List of Publications


DEVELOPING REQUIREMENTS FOR OPERATIONAL BUSINESS INTELLIGENCE SYSTEM

A.D.N. Sarma  
Department of CSE,  
Acharya Nagarjuna University,  
Guntur- 522 510, A.P., India  
adnsarma@yahoo.com

Dr. R. Siva Rama Prasad  
Assistant Professor,  
Acharya Nagarjuna University,  
Guntur- 522 510, A.P., India  
raminenisivaram@yahoo.co.in

A.R.C. Sarma  
IP&BI Division BITS, PILAN,  
HYDERABAD Campus,  
A.P., India  
sarma@bits-hyderabad.ac.in

Abstract

The term Operational Business Intelligence is now a well-used phrase in the industry. The business intelligence system will act as a management tool for decision making system at strategic level in the organizations. The business intelligence that is obtained from data warehouse systems is slowly extending to tactical and even at operational level by reducing the action time between the occurrence of business event and action. The knowledge that is obtained from data warehouse can be brought back and collaborated with an operational system to gain operational business intelligence that helps smooth running of business operations on day to day basis. In this paper we present developing requirements for Operational Business Intelligence system that is based on core business services and business process of the business organization. Developing requirements of an operational business system are different from traditional Business Intelligence system. The proposed model has 5 levels of abstraction that covers business context, operational, functional and user, informational and knowledge requirements of an Operation Business Intelligence system.

Keywords  Operational Business Intelligence,  Requirement engineering, Business Intelligence, Data Warehouse and Business Context.

1. Introduction

In recent years, Operational Business Intelligence (Operational BI) is gaining popularity in the Business Intelligence (BI) market. Traditional BI systems are confined to strategic decisions making by the use of organizational historical data. The use of BI has been expanding its domain from strategic to tactical to operational level. The knowledge that is obtained from data warehouse can be brought back and collaborated with an operational system to gain operational business intelligence that helps smooth running of business operations on day to day basis. Thus Operational BI refers to the application of BI methods, tools and technology that helps to the vast number of low level decisions to be taken in the daily operations of a business.

Operational BI is called dynamic BI. Business is dynamic in nature and business runs on its own operations. The smallest operation in the business can be seen as a transaction and group of individual transactions will form as transactional data. The source for transactional data will most likely from two or more business applications in an organization. Early attempts at building Operational BI systems merely replicated operational data, without any integration into the larger data warehouse. This allowed the BI infrastructure to produce reports, off-loading the process from the operational systems. Modern business demands integration between operational data systems with organization historical data that results design of new business systems with and is known as Operational BI Systems. Developing requirements for Operational BI system is found importance in modern business organizations and is less focused in the literature. Requirements engineering is the first phase of any software development life cycle and is the basis for entire project.

The remainder of the paper is organized as follows. Section 2 reviews current and past research activities related work focusing mainly on developing requirements for business intelligence system. In section 3, functioning of typical Operational BI system is presented. In section 4, developing requirement for operational business intelligence system is presented. Section 5 covers results and discussion on the proposed requirements model. In section 6 contains conclusion and feature work.

2. Related work

Requirements are an important phase in software engineering methodologies [7]. In literature much of work attempted intern of requirement development
for data warehouse system and traditional BI and is rarely found about Operational BI system. Hence, the author has chosen his study in this area and the demand of Operational BI is rapidly increasing in business in almost all organizations. Several authors [7, 9] have addressed the need to improve DM-BI methodologies, but they focuses on DM-BI goals definition and DM-BI tasks specification as exploratory data analysis and develop tools for DM-BI process documentation, model-building, and pattern-finding.

The requirement development for BI and the BI portfolio [2] is described as three steps process. These three steps are as BI Opportunity Analysis, BI Business Case and BI Portfolio. The goal of BI Opportunity Analysis is to identify – in business terms for business executives, managers and analyst and the main ways that BI can be used to improve business results. The outcome of first step is a qualitative BI Business Case that describes three broad areas namely management processes, customer processes, and operating processes. BI opportunity analysis and the resulting business case are key inputs to the Business Improvement Opportunity (BIO).

The requirement elicitation process is addressed by most commonly used data mining (DM) methodologies [25]. DM methodologies state the necessity of business understanding as the starting point for any DM project. The CRISP-DM (cross industry standard for data mining) methodology [27] consists of four levels of abstraction, hierarchically organized from general tasks to specific cases. The process is divided into six phases, each one having many general tasks of second level or sub phases. General tasks are projected to specific ones, where the actions that must be developed for specific situations are described. As a consequence, we find a general task “cleaning data” in second level; then in third level, those tasks that must be developed for a specific case, as for example “cleaning numerical data” or “cleaning categorical data”. In the fourth level, groups of actions, decisions and results about the specific data mining project are collected.

The methodology P3TQ (Product, Place, Price, Time, and Quantity) consists of two parts [29]: [a] Modeling (PI): provides a step-by-step guide to develop and to build a model to address a business problem or opportunity. Modeling depends very much on the business circumstances that prompt the modeling in the first place, as indicated by the five different entry scenarios to PI. Largely, PI provides lists of actions that must be completed, depending on circumstances; and [b] Data Mining (PII): provides a step-by-step guide to mining the data to produce the required model as identified in PI. Data Mining consists of a series of stages that have to be completed in order. Unlike modeling in which several tasks may take place at the same time, mining has to proceed from stage to stage. Each part is based on four types of “activity boxes”; action boxes: indicate one or more required “next steps” for you to take; discovery boxes: provide exploratory actions that you need to take to decide what to do next; technique boxes: provide supplemental information about the recommended steps to be described in the action or discovery boxes; and example boxes: gives a detailed description of how to use a specific technique, along with pointers to an excel worksheet.

SEMMA (Sample, Explore, Modify, Model and Assess) is a methodology oriented to select, explore and model a great amount of data; looking to discover business patterns in the data [25]. The process begins with the extraction of sample data on which analysis is going to be applied. Once the sample is selected, the methodology proposes to explore the data in order to simplify the model. The third phase involves entailing data to DM tool. The fourth phase involves running the DM tool on the selected data. The last phase consists of evaluating results by analyzing the model by contrast with statistical models or new samples.

The author [1] defined requirements elicitation of DM-BI projects as five steps process. The elicitation process begins with understanding the project domain, project’s data domain, project scope, human resources identification with appropriate skill and correct DW-BI tools. In addition to this a set of templates defined to document requirements. Demand driven information requirement analysis are aimed to determine information requirements of data warehouse juxtapose to business process oriented approach as described [7] with four phases. The four phases are: Initialization, As is analysis, To be analysis and Modeling. A large number of approaches to information requirement analysis have been developed by academia, consultants.

3. Operational BI System
A typical Operational BI system is shown in figure 1.

![Fig.1: A typical Operational BI System](image-url)
The inputs for operational business process and analytics block are external data sources and data warehouse and the output is operational intelligence. External data sources are most commonly referred as operational systems that are known as online transaction processing (OLTP) systems. These are the systems that are used to run the day to day core business operations of the company. They are so called bread and butter system. Operational systems make the wheels of business turn. They support basic business processes of the company. These systems typically get the data into database. Each transaction processes information about a single entity as a single order, a single invoice, or a single customer.

Operational BI is an event centric approach where as Business Intelligence is data centric approach. Operational BI optimizes the business processes and activities in identifying and detecting events, situations that correspond to interruptions and bottlenecks.

Operational BI system helps to monitor business activities and identify and detect situations relating to inefficiencies, opportunities, and threats. Some definitions define Operational Intelligence an event-centric approach to delivering information that empowers people to make better decisions [1]. Operational BI helps quantify: the efficiency of the business activities how the IT infrastructure and unexpected events affect the business activities (resource bottlenecks, system failures, events external to the company, etc.) how the execution of the business activities contribute to revenue gains or losses. Operational BI puts important, real-time information into the hands of the people at the front line of business and the information consumers across the enterprise. These end users need to understand their business at a detailed level and make operational decisions that require real-time data that is part of an application. Operational BI helps business users can make more informed decisions on a day-to-day basis. The Operational Business Intelligence system is not only provide efficient tool for day to day decision making but also helps timely solution and this intern saves lost of operational cost.

