AUTOMATIC DISCOVERY OF SIGNIFICANT EVENTS FROM DATABASES

By

AVINASH SHANKAR BHARADWAJ

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

DECEMBER 2011

ACKNOWLEDGEMENTS

It is always a pleasure to thank those people without whom it would have been impossible to complete my thesis. First and foremost, I would like to express my deepest gratitude to my thesis supervisor Dr. Chengkai Li for supporting me and guiding me throughout my research work. Without his constant help and advice it wouldn't be possible for me to finish my thesis.

I would also like to thank my committee members Dr. Fegaras and Dr. Elmasri for being on my panel and giving me advice related to my thesis work.

I cannot end without thanking my parents, on whose constant support and encouragement I have relied on throughout my time at the University of Texas at Arlington and also the constructive advice received for the completion of my thesis.

I would like to greatly appreciate the help of my friends here in Arlington who has always been providing me their helping hands whenever I need it.

August 7, 2011

ABSTRACT

AUTOMATIC DISCOVERY OF SIGNIFICANT EVENTS FROM DATABASES

Avinash Shankar Bharadwaj, M.S.

The University of Texas at Arlington, 2011

Supervising Professor: Dr. Chengkai Li

The advent of the internet has caused enormous amounts of data available online causing many significant facts to be hidden within this data. Searching for a significant fact within these large datasets is a query intensive process involving large amounts of queries which needs to be executed hence slowing the process of finding the significant facts from a large dataset. In this thesis, a novel approach has been designed exploiting the statistical characteristics of the data present in the dataset to reduce the number of queries on the dataset. A two phased approach is considered for making fact finding more efficient. The approach consists of design and implementation of the prediction and the decision making algorithm. The prediction algorithm predicts the time frame for a significant event to happen and the decision algorithm uses the results from the prediction algorithm to decide whether to check for a significant event or not. We compare our results obtained after the implementation of the designed algorithms and found that queries are executed lesser number of times compared to the other existing solutions to this problem.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

Table Page

CHAPTER 1

INTRODUCTION

1.1 Motivation

Significant fact finding is the process of finding important facts from a large dataset. If we consider the sporting league NBA, then the significant facts are answers to questions such as (1) Who are the top-100 scorers'? (2) Who have made the most number of assists? (3) Who are top hundred three pointer scorers?

It is the job of sports journalists all over the world to follow different sports and provide information to the fans of the sports whenever a particular player achieves a milestone. Consider basketball; people all over the world want to know when their favorite players achieve milestones. A player entering the list of top-100 scorers' or entering the list of top-100 three pointer scores' are some examples of the milestones. If a player becomes one of the top-100 scorers' in NBA history, then the story is worth reporting. There are many such significant events mentioned in real world news articles. Below is a list of such excerpts from several different news articles. Each excerpt shows how a significant event can be reported as news.

- "Lebron James became only the second ever player to record a triple double in the All Star game, Michael Jordan being the only previous player to manage that feat."
- "Bryant had put up 21 points by half time. He was halfway to Wilt Chamberlain's record of 42 from 1962. Intriguingly, Wilt's 42 were 9 points below his average for that season which tells you the kind of year that was."
- "Boston Celtics guard Ray Allen matched and eclipsed Reggie Miller's NBA record of 2,560 career 3-pointers Thursday night against the Los Angeles Lakers."

If a player achieves a milestone, then the achievement is reported by a journalist. However, the entire process can be automated using data mining technologies.

1.2 Challenges in finding significant events

This section discusses the various problems associated with solving the problem of significant fact finding.

Views are extensively used to find significant facts. If the significant event is a player entering the list of top-100 scorers' in NBA history, then the view will contain the top-100 players' by total career points. If a player scores 20 points in a particular game, then total score of the player changes. Thus, the list of top-100 scorers' may need to be updated. The straightforward method to update the view is to recalculate the view whenever there is an update on the data present in the database. The view can be recalculated by using a trigger. This method can be named as the absolute method.

Consider an example where there are 250,000 records in the database. Each record contains various parameters recorded for a player per game. After each game is played, new records representing the performance of players in that game would be inserted into the database. Following the absolute method, the view must be recalculated after each new record is inserted. If there are 200,000 records that have to be inserted into the database, then the view needs to be re-calculated 200,000 times. This makes the usage of the absolute method for maintaining views query-intensive and inefficient.

1.3 Designed Approach

To overcome the drawback of the absolute method, a new approach is designed where the views are updated selectively.

The designed approach predicts when the views have to be recalculated in order to find a significant event. Certain statistical parameters present in the data are used to predict when the views have to be recalculated. Consider that Kobe Bryant is currently not in the list of top-100 scorers. From the existing data, we can predict the number of games needed for Kobe Bryant to enter top-100

scorers list. If Kobe Bryant needs ten games to become one of the top-100 scorers, then the view is recalculated only after Kobe Bryant plays ten games.

To achieve the aforementioned goal, a two-phased approach is taken. The two phases are the prediction phase and the decision phase, respectively.

The prediction phase uses the statistics of certain attributes like points, total assist and total three pointers scored to predict when a particular view has to be updated. Along with the statistical data, the concept of prediction interval is also used. The prediction interval determines the range of values in which future observations will fall with a certain probability, given existing observations. Once the prediction interval is obtained, the minimum number of points that a player has to score to reach a milestone is calculated. Using the number of points that a player has to score, and the minima of the prediction interval, the number of matches can be calculated by dividing the minimum number of matches required with the minima of the prediction interval.

Consider that Kobe Bryant has played 25 games previously. The average and the standard deviation of the points scored by Kobe Bryant in the 25 games are calculated. Using the average points and the standard deviation, the prediction interval can be calculated for Kobe Bryant with a certain probability. If the prediction interval for Kobe Bryant is 15 to 25 points, then it means Kobe Bryant will score between 15 to 25 points in the next match with a certain probability. After calculating the prediction interval, the number of points needed for Kobe Bryant to reach top-100 scorers is calculated. If Kobe Bryant has a score of 6000 points, and the player at the 100^{th} position in the top-100 list has a score of 6149, then Kobe Bryant needs to score at least 150 points to get to the top-100 scorers list. If Kobe Bryant scores 25 points in every match, then he needs at least 6 more matches to become one of the top-100 scorers.

The decision algorithm maintains a counter for number of games a particular player has played. By comparing the counter against the results of the prediction algorithm, the decision algorithm determines whether to recalculate the view or not. If the number of games played by the player is equal to the number of games required to reach a milestone, then the view is recalculated.

In the previous example it is mentioned that Kobe Bryant would need six games to reach the milestone of top-100 players. So whenever Kobe Bryant plays a game, the decision algorithm increments the counter associated with the player by one. If Kobe Bryant plays six games, then the counter is incremented six times. Once the counter is incremented six times the prediction value and the counter are equal. Hence, the view is recalculated.

Unlike the absolute method this method is designed to recalculate the view selectively. Hence, it reduces the number of times the view is recalculated.

