Journal of International Technology and Information Management

Volume 15 | Issue 3 Article 3

2006

Simulation Model and Analysis of a Data Warehouse

Vikas Agrawal
Fayetteville State University

Udayan Nandkeolyar
The University of Toledo

P.S. Sundararaghavan
The University of Toledo

Mesbah U. Ahmed
The University of Toledo

Simulation Model and Analysis of a Data Warehouse

Vikas Agrawal
Fayetteville State University

Udayan Nandkeolyar
P. S. Sundararaghavan
Mesbah U. Ahmed
The University of Toledo

ABSTRACT

Data warehouses are large complex systems with many interacting nonlinear components. It is an amalgamation of many different systems and integration of such diverse elements is its primary concern. It is often difficult for a data warehouse manager to predict the performance of the system especially when demand pattern for the required data keeps changing. We have developed a simulation model using ARENA simulation package that will simulate the behavior and performance of a data warehouse system environment based on its overall design. Given such a model, a data warehouse manager can walk through various what-if scenarios and can pinpoint the areas of weaknesses in the system. This visibility could result in improved operational performance in a data warehouse.

INTRODUCTION

A data warehouse can be defined as a database of databases. It brings together a wide range of relevant information from many different heterogeneous independent data sources into one centralized data repository to support business analysis activities and decision-making tasks (Haag *et al*., 2005; Han & Kabmer, 2001). In this paper, we explore the application of systems dynamics approach to model a data warehouse environment and estimate its performance. Data warehouses are large complex systems with many interacting non-linear components. A small change in any one component may have a dramatic impact elsewhere in the system. Consequently the behavior of a data warehouse is often unpredictable (Hillard *et al.* 1999). Data warehouse managers may find a simulation model useful in identifying critical components, diagnosing problems and optimizing the overall design. It will allow system managers to pinpoint performance issues or troubleshoot problems.

We developed a simulation model of a hypothetical data warehouse environment using a range of system parameters. We found that such a model could become a useful tool for system managers to identify potential problem areas and take corrective action in a timely manner. It can also be used as a system design tool to improve operational efficiency and to justify investment decisions when it comes to upgrading various system components.

This paper is organized into eight sections. Section one introduces this research. Section two discusses the simulation and data warehouse performance in brief. Section three summarizes the relevant literature in brief. Section four presents the conceptual framework for simulating data warehouse operation. Section five presents the implementation of conceptual framework using Arena simulation package. Section six presents the experimental design setup for our model and Section seven discusses the experimental results in brief. We end with the concluding remarks and future research directions in Section eight.

SIMULATION AND DATA WAREHOUSE PERFORMANCE

Data warehouses are large complex systems with many non-linear interacting components (Hillard *et al.*, 1999). The architecture of a data warehouse by necessity is complex, and includes many elements. The reason for this is that a data warehouse is an amalgamation of many different systems. Integration of diverse elements is its primary concern, and to accomplish this integration, many different systems and processes are necessary. A data

warehouse consists of many architectural components that compose the data warehouse infrastructure. These components include technical infrastructure, data discovery, data acquisition, data distribution, and user analysis to name few (Kimball *et al,.* 1998, Schrader *et al.,* 1997).

The technical infrastructure includes operating system, hardware platform, database management system and network. The hardware, including the operating system, should be versatile so as to run a variety of tools should be able to run on the platform. Data should be able to flow to/from the platform with a minimal amount of effort. The network should minimize complexity, maximize bandwidth and should connect directly to all components and locations of the corporate enterprise that need access to the data warehouse (Schrader *et al.,* 1997). The database management system selection becomes a little more complicated than a straightforward operational system because of the unusual challenges of the data warehouse, especially in its capability to support very complex queries that cannot be predicted in advance.

Data acquisition is the process of loading data into the data warehouse from the various data sources spread across geographical boundaries. Feeding a data warehouse is an ongoing process, and repeatable processes must be in place for this to occur. Data acquisition itself can be broken up into several major sequential steps such as data extraction, data transformation into one standard domain, data loading into interim data store, data aggregation/summarization and finally loading the data into the warehouse (Kimball *et al.,* 1998, Schrader *et al.,* 1997). Each of these steps is complex in its own right and depends upon many factors such as software tools chosen to perform these steps, the architecture of the data layers and methods used for data cleaning to name few.

Once the data is loaded into the data warehouse, one needs to materialize a set of views (i.e. answers to most frequently asked queries) to improve query response time. This process is very involved and time consuming. These materialized views need to be refreshed periodically to reflect the latest changes in information at data sources. Maintaining views is a non-trivial and time-consuming process. Many researchers have addressed this issue (Colby *et al.,* 1997, Gupta 1999, Kalnis 2002, Mumick *et al.,* 1997).

Data distribution is another important component of data warehouse architecture. Some users query the data warehouse directly; others may be geographically distributed and smaller segments of the warehouse can be exported directly to their location for their specific needs. Others may be located in the same area as the warehouse, but for performance reasons a special technology may be used for their unique query needs. These special subsets of the warehouse distributed directly to the users are called *data marts*. The dissemination and propagation of data for the data marts is a separate architectural component, and should be carefully planned (Schrader *et al.* 1997).