4. **Operational BI system requirements**

The methodology of requirement development for Operational BI system different from traditional approaches of Data warehouse and Business Intelligence systems. Defining requirements is the process of determining what to make before making it. Requirements therefore define “the problem.” In contrast, “the solution” is defined by technical specifications. Developing requirements for Operational Business Intelligence system can be classified into five levels of abstraction. Figure 2 shows the logical flow of Operational BI System requirements.

![Fig.2 Operational BI requirements flowchart](image)

The following are the key business requirements of Operational BI System:

- Business context: Business services, stakeholders and mapping.
- Operational requirements: Operational parameters of the business processes, services and activities interms of key process areas for interesting measurements.
- Functional and user requirements
- Informational requirements
- Knowledge and other requirements.

4.1 **Business context requirements**

Business context requirements mean that knowing about business specific information of an organization interms of domain, core business services offerings, business processes associated with different business services that are important to the business to run, stakeholders and their mapping within and outside the business. A core business service is an offering that ensures the viability of the business, so that it can stay in business. At high level business context requirements cover the following:

- Business Services
- Stakeholder
- Business Services mapping

4.1.1 **Business Services**

In this the requirements of core business services of an organization will study in details including its sub-services if any. The interdependences between
sub services and other business services will be identified. The business process that is associated to each business service/sub-service is also examined. In addition to this the requirements of critical activities that are associated to each business process will be studied that must be performed to meet the organizational objective(s) while remaining solution independent.

4.1.2 Stakeholders

In any business, stakeholders will be the major actors of the business services operation and management for smooth running of business. In this all the stakeholders that are associated to the organization either directly or indirectly will be identified. The stakeholders of an organization will be classified as category. The category of stakeholder based on their business functions, services and process that are associated instead of expressing directly. The requirements of each category of stakeholders will be studied with reference to business context. Analyze the stakeholders; if two stakeholders share all of the same business services, then they should be combined. If however, they use one or more services differently than the other, then keep them separate.

4.1.3 Business services and stakeholders mapping

In this business services mapping with stakeholder category will be made. Next, link every stakeholder to the business with an interaction representing the business service they use, or the business service they provide to the business. Once you have done this for all stakeholders, and exhausted all business services, look at the stakeholders once again.

The outcome form business context requirements will be business context diagram that covers the list of all core business services of an organization, category of stakeholders and mapping them. Sometimes a single business context diagram can be drawn for small and mid sized organizations and where as separate context diagrams will be drawn for different divisions in the organization for the enterprise organization.

4.2 Operational Requirements

Operational requirements are the second level requirements of an Operational BI system. Well-written operational requirements of a system can be an effective vehicle. Operational requirements contain critical metrics that are to be measured for the performance of the business operations and processes. It covers critical measurements or facts and business dimensions along which the facts are normally analyzed. In addition to this covers system requirements that includes operating system, network and protocols of various operational systems. Figure 3 shows holistic view of overall Operational BI system requirements.

4.3 Functional and user requirements

The third level of Operational BI requirements addresses about functional and user requirements, which describe the tasks that user, must be able to perform using BI System. These are best captured in the form of use cases, which are stories or scenarios of typical interactions between the user and the system. User requirements are stated from the use’s point of view that describes what is needed for the user to do.

4.4 Information requirement

BI systems are an information systems and require study the information requirements of the organization and they are classified into the following four types viz data sources, data transformation, data storage and information delivery.

4.4.1 Data Sources

Operational systems are the major source of generating transactions. This operational data is an important to the business intelligence system that comprises data warehouse. The sources of information to Operational BI systems are from different operational systems, applications and legacy system. This includes all the details of the source data bases that exist with the organizations. You will be using the source system data in the data warehouse. You will collect the data from these source systems, merge and integrate it, transform the data appropriately, and populate the data warehouse. Typically, the requirements definition document should include the following information:
Available data sources such as database(s), flat files, XML files and information repositories such as WWW.
- Data volumes and data integrity
- Data structures within the data sources
- Location of the data sources
- Operating systems, network, protocols, and client architectures
- Data extraction procedures
- Availability of historical data

4.4.2 Data Transformation
After identification of all possible list data sources to the data warehouse system and then required to determine how the source data will have to be transformed appropriately into the type of data suitable to be store in the data warehouse. This requirement includes the details of transformation that involve mapping of source data to the data in the data warehouse. Indicate where the data about your metrics and business dimensions will come from. Describe the merging, conversion, and splitting that need to take place before moving the data into the data warehouse.

4.4.3 Data Storage
From your interviews with the users, you would have found out the level of detailed data you need to keep in the data warehouse. You will have an idea of the number of data marts you need for supporting the users. Also you will know the details of the metrics and the business dimensions. When you find out about the type of analyses the users will usually do, you can determine the types of aggregations that must be kept in the data warehouse. This will give you information about additional storage requirements.

Operational BI requirements definition document must include sufficient details about storage requirements. Prepare preliminary estimate on the amount of storage needed for detailed and summary data. Estimate how much historical and archived data to be in the data warehouse.

4.4.4 Information Delivery
The very objective of an operational BI system is to react faster to business needs and to anticipate business problems in advance before they become major issues. For this kind of system requires tighter integration between the BI system and operational system. A collaborative flat form is required to facilitate information delivered form both operational system and as well as BI system. So in this the information delivery requirements of the system will be studies. By the use of In-memory analytics can be used to implement low latency or zero latency system.

Information delivery requirements tell about how information will be delivered to the various types of users in the business and is also called information access requirements. Information delivery requirements must contain the following requirements on information delivery to the users:
- Drill down analysis
- Roll-up analysis
- Drill through analysis
- Slicing and dicing analysis
- Ad hoc reports

4.5 Knowledge and other requirements
In this level, the business domain knowledge requirements will be studied with reference to the business context of business organization. It is important to specify the knowledge to be mined, as this determines the data mining function to be performed. Kinds of knowledge requirements include concept description, association, classification, prediction and clustering. Also meta data, meta patterns or meta rules or meta queries, requirements and background knowledge of the business domain requirements will be studied.

5. Results and discussion
The proposed model for requirement development of an operational system is highly suitable and different from traditional data warehouse and business intelligence systems. The proposed model is based on business context of an organization that mainly focuses on domain of business, core business services, business process and events. This model is level wise hence different actors can be studied independently. The requirements collection of BI system will differ from one domain of business to other. This is addressed in our proposed model in business context requirements. In order to extend the knowledge that is acquired from data warehouse to operational system requires detail understanding on the operational system requirements and related metrics to be measured and these will be addressed in our proposed model under operational requirements. Finally the knowledge requirements cover domain specific knowledge, BI methods to be applied for extraction of knowledge and as well as meta-data requirements.

6. Conclusion and future work
This paper presents an approach for developing requirements for Operational BI system that address identified weakness in traditional data warehouse and BI requirement methodologies. The approach is based on business core services and business process
with reference to the business context of an organization. The proposed method for developing Operational BI system is level wise that covers business context, operational, functional and user, informational and knowledge and other requirements. As further work we will complete our approach for Operational BI system design and development for an operational system.

References


9 Key Features of Operational Business Intelligence

A.D.N.Sarma
Research Scholar
Dept. of Computer Science & Engineering
Acharya Nagarjuna University
Guntur- 522 510, Andhra Pradesh, India
adnsarma@yahoo.com

Dr.R.Siva Rama Prasad
Research Director
Dept. of International Business Studies
Acharya Nagarjuna University
Guntur- 522 510, Andhra Pradesh, India
raminenisivaram@yahoo.co.in

Abstract—Operational Business Intelligence (BI) is one of the fastest growing areas of BI domain. Operational BI provides information to the user in current time as opposed to traditional BI. In this paper, we present the nine key features of Operational BI system. The key features of the proposed system are low latency and reduced action time, access to lowest granularity data, real-time alerts notification, faster query response time, more ad hoc querying capability, support for Streaming SQL, configurable operational parameters and performance measurements, flexible to integrate the existing business processes and workflows and detailed timely information to the users. Explained how these key features are important to the proposed system. Mapping between the key features of the system with their equivalent functional modules is presented. Finally, Operational BI system is presented with the help of abstract layers. The functionality of each layer is mapped to set of functional modules that are identified from the key features of the system. A good Operational BI system not only saves money but also improves organizational efficiency that uses it well.

Keywords—action time, business intelligence, decision latency, low level decision making tool, operational business intelligence.