1.4 Overview of experiments

A set of experiments need to be conducted to verify the approach. We performed different experiments to compare our results against that of the absolute method. In the experiments, certain parameters are varied to measure the difference in performance.

The standard score determines the accuracy of the prediction algorithm. The accuracy value for the standard score of 2.58 is 99%. We also compare the difference between the game at which the designed algorithm captured the significant event, and the game at which the significant event was actually generated. If an event was generated at say game 342, then the event was detected only at game 345 then the difference between the numbers of games is 3.

From the results obtained by the above mentioned experiments, it can be verified that the approach designed recalculates the view about 10% of the time. This is in comparison with the absolute approach. The average difference between the games when the significant event is captured, and the significant event is actually generated is 3, if the accuracy of the prediction algorithm 99%.

Thus, this method for finding significant events can prove to be an efficient approach when compared to the absolute method which recalculates the view whenever there is an update on the database and without a large difference between when the significant event was actually generated and is actually detected.

CHAPTER 2

RELATED WORK

2.1 View Maintenance Problem

A view in a database contains the results of a query on a database table. Thus, any change to the data in the table will cause the view to change.

The view update problem can be stated as follows. Consider a database 'T', a view definition 'A', and an update operation 'U'. The view can be defined as A(T). When there is the operation 'U' on T, the changes in 'T' have to be propagated to the view A(T). The update operation 'U' can be an insert, delete, or an update operation.

Ashish Gupta et al. [14] describe the counting algorithm and the Delete and Re-derive algorithm (DRed) [14] for efficiently maintaining the views. However, these algorithms can be used only when there is a delete operation on the data.

The counting algorithm counts the number of multiple derivations for a tuple in the view. Whenever a record in the database is deleted, the associated count value of the tuple is decremented. If the count value for a particular tuple becomes zero, then the tuple is deleted.

In the DRed algorithm if a record in the master data is deleted, then all the tuples in the view related to that record are deleted. Once the tuples in the view are deleted, alternative derivations for each deleted tuple are found. If the tuple can be re-derived, then it is reinserted into the view.

Jose. A. Blakely et al. [13] describe a differential update algorithm. The differential update algorithm detects whether the insert operation causes the view to change or not. The algorithm works by comparing the each insert operation against a set of rules, which determine whether the operation, causes the view to change or not.

Eric N. Hanson et al. [15] describe two strategies, the immediate update strategy and the deferred view update strategy. In the immediate update strategy, the view is updated as soon as there is an update to the database. However, in the deferred view update the view is updated incrementally just before the data is retrieved from the view. Eric N. Hanson et al. [15] has found that the deferred view update strategy has a lesser cost, than the immediate update strategy.

All the above mentioned strategies recalculate the view entirely, when there is an update. However in some conditions, the contents of the materialized view can be directly updated without accessing the base data. This strategy can be explained using the following example.

Consider that a player within the top-100 scorers' plays a match. The top-100 scorers' view can be updated by adding the player's new score, to the total score of the player. This update can be performed without accessing the database table. According to F. W. Tompa et al. [17] views where the contents can be updated without accessing the base data are called as "conditionally autonomously computable" [17] views.

The Oracle database system provides various update strategies, which allow the materialized views to be maintained accurately. Some of the major update strategies provided in Oracle are as follows.

- Unconditional refresh
- Fast refresh
- Selective manual refresh

2.1.1 Unconditional refresh [12]

In the unconditional refresh, the contents of the view are unconditionally refreshed whenever the data changes. An unconditional refresh does a "complete refresh" of the materialized view. A complete refresh recalculates the materialized view using the view's defining query. To recalculate the view, the contents of the database have to be read. If the size of the data present in the table is large, then the time taken to recalculate the view would be large.

The method of unconditional refresh is used when the materialized view does not satisfy the necessary conditions for either a fast refresh, or a selective manual refresh.

2.1.2 Fast Refresh [12]

The concept of fast refresh is based on the theory of "conditionally autonomously computable" [17] views. In the fast refresh approach, the changes can be directly added to the existing data in the view. Thus, the fast refresh approach results in a fast refresh time. The time taken to perform a fast refresh is determined by the following factors.

- Whether the data in the materialized view container table are clustered by a time attribute.
- Whether a concatenated index is available on the materialized view keys.

These factors can be addressed by "partitioning the materialized view container by time, like the fact tables, and by creating a local concatenated index on the materialized view keys" [12].

A fast refresh can be conducted on the materialized views in most of the cases. However some of the exceptional cases in which a fast refresh cannot be performed on the materialized view are listed.

- "When there is more than one table in an aggregated materialized view, and when any DML on the fact tables, other than a direct load has occurred since the last full refresh was performed".
- When the materialized view contains detail relations that are views or snapshots.
- When the materialized view contains AVG(x) without COUNT(x).
- When the materialized view contains VARIANCE(x) without COUNT(x) and SUM(x).
- When the materialized view contains STDDEV(x) without COUNT(x) and SUM(x)." [12]

2.1.3 Selective Manual Refresh [12]

The on demand refresh or the manual refresh allows the user to decide when to recalculate the views. In a manual refresh scenario, the fast refresh technique cannot be used and a complete

refresh has to be performed on the view. However, since the view is selectively refreshed this method can be more efficient when compared to the unconditional and the fast refresh strategies.

If a database contains more than one materialized view, then there are two different approaches for a selective manual refresh. All the views in the database can be refreshed or a selected set of views can be refreshed.

2.2 Triggers [10]

The SQL trigger provides a way for the users of the database to actively manage a set of views. Whenever there is an insert, update or a delete operation on the data triggers can be used to manage the views. The statements defined within the trigger are invoked automatically whenever there is a SQL insert, update or delete operation is performed. The SQL triggers can use stored procedures to perform additional processing when the trigger is executed.

The SQL trigger cannot be directly called by an external application. The definition of the SQL trigger is stored within the database management system. One of the main applications of triggers is to maintain derived data.

2.3 Computational Journalism [11]

Computational journalism is the study of how computational concepts can be used for journalism. Various concepts of journalism can be benefited from the use of computational concepts. Some of the fields of journalism that can benefit from the concepts of computational concepts are newsgathering, investigative journalism, verification/fact finding, authoring/printing/publication/broadcasting of news, sharing and distribution of news stories, editing and commenting on news, etc.

“The goal in this field of Computational Journalism is to study the overlapping interests between computation and journalism to define how both of these can help with information gathering and dissemination for and by citizens to achieve an engaged and more actively participating citizenry”. [11]

Since the concepts of journalism like fact finding and investigative journalism involves processing huge amounts of data, the concepts of data mining can be effectively can be used in fact finding and investigative journalism.

In this thesis we are exploring the possibility of using data mining concepts to solve some of the problems associated with fact finding.

CHAPTER 3

PROBLEM SPECIFICATION

This chapter provides the definition for the view update problem and the data model. A brief description of the designed approach which is used to solve the view update problem is provided.