Once the data warehouse is properly set up, the data warehouse users need various data mining tools such as query-and-reporting tools, multidimensional analysis tools and statistical tools to mine the data warehouse for valuable information. Vendors such as Actuate's Enterprise Reporting Applications (http://www.actuate.com), Panorama's Business Intelligence Platform (http://www.panoramasoftware.com), Microsoft OLAP tools, Oracle Discoverer and Oracle Reports, and business intelligence software by MicroStrategy (http://www.microstrategy.com) provide data mining tools. Though these tools are powerful, they are seldom comprehensive. Often companies need to customize them to suit their individual requirements which could be very tedious and time consuming.

In summary, the performance of a data warehouse system is difficult to predict because of the following reasons:

1. The data warehouse system comprises of a number of different nonlinear interacting components
2. The data warehouse system is dependent on other information systems for its data, and
3. The level of usage of the data warehouse system is often difficult to predict.

In today's decision support environment where time is such a critical factor, the data warehouse manager must make sure that the end-users get reliable answers to their questions within a reasonable period of time. One method of ensuring timeliness and reliability is to experiment with real system components and find the best possible synergies among them. Experimenting with a real system's components is not only risky, costly and time consuming, but it may also not be possible to experiment with a live system component as it may bring down the entire system.

A simulation modeling technique is a viable approach to predict the behavior and performance of a complex data warehouse system. The simulation approach to data warehouse performance applies the principles of *systems dynamics* used in other real-world simulation applications such as biological, engineering, and nuclear research. Such an approach could also be used to simulate the behavior and performance of the data warehouse system based on its overall design. Such a simulation approach can have a number of benefits:

1. Data warehouse managers can improve overall performance of their systems. They can walk through various what-if scenarios and configure their new or updated systems to be more reliable and efficient.

2. Data warehouse managers can do better risk management and capacity planning. They can justify or negate investment decisions and can accurately predict when they need to update, add or replace specific system parts.

3. Data warehouse managers will have a more comprehensive and deep understanding of the interrelationships between various system components in an existing data warehouse design scenario and the multiple factors influencing the data warehouse performance. This provides an ability not only to understand the existing opportunities for synergies among different system components but also to pinpoint the areas of inefficiencies in the current data warehouse system design architecture.
   In the next section, we briefly summarize the literature review.

LITERATURE SEARCH

Chan (2004) has proposed an enterprise modeling framework for the deployment of data warehouses that provides the information roadmap coordinating source data and different data warehouses across the business enterprise. He has introduced a solution to address data warehousing issues at the enterprise level while avoiding the pitfalls of creating enterprise data warehouses and universal data marts.

Vasilakis *et al*., (2004) have highlighted the advantages of employing data warehousing techniques for storing and analyzing simulation output data. They pointed out that discrete event simulation modeling, which has been extensively used in modeling complex systems, is both computationally expensive and data intensive, and hence it is exceptionally cumbersome to conduct the required output and sensitivity analysis in a spreadsheet or statistical package. Hence there is a tremendous need to use advanced simulation modeling vehicle such as ARENA for building an elaborate simulation model.

Monteiro and Furtado (2005) have applied a simulation technique to the data warehouse to test a fragment directory with a simple data balancing strategy to handle skewed data efficiently in multidimensional partitioned data sets. They argue that the performance of data warehouses largely depends on good partitioning of data and a skew in the data may condemn partitioning effectiveness.

Badri (2003) has developed a simulation based DSS for modeling passenger arrival and departures in an immigration system of a major international airport. He argued that a computer based simulation tool can help in evaluating various parameters of the airport systems such as placements, pathways and queuing components before such components are purchased or installed. One such research published by Hillard, Blecher and O'Donnell (1999) applies the concept of *chaos theory* (discussed in next section) to the operation of data warehouses.

Hillard *et al.,* (1999) argued that data warehouses are large complex systems with many non-linear interacting elements. This creates a tendency for periods of chaotic behavior. The application of chaos theory could help in developing an understanding of the variability of data warehouse performance commonly experienced by the developers and managers of data warehouses. They further argued that chaos theory could be used to minimize the impact of such erratic performance on the users of the system.

Chaos theory is used to understand and make predictions about apparently random behavior of complex systems that have many interacting non-linear components. The key aspect of chaotic systems is that they are very sensitive to small changes in initial conditions (Gleick 1987, Medio 1992). This means that nearly identical systems, with only slight differences in initial conditions, may behave very differently. The basic principles of contemporary

chaos theory originated in the 1960s when weather forecasters were attempting to produce a series of mathematical tools that would comprehensively predict the weather, not only three days out but months or even years ahead of time.