I. INTRODUCTION

Operational Business Intelligence (BI) is one of the fastest growing areas of BI [1]. The use of BI in the organizations has been growing more than a decade for strategic and tactical decision making. Business is dynamic that runs on daily operations. The term dynamic refers to decision making changes with respect to time and the term operations means managing day to day business events inorder to make business functional. Operational users or front-end users of the business require an efficient low level decision making tool to run business efficiently on daily basis.

The importance of low level decision making tools in the organizations has been increasing for the last few years. Operational BI will act as the right tool for the organizations that provides low level decision making to the users who can run business smoothly on daily basis. The aim of this paper is to present the key features of Operational BI system and explain how these features are important to the system. In addition to this we mapped each key feature of the proposed system to its equivalent functional modules.

Operational BI refers to the application of BI methods and technology to the vast number of low-level decisions to be taken in the daily operations of a business [2]. A good Operational BI system not only saves money but also improves organizational efficiency that uses it well.

The rest of the paper is organized as follows. Section 2 deals with relevant work. Section 3 describes traditional BI versus Operational BI system and how a typical Operational BI system looks like. Section 4 covers the key features of Operational BI system. Section 4 covers the holistic view of Operational BI system. Section 5 describes about conclusions and future work.

II. RELEVANT WORK

Many researchers carried work on BI systems and very limited work has been reported in the literature on Operational BI. Operational Data Stores (ODS) has identified [1] as one of the architectural entities in Operational BI system. An Operational BI environment requires the BI system to be tightly integrated with operational systems. As described [2] service-oriented approach enables BI to be more easily integrated into the operational environment.

The architecture of Operational BI as described [1] combines low data latency with excellent query response times. The difference between operational system and a data warehouse is illustrated [1] in three different aspects such as volatile; detailed, high granular data; and current valued. The term volatile means the change in data at much higher frequencies than contents of a data warehouse. Operational system provides detailed, highly granular data. The current valued in the sense that they contain no (or little) historical data.

The differences between strategic, tactical and operational BI as described [3] in terms of business focus, users, time frame and data for metrics. As mentioned [3] how the existing BI architecture can be altered, replaced, or rebuilt to ensure faster deliver and integration of BI. As presented [4], the conceptual architecture for BI has five layers such as data sources, ETL layer, data warehouse, business views (logical model layer) and BI front end analytic applications. As studied [5] the five different data warehouse architectures to find which data warehouse is successful. Operational BI is described [7] based on abstract levels of process frame work as analyze, monitor, facilitate and execute process.
III. OPERATIONAL BI SYSTEM

A. Traditional BI Versus Operational BI

Traditional BI systems have only two major functional blocks known as data warehouse and analytics engine. These are historic, static and data driven in nature. Figure 1 shows the traditional BI system. The direction of data transfer between data warehouse to analytics engine is only in one direction. The output of traditional BI system will help to the users of the organizations for strategic and tactical decision making.

Operational BI system is different from traditional BI system. Operational BI system is dynamic and event centric. In general, Operational BI system provides knowledge from the current data and it helps the user to take low level decision making to run business operations on daily basis. A typical Operational BI system is shown in figure 2. The inputs to the system are data sources and data warehouse. Data sources include both operational data and business process data.

Operational BI refers to the information a business uses in daily operations for effective and efficient use of business. Operational BI is defined as a type of software that performs both business process monitoring and as well as BI capabilities in near time/ real time basis.

B. Typical Operational BI System

Operational BI is a system of reports, metrics, and dashboards designed to drive decisions that optimize a company’s performance in the present. Typical Operational BI system is shown in figure 3. It consists of five abstract layers. Data sources are the first layer at the bottom of the system. It consists of multiple operational data sources and legacy applications.

Data integration layer provides integration between data sources, data warehouse and Operational BI engines. Integration between historic data sources and data warehouse use conventional ETL approach whereas as the operational data is integrated on near real time/ real time basis.

The data warehouse used by an enterprise BI system serves as a historical source for Operational BI. The Operational BI system is then able to put historical knowledge from the data warehouse into an event-based information delivery framework.

Portal acts as an information dissemination tool to the various users of the system. It acts as single entry point for all the users of the system such as managers, executives, analysts and operational team. The operational team consists of mostly front-end user of the business and business managers. The users can login into the system and access resources that are available in the system such as reports, dashboards and other
business applications and services as per their accesses privileges. Operational BI will make the organization more agile. The agility is not achieved by simply deploying real-time BI applications, but instead by building Operational BI applications that meet the right-time needs of business users.

IV. KEY FEATURES OF OPERATIONAL BI SYSTEM

In this section we present the key features of Operational BI system. Explain the functionality of each key feature of the proposed system. Identification of functional modules required to each key feature of the system.

The nine key features of the proposed Operational BI system can be enumerated as follows:

1. Low latency and reduced action time
2. Access to lowest granularity data
3. Real-time alerts or message notifications
4. Faster query response time
5. More ad-hoc querying capability
6. Support for Streaming SQL
7. Configurable operational parameters and performance measurements
8. Flexible to integrate the existing business processes and workflows
9. Detailed timely information to the users

A. Low Latency and Reduced Action Time

Operational BI system speed is critical. Operational BI system is known as Business Process Intelligence or Real Time Data Warehousing as described [3]. This focuses on providing real-time monitoring of business processes. The traditional BI system can be transformed into an Operational BI system by reducing action time. The action time is defined as the time interval between occurrence of business event and action. There are four different factors involved in action distance. The action time is the algebraic sum of data latency, analysis latency, decision latency and response latency. Action distance and time curve as shown in figure 4.

1. **Data latency** is the time from the occurrence of the business event until the data is stored and ready for analysis.
2. **Analysis latency** is the time from the point when data is available for analysis to the time when information is generated out. It includes the time to determine root causes of business situations.
3. **Decision latency** is the time it takes from the delivery of the information to selecting a strategy in order to change the business environment. This type of latency mostly depends on the time the decision makers need to decide on the most appropriate actions for a response to the business environment.
4. **Response latency** is the time needed to take an action based on the decision made and to monitor its outcome. This includes communicating the decision made as a command or suggestion, or executing a business action in a target system.

Reduction of action time in the system can be achieved by introducing real-time integration between operational data sources and Operational BI engines.

B. Access to Lowest Granularity Data

The users of Operational BI system can access detail data as lowest time interval as possible typically it will be in minutes or even in seconds. The granularity of the data depends on the type of services and business processes offered by an organization. The system could have an ability to measure the values of an attributes from the smallest level of transaction or event information in the case of business process or the lowest level of hierarchical data. The very purpose of deploying Operational BI system is to satisfy the business needs of the organization for low level decision making.

Business process and operational data sources have lots of data. The data derived from business processes and operational data sources is known as operational data. Money is hidden in operational data. So, there is a great need of detailed and timely information to the user to take low level decisions. Operational BI system will act as a right tool for low level decision making. A good Operational BI system not only saves money but also improves organizational efficiency that uses it well. Low granularity of data can be achieved by implementing drill down reports and operational OLAP reporting to the user of the system.

C. Real-time Alerts or Message Notifications

One of the objectives of Operational BI system is to react faster to business needs and to anticipate business problem in advance before they become major issues. Operational BI system speed is critical and users of the system want to know how the business is performing on day to day basis. The system has to comply with various Service level agreements (SLAs) and Key performance indicators (KPIs) of the business. this demands to measure configurable parameters on near real time/real time basis. At the same time the system can alert the user...
before it crosses the threshold limits of performance indicators. Hence, the system should have real time alert notification and monitoring as one of the key features.

Once the system can deliver right message to the right person in right time and then it helps the operational users of the business to take timely decisions that reduces decision making time. The reduction in decision making time results reduced action time. The system will generates alerts or message notification on near real time / real time basis if there is any change in threshold limits of configured operational parameters against dynamically computed parameters arrived from operational business process and operational data sources. This feature in the system can be implemented with the help of alert and monitoring functional modules that facilitates not only timely alerts to the users but also monitoring various operational parameter to measure business efficiency.

D. Faster Query Response Time

One of the key features of Operational BI system is how quickly it processes the given query and provides response to the users with necessary information. The query may be normal SQL or search on a specific key term. The system needs to be processed user queries as fast as possible so that the changes in the system can be traced. The assumption here is that query response time is smaller than the changes in the system behavior that is to be measured and is independent of data arrival rate.