3.1 Data model

The views used are materialized and represent multiple significant facts. They are collated from the data present in the different database tables. These views can be generated using a general SQL expression as shown in the following figure.

$$
\begin{aligned}
&SELECT\ (table.id,\ \ agg\_function\ (table.p_i)) \\
&GROUP\ BY\ id \\
&ORDER\ BY\ agg\_function(table.p_i) \\
&LIMIT\ k
\end{aligned}
$$

Figure 3.1 - General SQL expression

The database consists of a master table and multiple views defined on the master table. The schema for the master table is $< id, p1, p2, \ldots . pn >$, where the field id is used for identification of subjects like NBA players. The fields $p_1 \ldots . . p_n$ reflect the different parameters recorded within the database. Each view V in the database contains the data derived from R.

Consider the example where a view for the top-100 scorers has to be obtained. This can be obtained by using the following query on the master table.

```
SELECT player_id, sum (points)
AS total_points
FROM player_stats
GROUP BY player_id
ORDER BY total_points DESC
LIMIT 100
```

Figure 3.2 - Example Query to obtain top - 100 scorers

The data in the table 3.2 acts as the master table, which contains the game data for each player. The view for the top-100 scorers is shown in table 3.1, which is obtained by executing query in Figure 3.2 on the table 3.2. The table 3.2 contains just two unique players. Hence, the view contains just two players and their total score arranged in ascending order. The view is illustrated in table 3.1.

Table 3.1 - Illustration of View

player_id	Sum(Points)
WESTMA01	158
KITEGR01	87

Table 3.2 - Illustration of Master table

player_id	game_year	game_dat	opposition	Points
WESTMA01	1983	Dec. 12	SEA	8
WESTMA01	1983	Dec. 14	POR	1
WESTMA01	1983	Dec. 16	IND	3
WESTMA01	1983	Dec. 17	LAC	2
WESTMA01	1983	Dec. 19	LAL	9
WESTMA01	1983	Dec. 21	HOU	2
WESTMA01	1983	Dec. 23	GSW	12
WESTMA01	1983	Dec. 26	BOS	1
WESTMA01	1983	Dec. 27	ATL	9
KITEGR01	1983	Dec. 29	NYK	8
WESTMA01	1983	Dec. 29	DAL	3
WESTMA01	1983	Dec. 30	LAL	2
KITEGR01	1983	Dec. 31	WA1	10
KITEGR01	1983	Feb. 1	SAC	0
WESTMA01	1983	Feb. 10	NYK	14
WESTMA01	1983	Feb. 11	DEN	2
KITEGR01	1983	Feb. 13	HOU	4
WESTMA01	1983	Feb. 17	ORL	8
KITEGR01	1983	Feb. 19	MIA	3
KITEGR01	1983	Feb. 2	GSW	4
WESTMA01	1983	Feb. 2	DAL	9
KITEGR01	1983	Feb. 20	DET	2
KITEGR01	1983	Feb. 22	SAS	2
KITEGR01	1983	Feb. 24	NJN	4
KITEGR01	1983	Feb. 26	DEN	4
KITEGR01	1983	Feb. 27	POR	7
KITEGR01	1983	Feb. 4	LAC	0
WESTMA01	1983	Feb. 4	DET	4
KITEGR01	1983	Feb. 6	BOS	4
WESTMA01	1983	Feb. 7	ATL	13
KITEGR01	1983	Feb. 8	MIL	1
WESTMA01	1983	Feb. 8	CHA	4
WESTMA01	1983	Jan. 10	CHA	6
KITEGR01	1983	Jan. 11	PHI	4
KITEGR01	1983	Jan. 12	NJN	2
WESTMA01	1983	Jan. 13	UTA	0
WESTMA01	1983	Jan. 15	PHO	4
KITEGR01	1983	Jan. 15	UTA	3
KITEGR01	1983	Jan. 16	SAC	5
WESTMA01	1983	Jan. 16	IND	5
WESTMA01	1983	Jan. 18	ORL	0
KITEGR01	1983	Jan. 19	LAL	2
WESTMA01	1983	Jan. 2	DEN	13
WESTMA01	1983	Jan. 20	MIA	6
KITEGR01	1983	Jan. 21	BOS	11
WESTMA01	1983	Jan. 23	DET	6
KITEGR01	1983	Jan. 23	CLE	7
WESTMA01	1983	Jan. 24	SAS	8
KITEGR01	1983	Jan. 25	MIA	0
WESTMA01	1983	Jan. 26	DEN	4

3.2 Framework of the view update procedure

The previous section describes how a view can be generated from the data present in the database. If the data in the master table is updated, then the views also have to be updated. This section gives the formal definition for the view update problem.

If a new tuple t_{n+1} is inserted into the set R, then the corresponding view V changes. The problem of maintaining the data within view V accurate, when the data in R changes is called as the view update problem.

If the records in the table 3.3 are added to the records present in the table 3.2, then the data in the view shown in table 3.1 changes. Thus, the view needs to be recalculated. The view is recalculated by executing the same query which created the view. Once the view is recalculated, the changes will be reflected in the view as shown in table 3.4.

Table 3.3 - Illustrating view update showing records added to the database

GRIFFAD01	2001	Jan. 1	SAS	0
AMAECJO01	2001	Jan. 1	SAS	0
WEATHCL01	2001	Jan. 1	PHI	4
ARMSTDA01	2001	Jan. 1	GSW	5
HAMILZE01	2001	Jan. 1	MIN	7
OVERTDO01	2001	Jan. 1	PHO	3
FISHEDE01	2001	Jan. 1	CHA	7
LARUERU01	2001	Jan. 1	TOR	7
GATLICH01	2001	Jan. 1	PHI	13
MURRALA01	2001	Jan. 1	SAS	22

Table 3.4 - Illustrating view update showing updated view

player_id	Sum(Points)
WESTMA01	158
KITEGR01	87
MURRALA01	22
GATLICH01	13
HAMILZE01	7
FISHEDE01	7
LARUERU01	7
ARMSTDA01	5
WEATHCL01	4
OVERTDO01	3
GRIFFAD01	0
AMAECJO01	0

There are many different strategies which can be used to update the view. Each update strategy updates the view at different instances of time. A naïve approach for updating the view is the absolute approach. In the absolute approach, if the data in the database is updated, then the view is recalculated using a trigger. Thus, the number of queries executed to maintain the view accurate is equivalent to the number of times the database is updated.

The number of times the view is updated can be considerably reduced if a selective update mechanism is used. In this thesis, an approach has been designed where the view is updated selectively based on the results of the prediction algorithm. Consider a set of tuples $\{t_{n+1,}, t_{n+2}, t_{n+3}, \dots \dots \dots t_k\}$ have been inserted into R. A set of prediction values which estimates, the time interval after which the view has to be updated. These prediction values are contained in the set P. When a record t_i is inserted into the set R, the prediction value for the entity being inserted is checked to find out whether the view needs to be updated or not. If the prediction value is zero, then the view is recalculated. However, if the prediction value is not zero, then the prediction value is decremented by one.