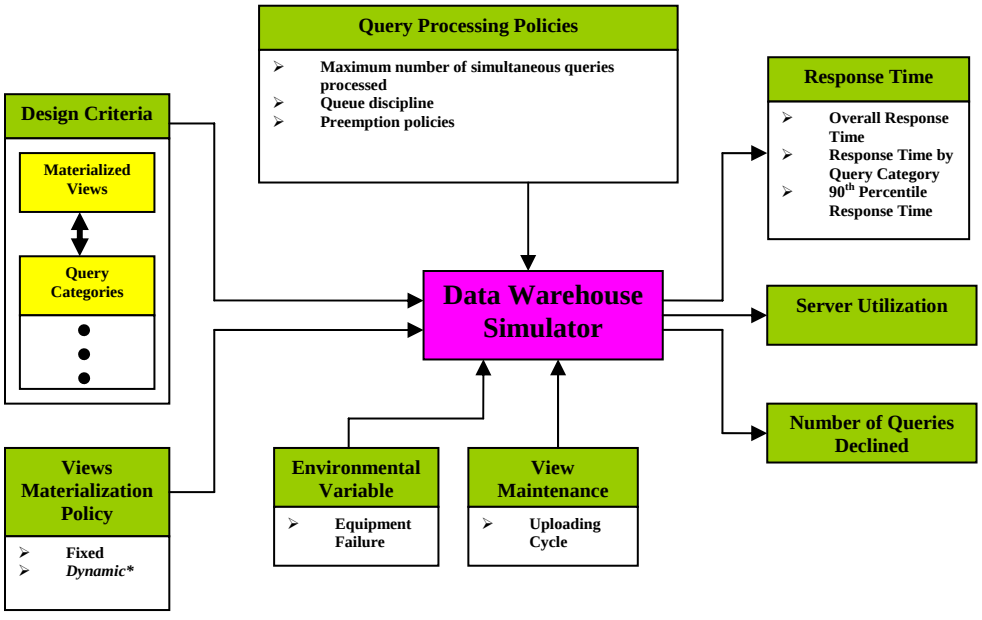
This effort led to the founding of models that accurately mimicked the real world. However, any variation in the input conditions, even at the finest level of accuracy, caused wild variations in the predicted weather outcome in just days or even hours. This effect is termed as the 'Butterfly Effect' as it is often explained using the notion that a butterfly stirring the air today in Bombay might transform storm systems next month in New York.

Researchers went on to build much simpler systems and discovered the same effect, ultimately coming to the conclusion that any system with many non-linear interacting components is a candidate for chaos. Chaos theory has been applied to the analysis of many different types of systems (Gleick, 1987). The most common application is to biological and mechanical systems. The value of chaos theory is that apparent random behavior of a chaotic system can usually be explained by a set of deterministic non-linear rules. In some circumstances this can mean that the unpredictable behavior of a system can be avoided by careful manipulation of certain parameters (Kopel, 1997). We use this as a motivation for our efforts to model a data warehouse using simulation. In the next section, we present the conceptual framework for simulating data warehouse operation.

CONCEPTUAL FRAMEWORK FOR SIMULATING DATA WAREHOUSE OPERATION

In general, the operational performance of a data warehouse depends on the interaction of various parameters and processes such as different types of queries posed to the data warehouse, the frequency of these queries, the length of time required for the uploading cycle to upload the latest information from data sources, the mean time between equipment failures, user behavior and the design of the database management system itself. The network of influences between and within these parameters is very complex. We have developed a conceptual framework of a simulation model for simulating the behavior and performance of a data warehouse. The details of this model are shown in Figure 1. The major components of the conceptual model are:

Figure 1. Conceptual Framework for Simulating Data Warehouse Performance.
(* *Not incorporated in our model*)



Design Criteria

The performance of a data warehouse is directly related to the number of views that are materialized. For more information on data cube and its associated views (or cuboids), reader may refer to Han and Kamber (2001). Given a set of materialized views, a query can either be answered directly from one of the materialized views or be answered indirectly from one of the ancestor materialized views that has data summarization available at lower level of dimensions. If a query is answered directly from one of the materialized views, it will be executed quickly with minimum response delay. On the other hand, if it is answered indirectly using one of the ancestor materialized views; it is likely to take more time to execute. Some queries may need to refer as far back as to the root view (or base cuboid), which is always materialized and which has the lowest level of summarization with all possible dimensions and levels in those dimensions. In this case the query would likely to take an even longer time to execute. In our model, we have used two such query categories depending on their relationships to the materialized views in the data warehouse.

View Materialization Policy

There are two view materialization policies possible. The first one is the *fixed view materialization policy* where the set of materialized views remains fixed once materialized. Such views are updated periodically to reflect the latest information changes in the data sources. The second policy is the *dynamic view materialization policy* which assumes that over a period of time business requirements change resulting in a change in the query demand pattern. Hence the set of materialized views are changed dynamically over time to suit to the changing needs of data warehouse users (Kalnis, 2002). In this work, we have focused on the *fixed view materialization policy*. However, from a simulation perspective, this is not a serious limitation since our query arrival pattern captures both types of queries i.e. queries that can directly be answered from one of the materialized views and queries that can be answered indirectly from one of the ancestor materialized views. The only limitation may be that we are not accounting for query processing time lost due to change in the number of materialized views.

Environmental Variables

Certain environmental variables have a profound effect on the performance of a data warehouse. One such environmental variable is equipment failure. There could be many causes of equipment failure such as power failure, server crash and failure of computer hardware such as hard-disk, RAM or router. In our model we have incorporated a provision for equipment failure. When equipment failure occurs data warehouse system fails and all the queries in the system are lost.

