarketBetStyles(int market, int pos1, int pos2) is required.

Creating An XML Miner

This is by far the easiest of the two miners to create as the structure of the parsed code is fairly predictable. Although this is true every bookmakers' XML pages are different, so there has been flexibility included into the inherited code. XML miners inherit from the XMLMiner class and the new object is controlled from the superclass by it calling the overriden methods.

Overriden Methods

There are three methods that need to be overriden in order to get an XML miner to compile:

protected void startTag(int tagID, Attributes atts)

> This method will be called by a superclass when the parser finds an XML element's start tag that is "used" (the name was in the array "tags" in the constructor).
>
> The tagID is equal to the array position that the string had in the "tags" array.
>
> The atts argument is an object that implements the org.xml.sax.Attributes interface and represents attributes of the tag. This can be used if the useful data in the XML is stored in the attributes of the start tag (e.g. <info data1="blue"></info>). Often this will not be the case and atts will not be needed as the useful data will be between the start and end tags of an element (e.g. <info>blue</info>).

protected void endTag(int tagID)

> This method is similar to startTag() but it is called when that parser finds a "used" end tag.
>
> Again the tagID argument is the position that the string had in the "tags" array.

protected void characters(Short tagID, String string)

> This method is called when text is found in between "used" start and end tags (e.g. <info>blue</info>).
>
> Again the tagID argument is the position that the string had in the "tags" array.

The string argument is the text that was found between the tags. The SAXParser used to parse XML here has been known to return a fragment of the string between the start and end tags here if certain characters are used in the XML. In this case this method will be called again immediately after the first call with the rest of the characters in the string argument and it is suggested that in this case the first string is buffered and the second string is added to it and the whole set of characters is used.

XML Approach

When any of the above methods are called the current miners all have a switch statement that decides how to handle the data depending on the tagID variable (position in the tag names array that the superclass was called with). Actions can then be carried out with the data that the method is given, or actions can be carried out due knowing that the parser has reached a certain part of the XML (like ending an event in the SubObjBuilder when the endTag method is called with an argument that corresponds to an "Event" tag). See the Submission Object Building section below for more details.

Creating An HTML Miner

As the structure of bookmaker website HTML is extremely variable, this type of miner is quite different from its XML sibling. The control of this object is in the new subclass, so it is has far more freedom to work with the specific html that it is written for. Please see the SkyBetMiner class for an example of an HTML miner.

Overriden Method

There are two methods that require overriding in order to get a new HTML miner to compile:

protected void parse() throws Exception

This method hands control of the mining thread to the new miner object, which is then free to mine however the implementer sees fit. Typically it will create an HTMLPageProcessor with object variables and call its processPage method.

public void actionPerformed(ActionEvent ae)

This method is invoked when an action occurs in objects that this object is registered with.

The idea of HTML mining in SAM is that an HTMLPageProcessor, once set up with page specific data, creates events containing the important information in the HTML document. The new miner class implements the java.awt.event.ActionListener interface (via a superclass) and is notified of any important information by way of an java.awt.event.ActionEvent that contains:

1. A reference to the HTMLPageProcessor that fired the event as the "Source". (Call getSource() on the actionEvent.)
2. A reference to the position in the array (used to construct the HTMLPageProcessor) of the regular expression that was matched. (Call getID() on the actionEvent.)
3. The string that was returned from the matched regular expression once the specified number of characters had been chopped off the front and back (specified again in the HTMLPageProcessor). (Call getActionCommand() on the actionEvent.)

HTML Approach

The general approach of HTML miners is to be notified when "interesting information" is parsed by an HTMLParser in a similar way to the XML miner, but they have freedom how they do this as they implement it for themselves. They do this by implementing the actionPerformed method above then calling methods that they have created to handle the data included in the ActionEvent in the same way that the characters method does in XML mining. This method can have a switch statement that switches on the ID field of the ActionEvent (which is the array position of the regular expression that has been satisfied), so that it is easy to see which code is written to handle which data.

Submission Object Building

With the "interesting information" the miners build a SubmissionObject gradually using a the submit methods on a SubObjBuilder. The SubObjBuilder needs to be explicitly told that the miner has come to the end of a bet or event (by calling endNewBet() or endNewEvent() respectively) at which point it tries to create the relevant submittable object. At any time a miner can call its own submit() method that will take any SubmissionObjects from the SubObjBuilder and send them to the submission manager.

Updating statistics

Both types of miners need to update statistics. The successfully mined data stats are provided by the SubObjBuilder, it is the overall events and bet lines that need to be kept track of. Two superclass method calls increment the events found (incrementEvents()) and the possible bets found (incrementPossibleBets()) and should be called every time a new event and possible bet are located. These statistics are used to notify the user of the miner's success.

Other things to consider when writing miners

Is the same URL mined every mining operation?

> It is possible to set the URL that is mined the next time that the miner is told to mine by calling the setNextURL method in the SAMMiner superclass. If this function is used a miner can be told to return to the originally mined URL every X mining operations using the content of the FullRefreshOnPeriod tag in the configuration file. An example of this is in the PinnacleMiner class.

Is the odds source in the same time zone and is daylight saving going to affect the mined times?

> When submitting event start times to the SubObjBuilder class there is an argument for the time difference. This should be carefully considered whilst writing a miner. A good example of a miner that is sensitive to both of these issues is PinnacleMiner. This has a manual time difference variable for the different time zone but the source does not change times for daylight saving. This manually inserting any daylight saving time.