The Table 3.5 contains the minimum number of games a player has to play before achieving a milestone. From the table 3.5 we can see that the player 'WEATHCL01' has to play 11 more games to reach the top-100 scorers list. The Table 3.5 contains the count value after the data is inserted. Suppose say the player 'GRIFFAD01' plays a game and the data related to the game is entered into the database, then the prediction values for the player 'GRIFFAD01' is decremented by 1. The results of the operation are shown in the table 3.6. Suppose the player 'WEATHCLO1' has played 11 games, then the minimum number of matches required to reach the top-100 point scorers list becomes zero. Hence, the view top-100 scorers has to be recalculated.

Table 3.5 - Prediction values before insert

player_id	min_matches_total_points	min_matches_total_assists	min_matches_total_three_pointers
AMAECJO01	432	926	2062
ARMSTDA01	167	27	25
FISHEDE01	568	144	183
GATLICH01	7	942	1085
GRIFFAD01	783	467	616
LARUERU01	874	647	272
MURRALA01	38	372	25
OVERTDO01	619	203	433
WEATHCL01	11	178	1767

Table 3.6 - Prediction values after insert

player_id	min_matches_total_points	min_matches_total_assists	min_matches_total_three_pointers
AMAECJO01	432	926	2061
ARMSTDA01	167	26	24
FISHEDE01	567	143	183
GATLICH01	7	941	1084
GRIFFAD01	**782**	**466**	**615**
LARUERU01	873	647	272
MURRALA01	37	371	24
OVERTDO01	619	203	433
WEATHCL01	11	177	1767

Selective update procedure provides an efficient way of maintaining the data in the view consistent. The number of queries executed to maintain the view efficient is lesser, when compared to the absolute approach.

3.3 Triggers

A trigger is executed when the view needs to be recalculated. The trigger will contain the query which is used to generate the view. If the trigger is executed, then the query present within the trigger is executed causing the view to be recalculated.

If ‘WEATHCLO1’ plays the 11 games necessary to reach the top-100 scorers’ view, then the view top-100 scorers has to be recalculated. The top-100 scorer’s view is recalculated by invoking a trigger. The invoked trigger contains a set of queries for recalculating the top-100 scorer’s view.

CHAPTER 4

THE SOLUTION FRAMEWORK

4.1 System architecture

The solution to the selective update problem is provided using two different algorithms, the prediction algorithm and the decision algorithm, respectively. The entire system used for solving the problem is illustrated in the following figure.

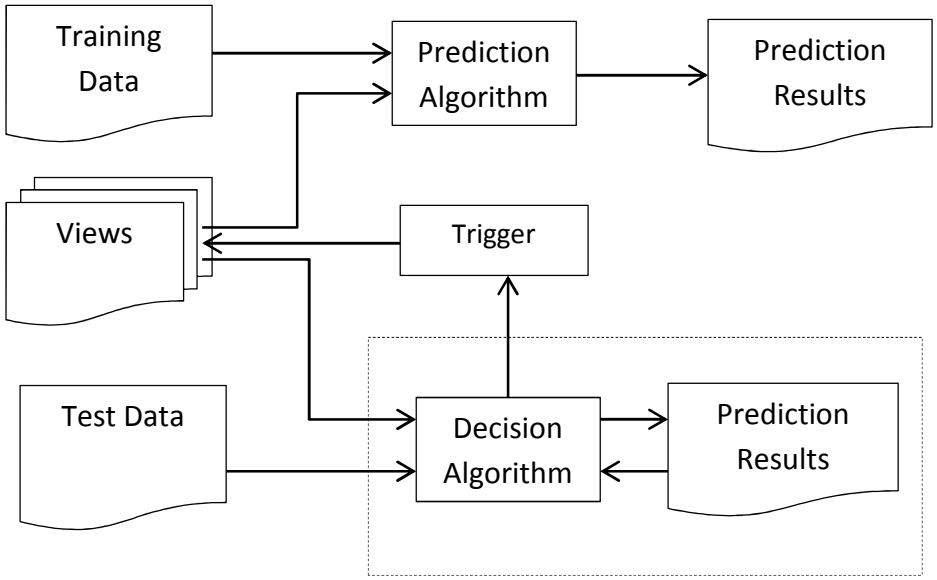

Figure 4.1 – System architecture

The system consists of three different parts.

- The database.
- The algorithms.

- The data maintained in the program memory.

Each part is described as follows.

4.1.1 The Database

The database consists of the training data, test data and the views. The views are defined on the training data. After the completion of the prediction phase, the prediction results table is added to the database.

The dataset contains NBA data from the year 1991 to 2009. NBA data from the year 1991 to 2000 is used as the training data. The data from the year 2001 to 2009 is used as the test data. The views in the database represent different significant events like top-100 scorers', top-100 total assists' and top-100 three pointer scorers'. These views are derived by executing SQL queries on the test data. The SQL queries are mentioned in the previous chapter.

4.1.2 The Algorithms

The prediction and the decision phases are implemented by the prediction and the decision algorithms, respectively. The prediction algorithm takes training data as input. Using the training data, the prediction algorithm computes the number of matches each player needs to reach a milestone. The concept of prediction interval is used by the prediction algorithm.

The input for the decision algorithm, are the results from the prediction algorithm and the test data. The decision algorithm then inserts each record from the test data one by one. The predicted number of matches for each player is decremented, whenever a record pertaining to that player is inserted into the database. If the record being inserted is Kobe Bryant then the count maintained against Kobe Bryant is decremented. Once the predicted number of games becomes zero, the view needs to be recalculated. The view is recalculated by invoking the trigger. If the view is recalculated, then the prediction algorithm recalculates the prediction values. If Kobe Bryant plays the 10 games needed to reach the top-100 scorers' list, then the value in the prediction results table would now be zero. Hence, the view is recalculated and the prediction value for Kobe Bryant is recalculated. All the

records, which include the new 10 records inserted are used to calculate the new prediction value for Kobe Bryant.

Different scenarios are encountered by the decision algorithm during execution. In one such scenario, a player will not figure in the training data. This means that the prediction algorithm has not predicted the number of games required for that particular player to achieve a milestone. In this scenario the decision algorithm waits until the particular player plays a stipulated number of games before updating the prediction values for that particular player. If Dirk Nowitzki has not played any previous game but his record is encountered by the decision algorithm, then the decision algorithm waits until Dirk plays 50 games before computing the prediction values for Dirk Nowitzki.

4.1.3 Data Present In Program Memory

To enable faster access of data, certain amount of data has to be stored in the program memory. The test data and the prediction results are frequently accessed by the decision algorithm in this approach. Hence, they are maintained in the program memory to enable faster access to the data. The data maintained in the prediction results table also double as the counter. Hence, an additional counter to count the number of games a player has played is not required.

If a particular player's prediction results have to be updated, then the prediction results table needs to be searched for the player's records. Thus, a linear search mechanism is used to find the player whose prediction values have to be updated.

CHAPTER 5

ALGORITHM DESCRIPTION

To achieve the aim of selectively updating the views two algorithms were needed. The prediction and the decision algorithms were developed to achieve the goal. This chapter provides the description on the working of the designed algorithms.