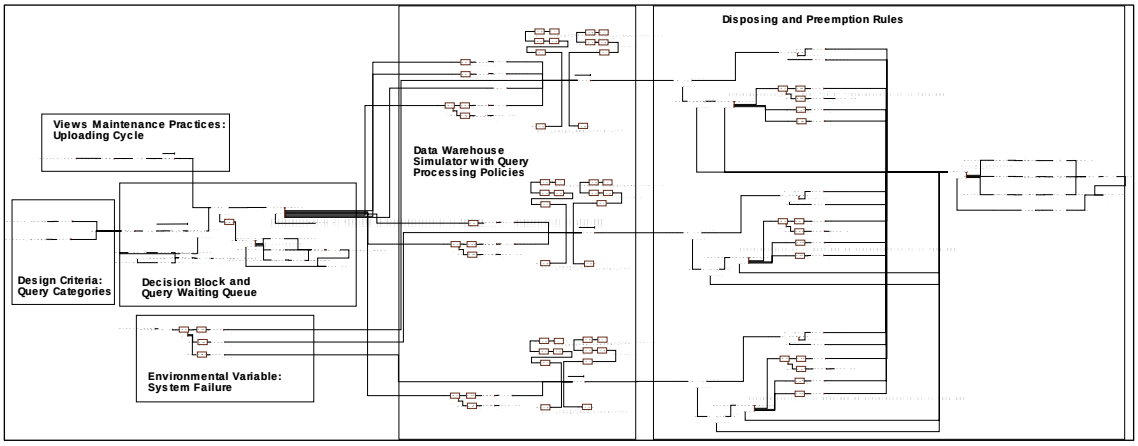
View Maintenance Practices

Once the set of views are materialized, they need to be maintained to reflect the latest changes in the source information. The data from data sources may be loaded into the data warehouse on a monthly, weekly or daily basis depending on individual organization's requirement. When the uploading cycle starts, all waiting queries are lost and no new queries are accepted until the uploading cycle is completed. In our model, we allow all queries already in process at the start of the uploading cycle to complete execution.

Query Processing Policies

If only one query is being processed at the data warehouse engine, it will be processed quickly since all the resources (CPU, RAM, etc.) are devoted to that single query. If more than one query is being processed simultaneously at the data warehouse engine, the processing of all queries will be delayed because of the sharing of resources. In our model, we have incorporated a provision to process up to three queries simultaneously. Time delays due to simultaneous queries execution are discussed in next section.

Figure 2. Implementation of Conceptual Framework of Simulation Model in ARENA.



In the next section, we present the implementation of our conceptual framework using Arena simulation package

IMPLEMENTATION OF CONCEPTUAL FRAMEWORK IN ARENA

The objective of the simulation approach is to predict the performance of a data warehouse under different design and operational conditions and query processing policies. We have considered several performance measures in our model such as overall average time to process a query, the average time it took to process a query by query category, the server utilization and the number of queries discarded, lost or not accepted either due to equipment failure or due to uploading cycle.

In order to gain some insight into the operation of a data warehouse, we populated a 1-GB TPCH Benchmark database from the Transaction Processing Council (www.tpc.org) website. The Transaction Processing Council website provides the data to populate very large experimental data warehouses ranging from 1 to 1000 GB and has been widely used by different industries for benchmarking purposes. By querying this data warehouse, we gained insights into some important query parameters relevant for building our simulation model. These parameters include average query processing time based on both materialized views and non-materialized views, delays in query processing times because of simultaneous processing and their interrelationships.

Figure 2 shows the main features of the simulation model developed using the ARENA simulation package. The major components of this model are discussed below:

Design Criteria Based on Materialized Views: Query Categories

This block generates queries that users pose to the data warehouse. There are two *create* modules. The first module generates entities (substitute for queries) that could be answered directly from the materialized views and the second one generates queries that could be answered indirectly from one of the ancestor materialized views. The inter-arrival time for queries is assumed to be exponentially distributed and the processing time is assumed to be normally distributed for both query types. The specific parameter values used are presented in Table 1. The type of distribution used for the inter-arrival time and service time was broadly justified from general queuing and database literature as well as our experience with the TCPH database.

Decision Block and Waiting Queue

In our simulation model, the server capacity is fixed to three i.e. it can process up to three queries simultaneously. This is modeled using three *process* modules in ARENA. Each incoming query first checks the availability of one of the three process modules before entering in the system. If the server is running full, then all

arriving queries wait in a queue. As soon as the server becomes available, queries in the queue may proceed for processing based on the rules discussed in the next section.

Data Warehouse Simulator with Query Processing Policies

The data warehouse simulator is the data warehouse engine that processes incoming queries based on their critical attributes. Modeling the processor of the data warehouse was challenging. While a typical computer can process several queries at a time, a processor construct in ARENA can process only one query at a time. We decided to allow at most three queries to be processed simultaneously. To achieve this, we created three process modules, and then activated them one at a time as the number of queries in the system increases from zero to three. Based upon our observations with the TPCH database, the processing time of queries increase as the number of queries being processed simultaneously increases. The following describes the rules used for query processing in the data warehouse simulator:

1. As the number of simultaneous queries to be processed increases, the processing time for all queries increases proportionately. This approximately mimics the simultaneous processing of multi-tasks in a real computer processor. Queries arriving while all three process modules are busy will be held in the queue until one of the process modules becomes free.
2. The number of queries being processed simultaneously determines the appropriate delay factor applied to the remaining processing time of all queries being processed. The delay factors used in our simulation are given in Table 1. The function of the delay factor is to linearly increase or decrease the remaining processing time as the number of queries being processed changes. For example, at simulation time T, if the system is processing two Type 1 queries simultaneously, (which implies that there are no waiting queries) with remaining processing times t_1 and t_2, and a Type 2 query arrives with a simulated processing time of t_3, then using the delay factors given in Table 1, the new processing time of the three queries will be given by (1.5/1.25)* t_1, (1.5/1.25)* t_2 and 2*t_3, respectively. Suppose at time T', processing of the Type 2 query is completed and no new query has arrived between T and T' (which implies that the waiting line is empty), the new processing time for the two Type 1 queries will be equal to (1.25/1.5)*t_1', and (1.25/1.5)*t_2' respectively. The factors given in Table 1 are now applied to the remaining simulated time to establish the adjusted remaining processing time. Since some of the original times had been changed when the number of simultaneous queries being processed increased from 2 to 3, it must now be adjusted suitably when it goes from 3 to 2. This is achieved by first preempting the current queries under process, changing the remaining processing time and resuming the processing by the respective servers from the point of preemption.

View Maintenance Practices: Uploading Cycle

The uploading cycle is modeled by using a single query that seizes the server for a certain period of time, thus simulating the time required for uploading the latest information from the data sources into the data warehouse. The uploading cycle query first checks if there are any queries that are being processed. If this is so, the uploading cycle query waits until such queries are executed. During this time all the waiting queries are lost. Once the queries in process are executed, the uploading query occupies the server. During this time, all incoming queries are also lost. As soon as uploading cycle is completed, the incoming queries will be allowed to enter the server.

Environmental Variable: System Failure

System failure is modeled using a single query that enters the server and seizes it for a specified period of time. When a system failure occurs, all the queries in the system as well as those waiting in the queue are lost. Incoming queries are also lost. If the data warehouse update (i.e., uploading cycle) is in process during the time of failure, it will fail and the data warehouse will revert back to its original state. As soon as the system is up again, the update cycle will resume first and then incoming queries will be entertained as per previously mentioned rules once the update cycle is done to completion.

Query Disposing Policies

A departing query leaving the system has to perform a check on the server processor before being disposed off. If there are no waiting queries in the queue, the remaining queries in the server processor will be speeded up as per earlier mentioned rules. If there is a query waiting in the queue, that query will occupy the processor vacated by the leaving query with added delayed processing time because of simultaneous processing based on earlier mentioned pre-emption rules.

EXPERIMENTAL DESIGN

The simulation experiments were designed to demonstrate the usefulness of a simulation modeling approach in a data warehouse design context. For each experimental setting, the model was run for a simulated period of eight days. Data for the first day was discarded to let the system reach steady state conditions. Table 1 shows various model input parameters with corresponding values.

In many service oriented simulation models, we hypothesize a Poisson arrival process, which leads to exponentially distributed inter-arrival times (Kelton et al., 2002). To reduce variance of the difference between outcomes (such as average waiting time, average turnaround time, etc.) of different experimental settings, we used common random number generators for various processes in the model. Processing times for the queries are assumed to come from the normal distribution with a mean of 10 and 20 minutes, and standard deviation of 2 and 3, respectively. As mentioned earlier, these were the estimates derived on the basis of our experiments performed on the TPCH database. Table 1 also shows the Delay Factors used for each type of query.

Table 1. Experimental Parameters.

Sr. No.	Query Type	Arrival Time Distribution (in minutes)	Processing Time Distribution (in minutes)	Delay Factors by number of queries being processed simultaneously		
				1	2	3
1	Query Type 1	EXPO(x)*	NORM(10,2)	1	1.25	1.5
2	Query Type 2	EXPO(y)*	NORM(20,3)	1	1.5	2
3	Failure	EXPO(7200)	EXPO(500)	–	–	–
4	Update	CONSTANT 1440	CONSTANT 60	–	–	–

* “x” and “y” are specified in Table 2

The time between failures is assumed to come from the exponential distribution with a mean value of 7200 minutes. The time to repair a failure is also assumed to belong to the exponential distribution with a mean of 500 minutes. Data uploading is assumed to take place daily starting at mid-night. This process is assumed to take 60 minutes.

Table 2. Experimental Settings.

Sr. No.	Experimental Setting	Inter Arrival Time (minutes)		Query Mix (y/x)
		Query Type 1 (x)	Query Type 2 (y)	
		[Proc = NORM(10,2)]	[Proc = NORM(20,3)]	
1	a	20	50	2.50
2	b	15	30	2.00
3	c	16	44	2.75
4	d	15	40	2.67
5	e	14	36	2.57
6	f	13	32	2.46
7	g	12	28	2.33
8	h	11	24	2.18
9	i	10	20	2.00

To achieve a variety of operating conditions, we kept the processing time parameters unchanged while the inter-arrival rate was varied. Table 2 shows the nine different experimental settings used in our experiment. For example, in setting "a," the mean inter-arrival time for Query Type 1 was set at 20 minutes and for Query Type 2, it was set at 50 minutes. Since Query Type 2 queries cannot be directly answered from any of the materialized views, they take longer to process and they also occur less frequently. With the query mix combination and mean arrival rate, we maintain a server utilization of 75% to almost 99%. The query mix is selected to be between 2.00 and 2.75.