Data compression functionality can be introduced with in the system that basically reduces the size of data stored in data base. The reduction of data size allows handling of larger sets of data in less amount of time. This intern allows faster query response. The advancements in hardware and software technology intern of multi-core, 64-bit processor, virtualization and row, column wise storage, main memory resident database algorithms, in-memory analytics leads to a new birth of BI into Operational BI. This feature can be obtained by implementing in-memory analytics and data compression functionalities and efficient data visualization techniques. In-memory analytics will perform by loading entire data into main memory. In-memory analytics the total data is resided in main memory of a computer. This supports faster query response time that allowing BI and analytic applications to support faster business decisions.

E. More Ad-hoc Querying Capability

An Ad-hoc query is a query that cannot be determined prior to the moment the query is issued. It is created in order to get information when need arises. It dynamically supplies the keyword, data source and the conditions without knowing the user to the system. It consists of dynamically constructed SQL by the user with the help of query tools or self service reporting system. This is in contrast to any query which is predefined and performed routinely. This functionality can be achieved by the help of in-memory analytics, operational OLAPs and data compression module. Operational OLAP refers extending OLAP features to operational data.

F. Streaming SQL Support

This feature is optional to the system. It depends on the rate at which operational data arrives to the system. The user wants to execute queries on fast moving data and then this feature is required. Mostly network monitoring, data streaming systems receive data continuously and need to run queries when data is in motion. Streaming SQL facilitates to run SQL queries on fast moving data obtained from operational system. It also provides ability to run a complex analysis and querying against a huge live data generated from operational systems.

G. Configurable Operational Parameters and Performance Measurements

The system can have a provision to configure operational parameters that are to be measured. Configuration of operational parameters is not only one time task but continuous. The system should have a support of even dynamic configuration of parameters. These parameters are stored in files and these files will reside in main memory of the computer system. These files can be updated on dynamically as and when there is change in parameter value that is computed in the system. The configured parameters are compared in a regular interval of time with dynamically computed parameters on continuous basis. Operational BI system has to provide up-to-date of performance against the key metrics that are to be measured with in the organizations. So it can help corrective action before it crosses the pre-defined threshold limits.

H. Flexible to Integrate the Existing Business Processes and Workflows

Operational BI is defined as measurement of business monitoring system. Every business will have at least one or more business processes. These business processes are to be integrated to Operational BI system inorder to measure various operational parameters. Business process can have sequence of event arranged in a proper order either serial or parallel that depends on the type process. This feature can be implemented in the system by introducing the functionality of business rules engine and work flow module.

I. Detailed Timely Information to the Users

Traditional BI systems are data centric and historic in nature. So they do not have a provision to have sufficient information to make the right decision to the operational business users who manages business operations on daily basis. There is a great need of tools to the organizations that helps low level decision making to run business on daily basis. Inorder to take low level decisions the system should have capability to produce detailed and timely information.

Business is dynamic in nature. Business processes and operational data sources can have lots of data. Money is hidden in operational data. So there is a great need of detailed and timely information to the user to take accurate and fast decisions. This requires the system should have BIand data analysis tools for slicking, dicing, filtering and probing. Inorder to present information to the user requires data visualization functionality in a simple, presentable and easily
understanding form. Hence, the system includes various types of reports such as normal reports, comparison charts, hierarchical chart and dashboards.

Operational BI allows the user to make decisions based on what is happening now rather than past. The system can access data from operational data sources on real time basis, generates and presents information to the users in real time. The detailed timely information to the users can include operational, performance, search and reporting information. This includes the following:

- **Operational information** — what are the services provided and how to help business users to gain better access on these services.
- **Performance information** — this provides how well services and business processes are being provided against benchmarks or agreed standards and targets,
- **Search information** — users will have a quick search facility to find required information based on certain key words, important terms.
- **Reporting information** — the user will be provided summary information in the form of reports, dashboards through multiple channels such as emails, smart phones and alerts in near real time/real time.

Portal provides a common interface to various users of the system and acts as a single entry point for both user requests and as well as system response. It acts as information dissemination tool. The users can login into the system and access various resources such as reports, dashboards and other business applications detailed timely information as per their accesses privileges assigned.

V. OPERATIONAL BI SYSTEM

In this section we present Operational BI system based on abstract layers. The functionality of each abstract layer is grouped into set of individual functional modules. In addition to this the key features of the proposed system are mapped with their equivalent functional modules.

A. Operational BI System

The proposed Operational BI system is shown in figure 5. It is based on the key features of the proposed system as described in the previous section. The system is broadly divided into six different abstract layers. These layers of the system are data sources, data services, Operational BI Engine, service deliver, deliver channel and user access layer. These layers of the system are data sources, data services, Operational BI Engine, service deliver, deliver channel and user access layer.

The functionality of each abstract layer can be achieved by combining individual functional modules as a group. These individual functional modules are derived from the salient features of the system as described in section 4. The various functional modules associated to each layer of the system are envisaged in figure 5. Metadata functional module will hold the data that describes and controls the system overall. Business services module include functionality of business rules, workflows, logging and monitoring. Operational BI engine includes core functionalities various engine of the proposed system. The users of the system include front line business users, business managers, IT users, finance users and even executives. The users will receive information from the system in the form of alerts (SMS) and emails. And also access the application through browser.

Figure 5. Operational Business Intelligence system
B. Mapping between Key Features of Operational BI and Functional Modules

The following table 1 shows the mapping between the key features of Operational BI system and its equivalent functional blocks.

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Features of Operational BI</th>
<th>Equivalent Functional Modules of Operational BI</th>
</tr>
</thead>
</table>
| 1.    | Low latency and reduced action time | • Real time integration of operational business process and operational data sources  
|       |                             | • In-memory analytics  
|       |                             | • Data compression  
|       |                             | • Alert notification to reduce decision latency  |
| 2.    | Access to lowest granularity data | • Drill down reporting  
|       |                             | • Operational OLAP reporting  |
| 3.    | Real-time alerts or message notifications | • Alerts and monitoring modules  |
| 4.    | Faster query response time | • In-memory computing  
|       |                             | • Data compression  
|       |                             | • Operational OLAPs  |
| 5.    | More ad hoc querying capability | • In-memory computing  
|       |                             | • Operational OLAPs  
|       |                             | • Data compression  
|       |                             | • Search  |
| 6.    | Support for Streaming SQL | • Streaming SQL  |
| 7.    | Configurable operational performance measurements | • KPIs/ SLAs module  
|       |                             | • Configuration of operational parameters  
|       |                             | • Dynamic computation of operational parameters  
|       |                             | • Monitoring  
|       |                             | • Dashboards  |
| 8.    | Flexible to integrate the existing business processes and workflows | • Workflow  
|       |                             | • Business rules  |
| 9.    | Detailed timely information to the users | • Analytics Engine  
|       |                             | • Alerts and monitoring  
|       |                             | • Logging and monitoring  
|       |                             | • Portal  
|       |                             | • User and security  
|       |                             | • Data visualization  |

VI. CONCLUSIONS AND FEATURE WORK

In this paper we presented the nine key features of Operational BI system. These are low latency and reduced action time, access to lowest granularity data, real-time alerts (or message notification), faster query response time, more ad hoc querying capability, support for Streaming SQL, configurable operational parameters and performance measurements, flexible to integrate the existing business processes and workflows and detailed timely information to the users. We presented typical traditional BI and Operational BI systems using simple diagrams. In addition to this we presented how a typical Operational BI system looks like and briefly explained. The nine key features of the proposed system are explained. In addition to this we defined how the key features of the system are achieved from their equivalent functional modules. Mapping between the key features of the system with their equivalent functional modules are presented in a tabular form. Finally, Operational BI system is presented and mapped all the identified functional modules of the system that are derived from the key features. Currently we are working on to realize a full functional architecture of Operational BI system. Operational BI system will act as a low level decision making tool to the organizations to run business on day to day basis. This is a new branch of BI and gaining immense popularity in these days. The system can be used in almost all the organizations including but not limited to HR, CRM, stock market, insurance, financial, medical, airlines, news, supply chain management, and telecommunications.

As further work, we are working on functional and business architectures, design and prototype implementation of Operational BI system. Another interesting direction of future work is to implement additional functional modules that are to be required to transform the existing BI system into Operational BI. One more interesting direction of future work is to develop software methodology for implement Operational BI system.