5.1 Prediction algorithm

The prediction algorithm follows the following steps in predicting the number of games needed by a particular player to reach a milestone.

a. Predict the lower and upper limits of the prediction interval for a particular player.
b. Find the number of points that a player has to score to reach a milestone.
c. Find the number of games needed by the player to achieve the milestone is calculated.

5.1.1 Predicting lower and upper limits of the prediction interval

Prediction interval estimates the interval in which future values may fall with a certain probability, given the observed data. To use the concept of prediction interval the data is assumed to be in the form of a normal distribution. The first step in determining the prediction interval is obtaining the statistics of the data from the training data.

The Table 5.1 shows the statistical parameters like mean and standard deviation values for different attributes like points, total assists made and total three pointers made.

Table 5.1 - Mean and standard deviation values

player_id	avg(Points)	avg(Total_Assists)	avg(Three_Pointers_Made)	stddev(Points)	stddev(Total_Assists)	Stddev(Three_Pointers_Made)
AMAECJO01	7.9141	0.9192	0.0202	6.4517	0.9231	0.2238
ARMSTDA01	12.1966	5.302	1.3077	7.2594	3.3282	1.3256
FISHEDE01	5.9906	3.0535	0.5912	5.2128	2.4675	0.9564
GATLICH01	10.6611	0.6779	0.0763	6.9527	0.8763	0.3179
GRIFFAD01	5.3923	1.7769	0.2154	4.787	1.8071	0.4801
LARUERU01	4.8493	1.274	0.5753	4.1303	1.5638	0.8588
MURRALA01	12.5736	1.4585	0.6755	7.1032	1.4404	0.8469
OVERTDO01	4.6896	2.09	0.2275	5.0147	2.297	0.5845
WEATHCL01	13.1614	1.6657	0.0231	6.7465	1.4566	0.2564

Once the statistical data is obtained, then the lower limit and the upper limit of the prediction interval are obtained. The lower limit of a prediction interval determines lowest possible score that a player will obtain in the next match. The upper limit of prediction is the highest possible score that a player will obtain. Equation 1 shows the calculation of the lower and upper limits of prediction interval.

Equation 1

$$\text{Lower limit of prediction interval (L)} = \mu - \sigma z$$

$$\text{Upper limit of prediction interval (U)} = \mu + \sigma z$$

In the above equation, μ is the mean, σ is the standard deviation of the parameter considered and z is the standard score. The standard score determines the probability with which the score will fall within prediction interval. If the standard score is set to 2.58, then the probability of the player scoring within the prediction interval in the next match is 99%. The table 5.2 shows the prediction interval and the associated standard score.

Table 5.2 - Standard score values for different prediction intervals

Prediction Interval	Standard Score
50%	0.67
90%	1.64
95%	1.96
99%	2.58

The table 5.3 shows the lower and upper limit of prediction values, respectively for each player. The values are obtained by using the formulae in Equation 1. The standard score of 1.96 for 95% accuracy is used for this calculation. From the values in table 5.3, it can be inferred that 'JORDAMI01' will score between 13 to 47 points in the next game.

Table 5.3 - Prediction interval values

Player ID	Min Interval	Max Interval
JORDAMI01	13.0446	47.1976
IVERSAL01	7.009276	45.334724
MALONKA01	12.84104	39.25596
WADEDW01	7.89506	41.69134
CARTEVI01	7.29218	41.75682
ONEASH01	6.52042	41.39078
ROBINDA01	5.135516	39.544884
WILKIDO01	1.552592	43.015608
OLAJUHA01	4.010888	40.259912
DUNCATI01	8.2569	35.8341
RICHMMI01	7.41486	36.67374
WEBBECH01	7.99238	35.58722
PETRODR01	8.148828	34.680172
ROBINGL01	7.171276	35.177324

5.1.2 Finding the number of points that each player has to score to reach a milestone

To find the number of games that particular player has to play to reach a milestone, the number of points that a player has to score to achieve a milestone has to be calculated. This can be done by using the prediction interval values and the data present in the view.

Consider a player aspiring to enter the top-100 scorers list. The point difference between the lowest scorer in the top-100 scorers list and the points of the player has to be found. A milestone can also be achieved when a player within the top-100 scorers list changes rank position. If a player is already in the top-100 scorers list, then the point difference between the player ranked above him and the player in question is calculated.

5.1.3 Predicting the number of games needed to reach a milestone

The minimum number of matches that a player will need to achieve a milestone can be defined using the following equation.

Equation 2

$$\text{Minimum number of matches} = \frac{(\delta) - (\Sigma)}{U}$$

Where U is the upper limit of prediction interval Σ is the total score of the player δ is the minimum score present in the view.

Consider a player like Kobe Bryant, suppose his total score achieved is 7000 points and the prediction interval score is 15 to 25. If he still needs to get 150 points to reach the top-100 scorers list, then he needs 6 matches scoring at the rate of 25 points each game to reach the top-100 players list.

The pseudo code for prediction algorithm is described as follows.

```
INPUT: a set R of all the records of each player present in the database.
OUTPUT: The set S containing tuples with the attributes player id and the number of
games needed to achieve each significant event

BEGIN


        C is a table which contains <player_id, stddev, mean, total_points>
        V is a view of top-100 scorers which contains <player_id, total_points>


        Foreach element in C

                Let k be the element read from c;

                Min Score = k.mean – k.stddev * z;

                // Z is the prediction interval(Refer table 5.2)
                //Find the minimum score with among all the records present in table V

                Min View Score = Min(V.total_points);
                Min Matches = (k.total_points - Min View Score)/Min Score;

        End While
End
```

Figure 5.1 - Pseudo code for prediction algorithm

The results of the prediction algorithm are described in the Table 5.4. The table contains the minimum number of games needed by each player to enter into three different views.

Table 5.4 - Minimum and maximum score values

player_id	min_matches_total_points	min_matches_total_assists	min_matches_total_three_pointers
AMAECJO01	432	926	2062
ARMSTDA01	167	27	25
FISHEDE01	568	144	183
GATLICH01	7	942	1085
GRIFFAD01	783	467	616
LARUERU01	874	647	272
MURRALA01	38	372	25
OVERTDO01	619	203	433
WEATHCL01	11	178	1767

5.2 Decision Algorithm

The decision algorithm uses the results from the prediction algorithm to make decisions on when to execute the trigger. If data is inserted into the database for a particular player, then the prediction value for that particular player is decremented. If the value becomes zero for a particular player, then the view is recalculated.

Now consider the input data which is illustrated by Table 5.5.