In traditional queuing analysis, server utilization is given by λ/μ for single server systems and $(\lambda/s)*\mu$ for a multi server system with s servers where λ is the arrival rate and μ is the service rate (Kelton et al., 2002). The system in this simulation is a little tricky in that the number of active servers changes from 1 to 3 and the service time and hence the service rate of all servers changes as the number of active servers changes. As described earlier, this helps us mimic the processor of a data warehouse server more accurately. We calculated the approximate utilization rate by judiciously combining simulated probability of the system having 0, 1, 2, 3 or more queries in the system, types of queries in the system, and the corresponding rates at which each active server is working. For example, if there are two Type 1 queries in the system, and the experimental setting is "*b*," inter arrival time of Query Type 1 is Expo(15), Query Type 2 is Expo(30), and service time is Normal (10,2) for Query Type 1 and Normal(20,2) for Query Type 2. The delay factors will make the simulated remaining service time of the queries in process and/or the new arrival to go to 1.25*(remaining time from the original service time at the time of arrival of the second query). This will help us estimate the instantaneous service rate of the system when the system has two Type I queries.

There are many combinations of system states each of which will have its own instantaneous service rate as calculated above. Weighted average of these service rates will be used as an estimate for the average service rate. The average inter-arrival rate is known for setting "*b*". Since the service rate already incorporates the expected service time for each type of query and calculates its rate using that, the arrival rate used is simply the sum of the two arrival rates. Ratio of the arrival to service rates gives the estimated utilization. The system gets emptied everyday at 12.00 midnight to process the update query. The simulated system is thus a series of one day simulations. This midnight emptying process also allows us to slightly overload the system and still not experience

total clogging of the system, which would have happened in settings "*h*" and "*i*". In the next section, we briefly discuss our experimental results.

EXPERIMENTAL RESULTS

Table 3, Table 4 and Table 5 present the system performance obtained from our model for all the experimental settings and for each query type. For each experimental setting, these tables show the simulated server utilization, the average query waiting time, the average query processing time, the average query turnaround time and the 90th percentile of the query turnaround time. As expected, we observe that as the mean

Table 3. Query Type 1 Statistics.

Experimental Settings	Simulated Server Utilization	Query Type 1			
		Average Waiting Time	Average Processing Time	Average Turnaround time	90 percentile of Turnaround Time
a	0.75	2.55	12.48	15.03	23.45
b	0.89	7.67	13.59	21.26	39.35
c	0.84	3.93	12.87	16.80	26.91
d	0.85	5.98	13.17	19.15	36.17
e	0.89	6.10	13.43	19.54	35.32
f	0.91	13.39	13.86	27.24	59.84
g	0.91	17.90	14.28	32.17	57.18
h	0.91	42.83	14.52	57.34	147.70
i	0.99	90.87	14.71	105.60	188.10

Table 4. Query Type 2 Statistics.

Experimental Settings	Simulated Server Utilization	Query Type 2			
		Average Waiting Time	Average Processing Time	Average Turnaround time	90 percentile of Turnaround Time
a	0.75	2.77	31.18	33.95	47.96
b	0.89	8.69	34.58	43.27	70.99
c	0.84	5.10	32.78	37.88	59.67
d	0.85	6.01	34.22	40.24	60.13
e	0.89	6.95	34.24	41.19	62.69
f	0.91	13.33	34.40	47.73	85.91

g	0.91	17.94	36.26	54.20	85.92
h	0.91	40.14	38.35	78.49	146.30
i	0.99	93.69	39.70	133.40	221.30

Table 5. Overall Statistics.

Experimental Settings	Simulated Server Utilization	Overall			
		Average Waiting Time	Average Processing Time	Average Turnaround time	90 percentile of Turnaround Time
a	0.75	2.61	17.80	20.41	30.42
b	0.89	7.98	20.02	28.00	49.04
c	0.84	4.23	18.00	22.24	35.35
d	0.85	5.99	18.58	24.57	42.33
e	0.89	6.33	19.01	25.34	42.66
f	0.91	13.37	19.50	32.86	67.00
g	0.91	17.91	20.28	38.19	65.04
h	0.91	42.04	21.51	63.55	147.32
i	0.99	91.76	22.65	114.42	198.64

Figure 3. Bar Chart Showing Comparison of Average Turnaround Time between Query Type1, Query Type 2 and Overall.