REFERENCES

[4] Liya Wu1, Gilad Barash1, Claudio Bartolini, “A Service-oriented Architecture for Business Intelligence”.


Abstract—Operational Business Intelligence (BI) is a new discipline of BI. Operational BI is gaining more and more popularity nowadays and finds the biggest growth in managing and optimizing business operations. Organizations have been gaining advantage of BI for their strategic and tactical decision making tool. Operational BI refers to application of BI methods and technology to the vast number of low level decision making. In this paper, we present enterprise architecture for Operational BI system. This includes layered, functional, technology, system and deployment architectures of the proposed system. The proposed enterprise architecture is based on the business functional modules of the system and enterprise architectural principles. The proposed enterprise architecture supports high availability, scalability and maintainability of the system.

Keywords- Business Intelligence; Deployment architecture, Enterprise Architecture; Layered Architecture, Operational Business Intelligence; System Architecture; Technology Architecture.

I. INTRODUCTION

Business Intelligence (BI) covers the needs of strategic and tactical decision making of business organization. Operational Business Intelligence (BI) is a new discipline of BI. Operational BI is gaining more and more popularity nowadays and finds the biggest growth in managing and optimizing business operations. BI covers the needs of strategic and tactical decision making of business organization and is accessible to limited number of users in the organization. Typical BI does not cover Operational BI of business organization for low level decision making. The front-end users or Line of business (LOB) users will be large in number and need to access the application. The importance of Operational BI increases day by day and finds the biggest growth in BI market. Operational BI is one of the fastest growing areas of BI [1].

There is a great need of low level decision making tools to the organizations to run daily business and this needs to be available to the all front-end business users of the system across the organization. Business is dynamic and runs on operations. In these days business is not confined to one location and is scattered across geography. Hence, there is a need of enterprise architecture for Operational BI system. Enterprise architecture is referred as architecture of the entire organization.

Operational BI refers to the application of BI methods and technology to the vast number of low-level decisions to be taken in daily operations of a business [2]. BI system acts as a management tool for decision making system at strategic level where as Operational BI system acts as a low level decision making tool in the organizations. The objective of the paper is to present enterprise architecture for Operational BI system that include layered, functional, technology, system and deployment architectures.

The rest of the paper is organized as follows. Section 2, we discuss relevant work. Section 3, describes the proposed enterprise architecture of Operational BI
system that includes layered, functional, technology, system and deployment architectures. The final section covers about discussion and conclusions of the proposed enterprise architecture of the system.

II. RELATED WORK

Many researchers [1], [2], [4], [8], [9], [13] reported work on BI architecture. There is no significant work has been reported on enterprise architecture of Operational BI. As described [15] enterprise principles and transformation methodologies are independent of the industry and organization. Enterprise system defined [15] as a framework for transformation that consists of a set of independent methodologies, tools and enterprise principles. The framework includes the five elements: Key principles of enterprise thinking, Enterprise transformation roadmap, Lean enterprise self assessment tools, Enterprise strategic analysis for transformation and enterprise architecting framework.

There are many enterprise architectural frameworks such as Zachman Enterprise Architecture Framework (ZIFA), The Open Group Architecture Framework (TOGAF), Extended Enterprise Architecture Framework (E2AF), Enterprise Architecture Planning (EAP), Federal Enterprise Architecture Framework (FEAF), Treasury Enterprise Architecture Framework (TEAF), Integrated Architecture Framework (IAF), Joint Technical Architecture (JTA) and many more available in literature. However, most architectural frameworks contain the following four basic domains as described [21].

- Business architecture: documentation that outlines the company’s most important business processes.
- Information architecture: identifies where important blocks of information, such as a customer record, are kept and how one typically accesses them.
- Application system architecture: a map of the relationships of software applications to one another; and
- The infrastructure technology architecture: a blueprint for the gamut of hardware, storage systems, and networks.

An operational BI environment requires the BI system to be tightly integrated with operational systems. As described [2] services-oriented approach enables BI to be more easily integrated into the operational environment. The differences between strategic, tactical and operational BI presented [3] in terms of business focus, users, time frame and data for metrics. It is mentioned [3] how the existing BI architecture can be altered, replaced, or rebuilt to ensure faster deliver and integration of BI into operational BI. Conceptual architecture for BI presented [4] and has five layers such as data sources, ETL layer, data warehouse, Business views (logical model layer) and BI front end analytic applications.

The five data warehouse architecture studied [6] such as Independent Data Marts Architecture, Data Mart Bus Architecture with Linked Dimensional Data Marts. Hub and Spoke Architecture (Corporate Information Factory) and Centralized Data Warehouse Architecture and Federated Architecture. Operational BI is described [7] based on abstract levels of process frame work as

- Analyze processes - obtain current view of activity in one or more applications to help workers make decisions and optimize processes.
- Monitor processes - monitor processes, alert users to exception conditions, and analyze data to determine root causes.
- Facilitate processes - embed metrics or reports within operational applications or portals.
- Execute processes - capture business events and apply rules to automate the execution of business processes.

An integrated view of metadata of data warehouse system discussed to support all users in the warehouse system. Proposed

III. OPERATIONAL BI ARCHITECTURES

In this section we present various architectures of Operational BI system such as layered, functional, system, technology and deployment architectures. Broadly speaking, architecture is the macroscopic design of the system. The proposed enterprise architecture of Operational BI system is not only based on the business functional modules of the system but also uses enterprise architectural principles. Operational BI system optimizes business operations on a daily basis. It provides reports, metrics, and dashboards to drive decisions that optimize business performance in the present. Thus Operational BI system functions on near real time. It has very low latency and reduced action time.

A. Layered Architecture

Figure 1 shows layered architecture view of a generic system.

![Figure 1. Generic layered architecture view of system](image1.png)

Modern enterprise applications are designed using several components connected to one another; each component provides a specific functionality. Components that perform similar functionality are grouped into layers. These layers are further organized as a stack. It contains four layers namely resource layer, data layer, business layer, presentation layer and user layer. The resource layer contains various data base resources to the system that includes operational data sources and legacy applications.

The layered architecture for Operational BI system is shown in figure 2. The layered architecture is one of most common way of representing enterprise architectures.

![Figure 2. Layered architecture of Operational BI system](image2.png)

As described [21] layered frameworks and models for enterprise architecture have proved useful because layering has the advantage of defining contained, non-overlapping partitions of the environment. The components in a higher layer use the services of components in a lower layer. A component in a given layer will generally use the functionality of other components in its own layer or the layers below it.

The major functional blocks of enterprise architecture are mapped to individual layers known as data sources, data layer, business layer, service layer, presentation layer and user layer. Layered architecture focuses on
the grouping of related functionality within an application into distinct layers that are stacked vertically on top of each other. Functionality within each layer is related by a common role or responsibility. These layers are loosely coupled to each other. The layered architecture provides reuse of the functionality, improve performance, scalability and maintainability of the system.

The presentation layer contains the components that implement and display the user interface and manage user interaction. This layer includes controls for user input and display, in addition to components that organize user interaction. It has two major types of components such as User Interface (UI) and Presentation logic components. User Interface components are the application’s visual elements used to display information to the user and accept user input. Presentation logic is the application code that defines the logical behavior and structure of the application in a way that is independent of any specific user interface implementation.

Business layer contains business data and business logic. This layer includes most business processing for the application is centralized as described [25]. It usually includes application facade and business logic components. Application façade provides a simplified interface to the business logic components. It reduces dependencies because external callers do not need to know details of the business components and the relationships between them. Business logic is defined as any application logic that is concerned with the retrieval, processing, transformation, and management of application data.

Data access layer contains data Access components. These components abstract the logic required to access the underlying data stores. Generally it is implementing using common data access logic in separate reusable helper or utility data access components. Service agents access data provided by an external service that are available to the system.

Services layer provides the service to be required to the system and includes OLAP, reports, dashboards, alert notification, monitoring and analytics. Access of the services is available to the user through presentation layer and this includes event messaging service. In this paper, we considered data warehouse is available as separate service to the proposed system.

B. Functional Architecture

In this section we present the functional architecture for Operational BI system. Broadly speaking, architecture is the macroscopic design of a software system. We define enterprise architecture as set of visible functional components and support functional components of a business system and the relationship between them. The term component means a functional block or blocks that performs one or more functional characteristics of the business system. According to [20] Functional Architecture (FA) is an architectural model which represents at a high level the software product's major functions from a usage perspective, and specifies the interactions of functions, internally between each other and externally with other products. The functional architecture of the proposed system is shown in figure 3.