Table 5.5 - Input Data

player_id	game_year	game_Date	opposition	Three_Pointers_Made	Total_Assists	Points
GRIFFAD01	2001	Jan. 1	SAS	0	2	0
AMAECJO01	2001	Jan. 1	SAS	0	2	0
WEATHCL01	2001	Jan. 1	PHI	0	2	4
ARMSTDA01	2001	Jan. 1	GSW	0	11	5
HAMILZE01	2001	Jan. 1	MIN	0	0	7
OVERTDO01	2001	Jan. 1	PHO	1	3	3
FISHEDE01	2001	Jan. 1	CHA	2	0	7
LARUERU01	2001	Jan. 1	TOR	1	2	7
GATLICH01	2001	Jan. 1	PHI	0	0	13
MURRALA01	2001	Jan. 1	SAS	3	4	22

If the record for player “GRIFFAD01” is inserted into the database table, then the count values for that player should be decremented. The count values after the insert is shown in Table 5.6.

Table 5.6 - Count Values after one insert

player_id	min_matches_total_points	min_matches_total_assists	min_matches_total_three_pointers
AMAECJO01	432	926	2062
ARMSTDA01	167	27	25
FISHEDE01	568	144	183
GATLICH01	7	942	1085
GRIFFAD01	782	466	615
LARUERU01	874	647	272
MURRALA01	38	372	25
OVERTDO01	619	203	433
WEATHCL01	11	178	1767

From Table 5.6 we can see that the minimum number of matches needed by “GRIFFAD01” is decremented by 1 when compared to the values in Table 5.4. However, for the player ‘GRIFFADO1’ the value is not 0. Hence, the view is not recalculated.

If “WEATHCL01” plays 11 games, then after 11 records are inserted into the database. Thus, the count value for ‘WEATHCL01’ becomes zero. This is illustrated in the Table 5.7. Hence, the view top-100 scorers’ have to be recalculated.

Table 5.7 - Prediction values when top-100 scorers view needs to be recalculated

player_id	min_matches_total_points	min_matches_total_assists	min_matches_total_three_pointers
AMAECJO01	432	926	2062
ARMSTDA01	167	27	25
FISHEDE01	568	144	183
GATLICH01	7	942	1085
GRIFFAD01	782	466	615
LARUERU01	874	647	272
MURRALA01	38	372	25
OVERTDO01	619	203	433
WEATHCL01	0	156	1756

If the trigger is invoked to recalculate the view, then the decision algorithm needs to recalculate the prediction values for “WEATHCL01”. The prediction results can be recalculated by invoking the prediction algorithm.

The prediction values are recalculated by taking the updated values shown in table 5.8. The prediction results after the values are recalculated are shown in Table 5.9.

Table 5.8 – Updated data

GRIFFAD01	2001	Jan. 1	SAS	0
AMAECJO01	2001	Jan. 1	SAS	0
WEATHCL01	2001	Jan. 1	PHI	4
ARMSTDA01	2001	Jan. 1	GSW	5
HAMILZE01	2001	Jan. 1	MIN	7
OVERTDO01	2001	Jan. 1	PHO	3
FISHEDE01	2001	Jan. 1	CHA	7
LARUERU01	2001	Jan. 1	TOR	7
GATLICH01	2001	Jan. 1	PHI	13
MURRALA01	2001	Jan. 1	SAS	22

Table 5.9 –Updated view

player_id	Sum(Points)
WESTMA01	158
KITEGR01	87
MURRALA01	22
GATLICH01	13
HAMILZE01	7
FISHEDE01	7
LARUERU01	7
ARMSTDA01	5
WEATHCL01	4
OVERTDO01	3
GRIFFAD01	0
AMAECJO01	0

Table 5.10 - Recalculated prediction score values

player_id	min_matches_total_points	min_matches_total_assists	min_matches_total_three_pointers
AMAECJO01	432	926	2062
ARMSTDA01	167	27	25
FISHEDE01	568	144	183
GATLICH01	7	942	1085
GRIFFAD01	782	466	615
LARUERU01	874	647	272
MURRALA01	38	372	25
OVERTDO01	619	203	433
WEATHCL01	3	165	1747

There will be scenarios where a new player enters the NBA league. If a new player enters, then there will be no previous recorded history for a new player. The algorithm will wait for a stipulated amount of time, so that adequate statistics can be calculated about the player before predicting the number of matches the player will take to reach a milestone.

```
INPUT: The set S contains the prediction results <player_id, min_matches>
       The table T contains all the player data <player_id, points>
       The table R contains data related to the new game played <player_id, points>
       The view V <player_id, total_points>

BEGIN

       Foreach element in R

              Let k be an element read from the set R;
              Let l be a record from S where l.player_id == k.player_id
              l.count --; // Decrement the value of count
              if (l.Count == 0) then
                     Insert record k to the set T;
                     Update set S with new prediction data;
                     Recalculate the view V;
              else
                     Insert record K to the set T;

              end if-else
       end While
END
```

Figure 5.2 - Pseudo code for decision algorithm

CHAPTER 6

EXPERIMENTAL RESULTS

This chapter explains the different setups for the experiments performed and also describes the results obtained from each experiment. These results are used to evaluate the performance of our methods in different settings.

6.1 Dataset for experiments

For our experimental setup we use programs developed in Java. The dataset used is stored as database tables in MySQL. Two datasets, the training and the test data, are used. Based on the statistical characteristics of the training data, the prediction algorithm obtains a set of initial prediction results. The test data is then used to insert the records and test the performance of the decision algorithm.

To obtain the training data and the test data, the existing data was split into half. The training data contains 218,219 player records from the year 1991 to 2000. The test data contains 211,188 records from the year 2001 till the year 2009.

6.2 Experimental Results and Comparison

This section describes the various experiments, analysis of the experimental results and the results of the experiments.

6.2.1 Benchmark test

The benchmark test is used to establish a benchmark against which our experiments can be compared. In the benchmark test, if there is an update operation performed on the database, then the trigger is invoked to recalculate the view. The update is also checked for a significant fact.

In the benchmark test, if 10,000 records are inserted into the database, then the trigger is invoked 10,000 times. The table 6.1 shows some of the significant events captured by the benchmark test.

Table 6.1 - Results of benchmark test

player_id	match_no	comments
JONESED01	382	ADDITION TO TOP 100 ASSISTS
MURRALA01	1116	ADDITION TO TOP 100 THREE POINTERS
PIERCPA01	1757	ADDITION TO TOP 100 THREE POINTERS
DAVISDA01	1790	ADDITION TO TOP 100 POINTS
ROSEJA01	5297	ADDITION TO TOP 100 POINTS
NASHST01	6283	ADDITION TO TOP 100 ASSISTS
DUNCATI01	6595	ADDITION TO TOP 100 POINTS
NASHST01	7439	ADDITION TO TOP 100 THREE POINTERS
MURRALA01	7617	ADDITION TO TOP 100 POINTS
STOJAPR01	8711	ADDITION TO TOP 100 THREE POINTERS
BRYANKO01	8714	ADDITION TO TOP 100 POINTS
BARRYJO01	9604	ADDITION TO TOP 100 THREE POINTERS
STOJAPR01	9694	ADDITION TO TOP 100 THREE POINTERS
BARRYJO01	9872	ADDITION TO TOP 100 THREE POINTERS

6.2.2 Number of triggers invoked for different prediction intervals

The prediction interval is a parameter which determines the accuracy of the predicted values. If the prediction interval is 99%, then there is a 99% chance that the next possible value will appear between the predicted values. Varying the prediction interval results in different accuracy, however this also affects the number of times the trigger is invoked. This experiment illustrates the effect on number of times a trigger is invoked when the accuracy is varied.