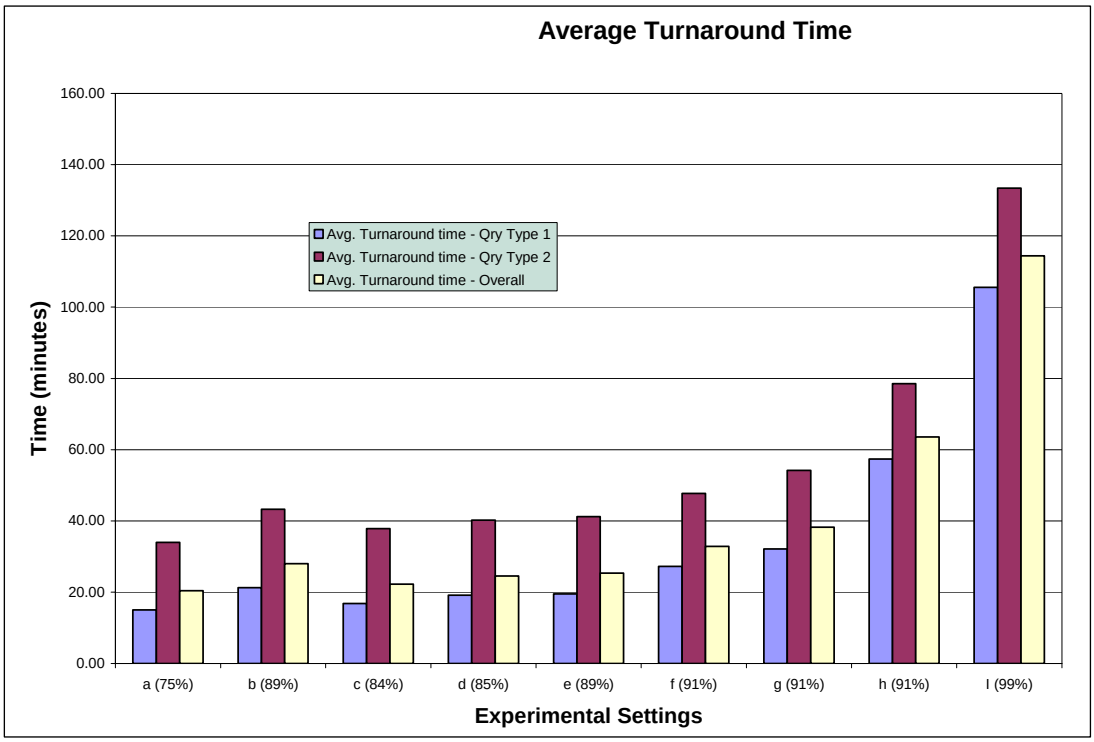


Figure 4. Bar Chart Showing Comparison of 90th Percentile Turnaround Time between Query Type 1, Query Type 2 and Overall.

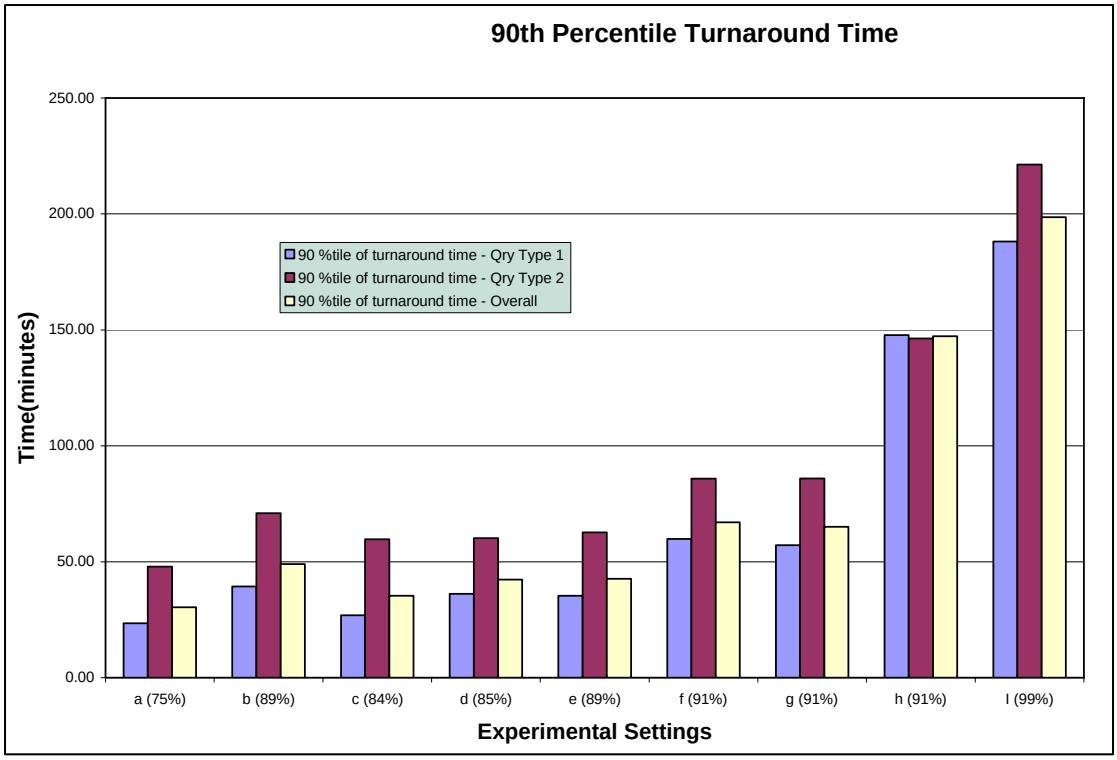


Table 6. Number of Queries Discarded.

Sr. No.	Experimental Settings	Queries Discarded due to Data Warehouse Update		Queries Discarded due to Data Warehouse Failure		Total Queries Discarded
		Query Type 1	Query Type 2	Query Type 1	Query Type 2	
1	a	33	7	4	1	45
2	b	36	19	3	3	61
3	c	37	10	5	0	52
4	d	37	10	6	2	55
5	e	44	9	0	5	58
6	f	52	12	3	4	71
7	g	66	27	4	3	100
8	h	86	36	11	3	136
9	i	173	78	32	13	296

inter-arrival time decreases in the subsequent experimental setting from "a" through "i," the simulated server utilization, the waiting time, the average turnaround time and the 90th percentile turnaround time increases. Mean

processing time increases modestly. This increase is a result of the inherent delay in simultaneous processing of queries as modeled here by the use of delay factors. As sever utilization increases, larger values of the delay factor are applied more frequently thereby increasing the time to process queries.