![Figure 3. Functional architecture of Operational BI system](image)

The major functional modules of the system are: Data sources, Data services, Metadata services, Business services, Alert engine, Analytics engine, Security, Report engine and Portal. Each of these major functional blocks can have one or more sub-functional modules that have similar functionality.

According to IEEE Standard 1471-2000 [21], architecture is defined as “the fundamental organization of a system
embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution”.

Operational BI system accesses data from operational systems as it is generated and presents it in real time basis to the business users for their low level decision making on key business performances measurements. The data warehouse used by an enterprise BI system serves as a historical source for operational BI. The Operational BI system is put historical knowledge from the data warehouse into an event based information delivery from operational data sources.

C. Technology Architecture

Technology architecture provides three views: the conceptual, the logical, and the physical as described [24]. These views are used by architects to generate models within organizations that support and meet their operational requirements. A key requirement for organizations is the integration of these disparate technology architectures into one all encompassing architecture. The provision of this single, common architecture is vital to the creation of efficient, effective, and flexible organizations.

Conceptual view – This is used to map out areas of technology into a structure and framework. This is used to define, name, and position these areas for a common understanding between the IT supplier and the organizations using the technology.

Logical view – In this view the major functional elements that provide support for enterprise-scale operational requirements and their interrelationships are provided. Enterprise technology elements such as databases, mail systems, transaction support, and reliable messaging are provided in the logical view. The technologies that are provided at this level are normally packaged together as servers by enterprise software vendors.

Physical view - Each of the elements in the technology architecture requires mapping to elements of real technologies for both hardware and software. In this way, technology architectures are realized as complete systems of networks, servers, operating systems, and so on. Actual physical locations, server product names, and connectivity are shown at this level.

D. System Architecture

The system architecture of Operational BI is shown in figure 3.

![System architecture of Operational BI system](image)

Data resource layer contains all the legacy application data, data sources, data warehouse and Metadata. OBI engines with admin services, user management and other resources code is deployed on Application server.

Operational BI engines includes ETL, Real Time ETL, business rules, OLAP, data compression, reporting, dashboards, alert notification and monitoring, analytics and SQL streaming. In addition to this admin services includes managing various business services available to the user, user management and message templates. Web server contains various UI components and as well as portal. All the users of the system will access the resources through portal. Portal acts as singe entry point to access the applications resources available in the system to the users but also acts information dissemination.

E. Deployment Architecture

The deployment architecture of Operational BI is shown in figure 4. The
data store consists of one or more data sources that include operational data sources, historical data (data warehouse), Metadata and legacy applications data.

The proposed functional architecture of the system is decomposed into individual sub-systems. The major sub-systems of the proposed architecture are Data sources, Data services, Metadata services, Business services, Alert engine, Analytics engine, User and security, Report engine and Portal. The sub-systems are further decomposed into individual modules and sub-modules that have similar functional characteristics.

We presented technology architecture of the proposed system that consists of three views such as conceptual, logical and physical. A key requirement of technology architecture is to integrate disparate technology and applications in the organizations as a single view implementation to the users of the Operational BI system.

The system architecture of Operational BI system is presented that contains various Operational BI engines. This Operational BI engine code is deployed in Application server. Finally, deployment architecture of the proposed system is presented. We conclude that the proposed enterprise architecture for Operational BI is highly scalable, extendable and simple maintainable. The enterprise architecture meets the need of business users and provide timely information and as well as knowledge to run business operations on daily basis.

As further work, we are working to design and implement using Model View Controller (MVC) architecture as a prototype implementation. The prototype will be tested and implemented for one or more business domains.

REFERENCES

[4] Liya Wu1, Gilad Barash1, Claudio Bartolini, “A Service-oriented architecture for business intelligence.”


Abstract: Alert notification and monitoring system has become a common functional module in most of the business applications today. In this paper, we present the system architecture and functional architecture of alert notification and monitoring system. The proposed alert notification and monitoring system is one of the major sub-systems of Operational Business Intelligence (BI). The system will receive input messages from various functional blocks of operational business process and operational databases. The system will generate alerts on near real time basis if there is any change in threshold limits of configured operational parameters against dynamically computed parameters from the incoming messages and also facilitates to monitor the same. The proposed system has three major entities known as alert notification and monitoring engine, system services and support services. The various modules in alert notification and monitoring engine are message queue, message processing, alert notification, message delivery and monitoring. The functionality of each block of alert notification and monitoring system is explained.
Author Information

Paper title: Alert Notification and Monitoring for Operational BI System

First Author:
A.D.N.Sarma
Department of Computer Science & Engineering
Acharya Nagarjuna University
Guntur- 522 510, Andhra Pradesh, India
adnsarma@yahoo.com

Second Author:
Dr. R.Siva Rama Prasad
Department of International Business Studies
Acharya Nagarjuna University
Guntur- 522 510, Andhra Pradesh, India
raminenisivaram@yahoo.co.in
ALERT NOTIFICATION AND MONITORING
FOR OPERATIONAL BUSINESS INTELLIGENCE SYSTEM

Abstract—Alert notification and monitoring system has become a common functional module in most of the business applications today. In this paper, we present the system architecture and functional architecture of alert notification and monitoring system. The proposed alert notification and monitoring system is one of the major sub-systems of Operational Business Intelligence (BI). The system will receive input messages from various functional blocks of operational business process and operational databases. The system will generate alerts on near real time basis if there is any change in threshold limits of configured operational parameters against dynamically computed parameters from the incoming messages and also facilitates to monitor the same. The proposed system has three major entities known as alert notification and monitoring engine, system services and support services. The various modules in alert notification and monitoring engine are message queue, message processing, alert notification, message delivery and monitoring. The functionality of each block of alert notification and monitoring system is explained.

Keywords—alert notification system; monitoring system; messaging system; check algorithm; dynamic messaging; operational business intelligence.

1. INTRODUCTION

Alert notification and monitoring system has become a common functional module in most of the business applications today. Once an event is occurred in the system and then it generates an alert. The generated alert will contain a message that conveys brief description of the event. The messaging system will deliver the messages to the users in one more forms such as an email, SMS, MMS and voice alert. The proposed alert notification and monitoring system is considered as one of the sub-systems of Operational Business Intelligence (BI) system as described [8]. Alert notification and monitoring system is a combination of software and hardware. It delivers right message to right person in right time on near real time/real time basis. The aim of the paper

The rest of the paper is organized as follows. Section 2 deals with relevant work. Section 3 covers the proposed alert notification system. Explained how the proposed alert notification system works. Section 4 describes various algorithms of alert notification system. Section 5 covers about conclusion and future work.

2. RELATED WORK

Event notification has been studied and implemented widely [2,3,4,5,6]. As described [3] the design and implementation of a bridge that enables bi-directional messages transfer between applications based on CORBA Notification Service (CNS) and Java Message Service (JMS) architectures. This bridge provides a simple connection of both JMS and CNS systems that enables B2B messaging. The bridge is divided into three software layers namely front end, back end and converter. The front end layer performs initialization of configuration files and starting of the connections between CNS and JMS. The back end layer is responsible the connection between a concrete channel for message reception. Converter layer converts message from one service architecture to the other.

Federated event service modeling language (FESML) developed [4] that facilitate integration challenges of federated event services. It is developed using the Generic modeling environment (GME). The artifacts of FESML provide include event consumers, event suppliers, event channels, CORBA Gateways, UDP Sender, UDP Receiver and Multicast ports. FESML can be used to model a federation of events channels using CORBA Gateways.

As studied [5] the suitability of the Common Object Request Broker Architecture (CORBA) Event Services as a reliable message delivery mechanism. Implemented an application level scheme to build a reliable communication and is based on resynchronization of state. With this scheme there is no log/retry mechanism of the Event Service is used nor is the rebuilding the restarted objects. The failure of the Event Service is detected with the help of a daemon that will ping the Event Service periodically. If the daemon notices that the Event Service has died, it restarts the Event Service.