Four different settings for the prediction interval and three different views are used to evaluate the algorithm. The results of the experiments are illustrated as follows.

Table 6.2 - Number of time trigger invoked for different prediction intervals

Prediction Interval	View_top_hundred	View_total_assists	View_total_three_pointers
50%	195	157	1002
90%	592	493	1352
95%	617	520	1399
99%	635	533	1447

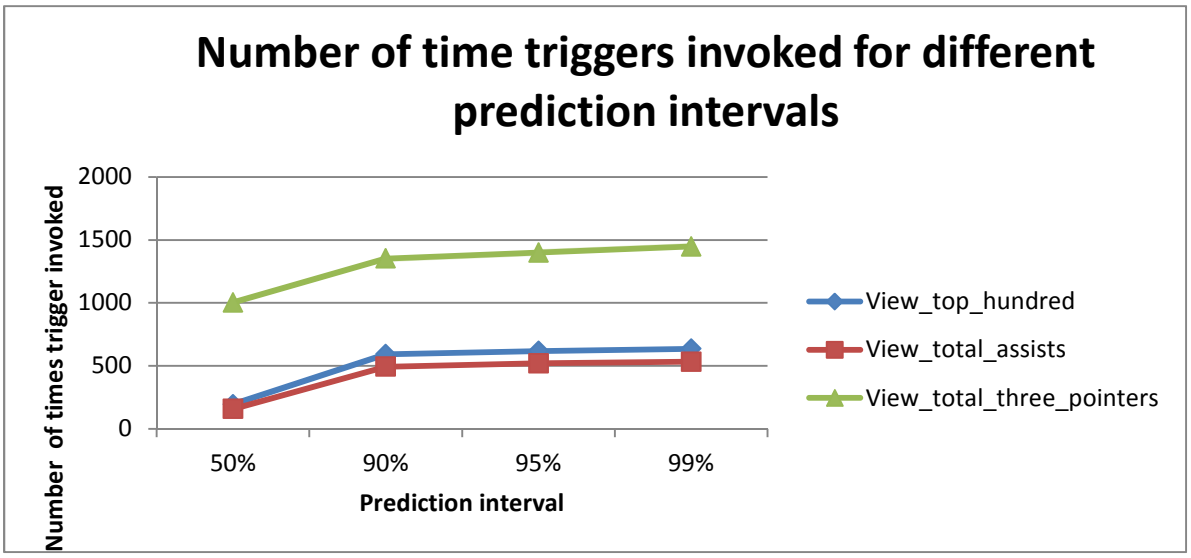

Figure 6.1 - Graph Number of times trigger invoked for different prediction intervals

From the figure 6.1 it can be observed that if the prediction interval i.e. the accuracy increases, then the number of times the trigger is invoked also increases.

Suppose a player achieves a milestone at the 382nd match. If the algorithm is using a prediction interval of 99%, then the significant event may be captured after the 385th match the player plays. If the prediction interval being used is 50%, then the significant event may be found after the 410th match the player plays. This example is illustrated in the following experiment.

6.2.3 Average difference in the number of matches required to find the significant event

This experiment is used to measure the difference in the number of matches required to find each significant event, against the results of the benchmark test. The benchmark test has captured at which match the significant event happens. Using the data from the benchmark test, and this experiment, the difference in matches at which the significant event is captured for each player can be calculated. Once the difference in matches for each player is calculated, the average difference in matches is calculated and tabulated. The results can be illustrated by the following figure.

Table 6.3 - Average difference in matches

Prediction Interval	99%	95%	90%	50%
Average Difference in Matches	3.727273	10.625	23	37.72727

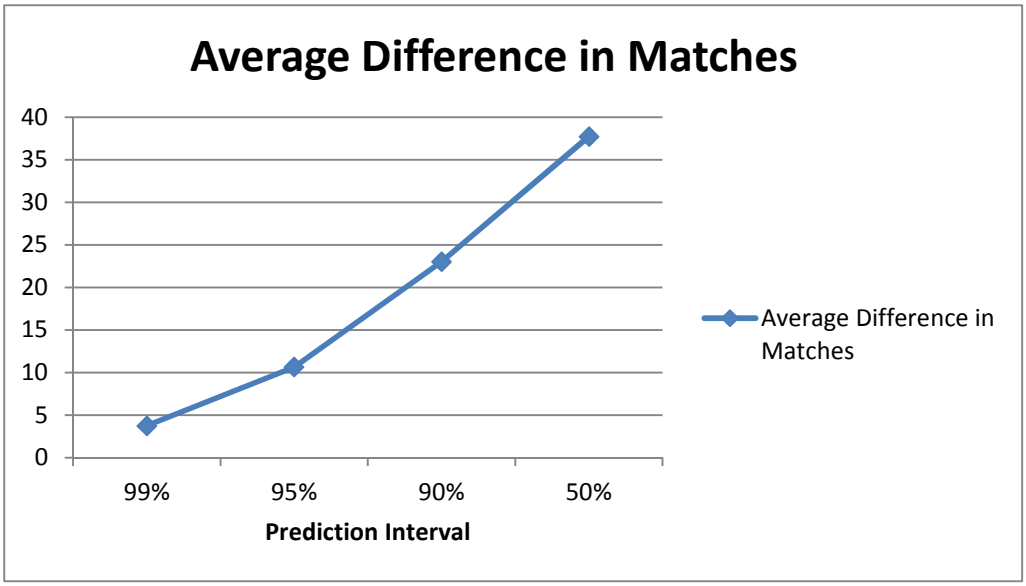

Figure 6.2 – Graph average difference in matches

From the figure 6.2 we can understand that if the prediction interval decreases, then the difference between the games also increases.

6.2.4 Comparison of results when a particular number of previous games of a player is considered

This experiment measures the number of times a trigger is invoked when the number of previous games a player has played is considered for statistical evaluation. By altering the number of previous

games considered for making the prediction, the accuracy of the model can be altered. If a lesser number of records are given to the prediction algorithm, then the algorithm has faster execution time. However, the accuracy of the prediction algorithm reduces. Similarly, if a larger number of records are provided to the prediction algorithm, then the accuracy of the prediction algorithm increases. However, the execution time of the prediction algorithm also increases.

The following result shows the effects of the different number of times a trigger is invoked when the number of previous games considered is altered.