Figures 3 and 4 represent parts of these data as bar charts. They show the comparison between the average turnaround time and the 90^{th} percentile turnaround time respectively for Query Type 1, Query Type 2 and Overall (that includes both query types) for experimental settings "a" through "i" along with the corresponding simulated server utilization. One can see that as the value of inter-arrival time decreases, the average turnaround time and the 90^{th} percentile turnaround time increases. As one can notice from these figures, after experimental setting "f" the performance of system deteriorates considerably. A data warehouse manager could use such information to plan future upgrades of various system components to improve system performance levels.

Table 6 shows the number of queries discarded due to data warehouse update and data warehouse failure for Query Type 1, Query Type 2 and the total number of queries discarded. Again one can notice the increase in the number of queries discarded as the query inter-arrival time decreases. Since update takes place every day, this has a direct effect on the number of queries discarded during that time. But queries discarded due to data warehouse failure do not seem to be affected considerably. In the next section, we conclude this paper and suggest future research directions.

CONCLUSION AND FUTURE RESEARCH

Data warehouses are large complex systems with many interacting nonlinear components. A small change in any one component may produce dramatic changes somewhere else in the system. As such a data warehouse environment is very dynamic and sensitive and can exhibit periodic chaotic behavior. In this paper, we presented a conceptual framework for simulating the operation of data warehouses. We have also developed a simulation model that will predict the behavior and performance of a data warehouse system given initial values for some system parameters. We used the TPCH benchmark database to obtain insights into some critical parameters like query processing times, effect of simultaneous processing on query processing times, and their relationship to the materialized views.

A simulation model was built using the ARENA simulation package. In this simulation model, we provided for different query categories with different processing times, effect of equipment failure on the system, and uploading cycle to upload the current information from various data sources to make the model more realistic. We also provided for delaying and speeding up of query processing depending upon the number of simultaneous queries occupying the data warehouse processor.

Our experiments indicate that:

1. A simulation model can be used to examine the performance of a data warehouse system
2. Such a model can be used to estimate the impact of changes in operating policies on system performance
3. Simulation can also be used to estimate the impact of changing operation conditions such as query type mix or number of queries
4. Data warehouse managers can use the output from a simulation model to request for resources to ensure the system performance meets expectations

Future research could focus on validating the model against some real-world data warehouses. In addition, one can extend the simulation model discussed here to incorporate more features making the model more realistic and comprehensive.

REFERENCES

Badri, M. (2003). A Simulation Based DSS for Modeling Passenger Arrivals and Departures in an Immigration System of a Major International Airport. Proceeding of Summer Computer Simulation Conference, Montreal, Canada.

Chan J. (2004). Building Data Warehouses Using The Enterprise Modeling Framework. Journal of International Technology and Information Management, 13(2), 97-110.

Colby, L., Kawaguchi, A., Lieumen, D., Mumick, I. & Ross, K. (1997). Supporting Multiple View Maintenance Policies. Proceedings of ACM SIGMOD 1997, International Conference on Management of Data, 405-416.

Gleick, J. (1987). Chaos: Making a New Science. Sphere Books: London, UK.

Gupta, H. (1999). Selection and Maintenance of Views in a Data Warehouse. Ph.D. dissertation, Department of Computer Science, Stanford University.

Haag S., Cummings M., & McCubbrey D. (2005). Management information systems for the information age, 5e. McGraw-Hill Irwin.

Han, J. & Kamber, M. (2001). Data Mining – Concepts and Techniques. Morgan Kaufmann.

Hillard, R., Blecher, P. & O'Donnell, P. (1999). The Implications of Chaos Theory on the Management of a Data Warehouse. International Society for Decision Support Systems.

Kalnis, P. (2002). Static and Dynamic View Selection in Distributed Data Warehouse System. Ph.D. dissertation, Computer Science, The Hong Kong University of Science and Technology.

Kelton, W. D., Sadowski, R. P., & Sadowski, D. A. (2002). Simulation with Arena, McGraw-Hill publishing.

Kimball, R., Reeves, L., Ross, M. & Thornthwaite, W. (1998). The Data Warehouse Lifecycle Toolkit : Expert Methods for Designing, Developing, and Deploying Data Warehouses, Wiley publishing.

Kopel, M. (1997). Improving the performance of an economic system: Controlling chaos. Journal of Evolutionary Economics, 7, 269-289.

Medio, A. (1992). Chaotic Dynamics: Theory and Applications to Economics. Cambridge University Press, Cambridge, UK.

Monteiro, A.M.C. & Furtado, P.N. (2005). Data Skew-Handling in Parallel MDIM Data Warehouses. The IASTED International Conference on Databases and Applications.

Mumick, I., Quass, D. & Mumick, B. (1997). Maintenance of Data Cubes and Summary Tables in a Warehouse. Proceedings of ACM SIGMOD International Conference of Management of Data, Tucson, Arizona.

Schrader, M., Dakin, J., Hardy, K., Townsend, M., Whitmer, M. & O'Neil, B. (1997). Oracle Data Warehousing Unleashed, SAMS publishing.

Vasilakis, C., El-Darzi, E. & Chountas, P. (2004). A Data Warehouse Environment for Storing and Analyzing Simulation Output Data. Proceedings of the 2004 Winter Simulation Conference.