3. ALERT NOTIFICATION AND MONITORING SYSTEM

3.1 System Architecture

The term architecture is defined as the art or science of building of the system. System architecture stands an abstract description of the entities of a system and their inter connections and interactions.
The system architecture of the proposed alert notification and monitoring system is envisaged in figure 1. The inputs to the proposed system are messages that are generated from operational business processes and operational data sources. These incoming messages are put into message queue and these messages will be processed in serial.

The major modules in the proposed alert notification and monitoring system are message queue, message processing, alert notification, message delivery and monitoring. In addition to this scheduler, admin service and Quality of Service (QoS) modules are as a part of system services. Admin services help to control and manage the functionality. Scheduler will run as per the predefined time intervals to perform the specified activities. QoS will responsible for alert reliability and failure handling. Currently QoS is out of scope of this paper. In order to work the system requires few more resources such as configuration parameters (files), message templates (XML files) and database and these modules together known as support services. Broadly the users of the system can be classified into three major categories as admin, end user and executives. Admin user who manages the administration of the application by managing system services; end users of the system will receive alert notifications based on changes in the threshold levels of configured parameters. Executive users who monitor operational parameters that is to be measured on daily basis with the help of reports and dashboard.

The monitor unit will provide output to the executives to monitor reports such as line graphs, bar graphs, pie chart and dashboards. The message delivery unit will generate alerts to the subscription users in the form of SMS, Emails and alert to the individual application modules.

3.2 Functional Architecture

The functional architecture of alerts and monitoring system is shown in figure 2. The major functional blocks of the proposed system are message queue, message parsing, computation process, check algorithm and dynamic message formation, XSLT processor, dispatcher, monitor and data visualization. In addition to this there are few modules such as config parameters, KPIs/ SLAs, business rule modules and database. These modules are known as support resources to the proposed system.

Alert engine is one of the most important functional blocks of Operational BI system as described [8]. It generates alerts on real time basis from operational business processes and operational data. Alert engine receives input messages from various functional blocks of Operational BI system. The message formats are pre-defined XML templates and the structure of message template contains header and body. The typical message header contains attributes such as template id, source id, message id and the body contains contents of alert information. Alert engine parses the incoming messages, fetches data from the config files and validates the configured
parameters limits with computed aggregate values. The computed results will be stored in files and these files are updated dynamically. The computation results will be pushed to monitor module. The output of monitor is feed to data visualization module.

![Functional architecture of Alert notification and monitoring system](image)

Data visualization module provides for display of monitor data in the form of various reports and dashboards.

Alert engine selects the message and dynamically composes as per the operational parameters configured in KPIs/SLAs modules. The output of an alert engine is dynamically composed message (or alert) and it will send to Extensible Stylesheet Language Transformations (XSLT) processor.

XSL stands for EXtensible Stylesheet Language, and is a style sheet language for XML documents. XSLT converts the incoming messages into target user presentable form. In order to present multiple views of alerts to the target user separate XSLT style sheets are to be required. The output of XSLT will be given to the dispatcher.

Dispatcher can have a queue management internally. All the messages that are generated will be put in queue. Dispatcher will dispatch messages from queue based on priority of the alert. These alert messages will push these alerts or messages to a web based persistent communications channel and finally deliver to the right decision maker to notify the issue. The various types of alert notification types are Email, SMS, MMS, audio clip and video clip, programmatic and log file.

### 3.3 Check Algorithm

Check algorithm is one of the most important functional blocks of the proposed alerts notification and monitoring system. It uses associative search for comparison for threshold limits configured as per config file against the computed values. The comparison will be made as and when there is change in the values. This algorithm compares the values of pre-configured parameters with recently computed values that are stored in dynamic files.

The output of check algorithm module will be a message and will only be available if there is any limit crossing of the configured parameter values of config file, KPIs/SLAs module and business rule module that are to be measured against computed parameter values. The output of check algorithm module is feed to dynamic message formation and as well as monitor modules simultaneously. The pseudo code of Check algorithm is shown in figure 3.

1. READ data from config file
2. READ operational computed data
3. FOR EACH DO
4. Compares each config parameter value against operational computed data
5. IF (computed parameter > config parameters)
6. BEGIN
7. Generate Alers()
8. Feed data to Monitor()
9. END
10. END IF
11. END FOR
3.4 Input Message Template

In this section we present the structure and XML format of input message. The message formats are pre-defined XML template. Each message will have DTD that defines the schema of the message. The message has two parts that is header and body. The structure of input message is shown in figure 4.

<table>
<thead>
<tr>
<th>Header</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg_id: int</td>
<td>msg_size: int</td>
</tr>
<tr>
<td>module_id: String</td>
<td></td>
</tr>
<tr>
<td>templ_id: int</td>
<td></td>
</tr>
<tr>
<td>timestamp: long</td>
<td></td>
</tr>
<tr>
<td>msg_type: String</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4. Structure of input message

The typical message header contains elements that describe the properties of the message. The typical elements of the input message header are message identification (msg_id), module identification from which it was send (module_id), template identification (templ_id), at what input message generated (timestamp), message type (msg_type). The default type of msg_type is text.

The body of input message contains an attribute known as message size (msg_size). Figure 5 shows XML document structure of input message. The body of the message contains the actual data. The root element of input message is in_msg.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<in_message>
    <head>
        <msg_id= >
        <module_id= >
        <templ_id= >
        <timestamp= >
    </head>
    <body msg_size= >
    </body>
</in_message>
```

Figure 5. XML document structure of input message

3.5 Output Message Template

The structure of output message template is identical to input message template. The header section contains few more elements to describe message properties. The header contains various elements which consist of name/value pairs. The values may be of different types for different output messages. The structure of output message template is shown in figure 6.

<table>
<thead>
<tr>
<th>Header</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg_id: int</td>
<td>msg_size: int</td>
</tr>
<tr>
<td>templ_id: int</td>
<td></td>
</tr>
<tr>
<td>priority: String</td>
<td></td>
</tr>
<tr>
<td>type: String</td>
<td></td>
</tr>
<tr>
<td>msg_sync: Boolean</td>
<td></td>
</tr>
<tr>
<td>model: String</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6. Structure of output message

The various elements in the output message header are message identification (msg_id), template identification (temp_id), time at which the message is delivered from the system (timestamp), priority of the message e.g. low, medium and high (priority), type of the message e.g. SMS, Email (type), synchronization flag 0 indicates no sync and 1 indicates sync (msg_sync), and model: pull model / push model / hybrid model (model).

The body contains an attribute known as message size (msg_size). The body contains the actual message that is delivered to subscription user. Figure 5 shows XML document structure of output. The root element of input message is out_msg.