Table 6.4 - Previous n games

Previous n games selected	10	50	100
View_top_hundred_points	145	189	195
view_total_assists	123	149	157
view_three_pointers_made	956	992	1002

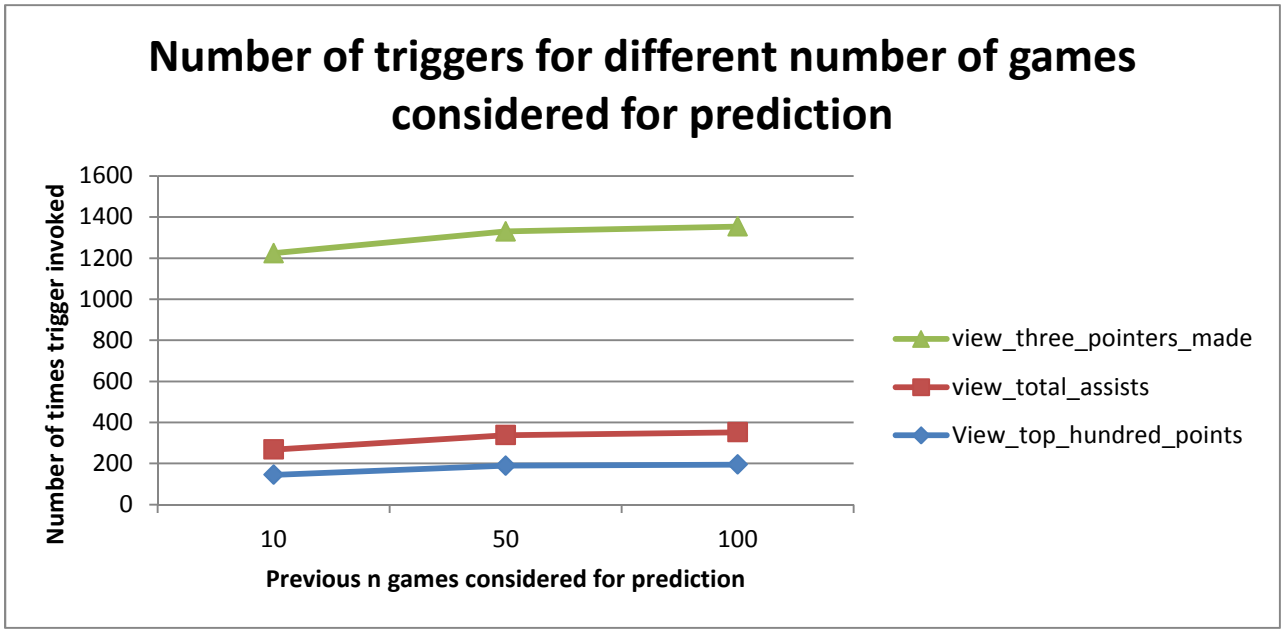

Figure 6.3 – Previous n games

From the previous experiments it can be observed that for the accuracy to be high, the triggers have to be invoked frequently. From the figure 6.3 it can be seen that if the previous n games considered increases, then the number of times the trigger is invoked is also higher.

CHAPTER 7

CONCLUSION

With the internet growing, large amounts of data are being added to the internet at a rapid pace. Large datasets are becoming very common in real life. Analyzing this vast amount of data and finding significant facts from these large datasets will be a necessity in the near future. Thus an efficient way of finding significant facts from a large dataset is essential.

In this thesis we have designed a two phased approach for solving the problem of significant fact finding from large datasets efficiently. For the implementation of the two phased approach, design and implementation of two different algorithms were needed. Thus, the prediction algorithm and the decision algorithm were designed and implemented.

The results of the approach have been compared with a benchmark test and it has been proven that our approach is more efficient when compared to the benchmark test and it also achieves an accuracy which is close to the accuracy achieved by the absolute method.

REFERENCES

[1] http://en.wikipedia.org/wiki/Prediction_interval#cite_note-MedicalStatisticsA2-3.

[2] José G. Ramírez, W.L. Gore and Associates Statistical Intervals Confidence, Prediction, Enclosure. SAS white paper pages Section 1 Pages 1-11.

[3] http://en.wikipedia.org/wiki/Relational_model.

[4] Jens Lechtenb¨orger. The Impact of the Constant Complement Approach Towards View Updating. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (PODS '03). ACM, New York, NY, USA, 49-55.

[5] Nabil I. Hachem, Ke Qiu, Michael A. Gennert & Matthew O. Ward Managing Derived Data In The Gaea Scientific DBMS, VLDB 1993: 1- 12.

[6] Hegner, Stephen J., An order-based theory of updates for closed database views, Annals of Mathematics and Artificial Intelligence, 1-2 (January 2004), 63-125.

[7] Hegner, Stephen J., Uniqueness of update strategies for database views, in Foundations of Information and Knowledge Systems, Second International Symposium, FoIKS 2002, Salzau Castle, Germany, February 20-23 2002, Proceedings, Thomas Eiter and Klaus-Dieter Schewe, editors. Springer-Verlag Lecture Notes in Computer Science, Vol. 2284, 2002, pp. 230-249.

[8] Ashish Gupta and Inderpal Singh. Maintenance of Materialized Views: Problems Techniques, and applications, MIT Press, Cambridge, MA, USA 145-157.

[9] John Grant, John Horty, Jorge Lobo, and Jack Minker. 1993. View updates in stratified disjunctive databases. *J. Autom. Reason.* 11, October 1993 249-267.

[10] Rameez Elmasri, Shamakant B Navathe. Fundamentals of Database Systems. 5$^{t`h}$ edition. 2009. Part 2 Chapter 8 and Chapter 9.

[11] Nick Diakopoulos, Brad Stenger. Presentation on computational journalism, 2009. Pages 1-30.

[12] Oracle Corporation. Oracle8i Tuning Release 8.1.5 A67775-01. Chapter 32.

[13] Jose A. Blakeley , Per-Ake Larson , Frank Wm Tompa, Efficiently updating materialized views, Proceedings of the 1986 ACM SIGMOD international conference on Management of data, p.61-71, May 28-30, 1986, Washington, D.C., United States

[14] Ashish Gupta , Inderpal Singh Mumick , V. S. Subrahmanian, Maintaining views incrementally, Proceedings of the 1993 ACM SIGMOD international conference on Management of data, p.157-166, May 25-28, 1993, Washington, D.C., United States

[15] Eric N. Hanson, A performance analysis of view materialization strategies, Proceedings of the 1987 ACM SIGMOD international conference on Management of data, p.440-453, May 27-29, 1987, San Francisco, California, United States

[16] Oded Shmueli , Alon Itai, Maintenance of views, Proceedings of the 1984 ACM SIGMOD international conference on Management of data, June 18-21, 1984, Boston, Massachusetts

[17] F. W. Tompa , J. A. Blakeley, Maintaining materialized views without accessing base data, Information Systems, v.13 n.4, p.393-406, Oct. 1, 1988

BIOGRAPHICAL INFORMATION

Avinash S Bharadwaj completed his Bachelors in Computer Science and Engineering from Visveswaraya Technological University (VTU) in July 2009. He worked as an intern at Sasken Communication Technologies Pvt Ltd. He started his Masters in Computer Science at The University of Texas at Arlington in fall 2009 and joined The Innovative Databases and Information Systems Research (IDIR) Lab at UT Arlington in spring 2010. His research involves data mining, web data management, and statistical analysis of data.