```
<out_message>
    <head>
        <msg_id= >
        <templ_id= >
        <timestamp= >
        <priority= >
        <msg_sync= >
        <model= >
    </head>
    <body msg_size= >
    </body>
</out_message>
```
3.6 Advantages

Alert notification engine has wide variety of applications in day to day life. Few applications are illustrated in various business verticals.

- Banking and finance: debit/credit card transaction details, check clearances, funds transfer alerts, check clearness, bill payment and fraud detection.
- Real-time information of stock market.
- News alerts
- Supply chain management: inventory level alerts, reorder levels of stock.
- In travel: ticket booking confirmation, ticket cancellation, travel schedule notification, time table notification and delays in operation.
- Environmental: Earth quakes, disasters, whether, rainfall, temperature, humidity and snow fall.
- Traffic: road clearances, accidents, traffic violation.
- Avionics: flight timings.
- Telecommunications:
- Enterprise: SLAs/ KPAs monitoring, sales,

4. CONCLUSIONS AND FEATURE WORK

In this paper we presented alert notification and monitoring system for Operational BI. Alert notification and monitoring is one of the important sub-systems of Operational BI as described [8]. We presented system architecture and functional architecture of alert notification and monitoring system. Each functional module of the proposed system is explained. The structure and formats of input and output message templates are presented. The XML document structure of input and output message templates are presented. The pseudo code of the Check algorithm is present. Explained how check algorithm generates alerts in the system to dynamic message module and as well as monitor module to view the reports. The proposed system works on near real-time/real time basis for alert notification and as well as monitoring the measurement of operational parameters of the business on day to day basis.

The proposed system can be developed as a generic system that facilitates to integrate any kind of business applications. The proposed monitor functionality can be further extended with suitable Application Programming Interface (APIs) to display real time monitoring of configured operational parameters versus computed parameters. The additional functionality such as reliability of alert notification can be added to the existing functionality to the proposed system with the help of QoS. Further, the work extends in the lines deliver of multiple channels support and various formats of alerts such as audio, video and multi media messages. The system can be further improved to interact the system by the user to drill down the information to make quick decision that reduces decision latency intern improves the action time.

REFERENCES

Abstract—Operational Business Intelligence (BI) is gaining immense popularity nowadays and finds the biggest growth in managing and optimizing day to day business operations. Organizations have leveraged the use of BI for their strategic, tactical use and proliferating into low level decision making for smooth running of business operations. In this paper we present the functional architecture of Operational BI system. The proposed architecture is based on its key features of the system. The key features of the proposed system are listed. The typical context diagram of Operational BI system is presented and explained. The proposed system is decomposed into various functional modules and sub-modules. The similar functional modules and sub-modules are grouped into sub-system that constitutes the full system. Finally, the functionality of various modules of the proposed system is described.

Index Terms—business intelligence, context diagram, functional architecture, operational business intelligence.

I. INTRODUCTION

Operational Business Intelligence (BI) is one of the fastest growing areas of BI [1]. Operational BI refers to the application of BI methods and technology to the vast number of low-level decisions to be taken in the daily operations of a business [2]. The importance of Operational BI increases day by day in all most all types of organizations. The BI system will act as a management tool for decision making at strategic level where as Operational BI system will extend the decision making at the low level of business operations. The objective of this paper is to present the functional architecture of Operational BI system based on its key features. The fully realized system inturn acts as a low level decision making tool to the business users of the organization to manage business operations on daily basis.

The rest of the paper is organized as follows. Section 2 deals with relevant work. Section 3 covers the key features of Operational BI system. Section 4 covers the typical context diagram of Operational BI system. Section 5 describes the proposed functional architecture of Operational BI System. The proposed architecture is decomposed into module and sub-modules based on its functionality. The functional modules of the proposed system are briefly explained. Section 6 describes how various functional modules fulfill the key features of the proposed functional architecture. The final section covers about conclusion and future work.

II. RELATED WORK

Lot of work has been reported on BI architecture and its implementations in the literature. Operational BI is an open area for research, design, development and implementation in the organizations for low level decision making. Operational Data Stores (ODS) has identified [1] as one of the architectural entities in Operational BI system. An Operational BI environment requires the BI system to be tightly integrated with operational systems. As described [2] service-oriented approach enables BI to be more easily integrated into the operational environment.

As proposed [2] enterprise data integration architecture consists of data integration applications, techniques, technologies, and services for providing a unified and consistent view of enterprise-wide business data. The right-time component of the architecture as described [2] collects actionable business events for analysis by operational BI applications. The three main techniques used for integrating data are data consolidation, data federation, and data...
propagation. Data integration applications may employ one or more of these techniques. Data integration technologies such as Enterprise application integration (EAI), Extract transformation load (ETL), Enterprise information integration (EII), Enterprise data replication (EDR), Right time ETL (RT-ETL), Enterprise content management (ECM) and Web services. Data integration management services include data quality management, metadata management and system management. The differences between strategic, tactical and operational BI as described [3] in terms of business focus, users, time frame and data for metrics. It is mentioned how the existing BI architecture can be altered, replaced, or rebuilt to ensure faster deliver and integration of BI [3]. As presented [4], the conceptual architecture for BI has five layers such as data sources, ETL layer, data warehouse, business views (logical model layer) and BI front end analytic applications.

As studied [6] data warehouse architectures to find which data warehouse is successful. These architectures namely Independent Data Marts Architecture, Data Mart Bus Architecture with Linked Dimensional Data Marts, Hub and Spoke Architecture (Corporate Information Factory) and Centralized Data Warehouse Architecture and Federated Architectures. Operational BI is described [7] based on abstract levels of process framework as

- Analyze processes - obtain current view of activity in one or more applications to help workers to make decisions and optimize processes.
- Monitor processes - monitor processes, alert users to exceptional conditions, and analyze data to determine root causes.
- Facilitate processes - embedded metrics or reports within operational applications or portals.
- Execute processes - capture business events and apply rules to automate the execution of business processes.


III. FEATURES OF OPERATIONAL BI

Operational BI system is different from traditional BI system. Operational BI system is dynamic and event centric where as traditional BI system is static, data centric and historic in nature. The architecture of Operational BI as described [1] combines low data latency with excellent query response times. The difference between operational system and a data warehouse is illustrated [1] in three different aspects such as volatile; detailed, high granular data; and current valued. The term volatile means the change in data at much higher frequencies than contents of a data warehouse.

Operational system provides detailed, highly granular data. The current valued in the sense that they contain no (or little) historical data. The knowledge obtained from Operational BI system is collaborated with the knowledge already gained from data warehouse system to the users for effective and efficient use of operational decision to run business on daily basis. As described [1], [17], [19] the key features as of the proposed Operational BI system can be enumerated as follows:

- Low latency and reduced action time
- Access to lowest granularity data
- Real-time alerts or message notifications
- Faster query response time
- More ad hoc querying capability
- Support for Streaming SQL
- Configurable operational parameters and performance measurements
- Flexible to integrate the existing business processes and workflows
- Detailed timely information to the users

IV. CONTEXT DIAGRAM OF OPERATIONAL BI

The very objective of Operational BI system is to react faster to business needs and to anticipate business problems in advance before they become major issues. The context diagram of typical Operational BI system is shown in figure 1. It consists of data sources at the bottom layer. Data integration layer provides integration between data sources, data warehouse and Operational BI engines. Integration between historic data sources and data warehouse use conventional ETL approach where as the operational data is integrated on near real time/ real time basis.
V. FUNCTIONAL ARCHITECTURE OF OPERATIONAL BI

In this section, we present the proposed functional architecture of Operational BI system. It is based on the key features of the system as envisaged in section 3. The functional architecture of Operational BI system is shown in figure 2. The major sub-systems of the proposed functional architecture are data sources, data services, analytics engine, business services, alert engine, reporting engine and portal.

![Fig. 2. Functional architecture of Operational BI system](image)

The proposed functional architecture is highly modular and scalable to enterprise level. The sub-systems are further divided into various major functional blocks based on its functionality.

A. Data sources

The bottom layer of Operational BI system has set of data sources as shown in figure 3. These data sources will acts as inputs to the Operational BI system. The data source layer contains one or more data sources that include ERP/CRM, text files, XML files, data files, legacy systems, emails, internet repositories and other applications. These data sources may be internal or external to the system. The data source may be either structured or unstructured data.

![Fig. 3. Data sources](image)

B. Data services

The major functional blocks in data services layer are shown in figure 4. Data integration is the process of the standardization of data definitions and data structures by using a common conceptual schema across a collection of data sources. Data integration provides suitable integration between data source to data services layer. Mostly, operational data sources are integrated on real time basis.

![Fig. 4. Data sources](image)

Data warehouse is a collection of data designed to support management decision making and contain a wide variety of data. It presents a coherent picture of business conditions at a single point in time. It holds all historic data of an organization.

Data storage module provides storage of operational data. Operational data is extracted from different data sources. The extracted data can be stored either in main memory or flash memory or even secondary storage memory. The advantage for storing the data in main memory is to achieve in-memory analytics for real time processing of data and response.

Ad-Hoc query module provides environment to run query that cannot be determined prior to the moment the query is issued. Ad-Hoc query is dynamically constructed SQL and is created in order to get information when need arises. This is in contrast to any query which is predefined and performed routinely. Inorder to run ad-hoc queries efficiently the system must require resources such as huge amount of main memory and very fast devices as temporary disk storage.

Streaming SQL module facilitates to run SQL queries on fast moving data obtained from operational system. Streaming SQL provides ability to run a complex analysis and querying against a huge live data generated from operational systems. Streaming SQL uses standard SQL, except that streaming SQL queries run forever, processing data as they arrive over specified time windows. The required functionality is obtained by implementing suitable streaming SQL algorithms.

OLAP stands for online analytical processing and is another name for multidimensional analysis. The basic operations of OLAP are Slice, Dice, Drill Down/Up, Roll-up and Pivot. Consolidation involves the aggregation of data that can be accumulated and computed in one or more dimensions. Drill-down is a technique that allows users to navigate through the details. The functionality of slicing includes take out a specific set of data of the cube and view whereas as dicing provides the slices from different viewpoints. Pivot provides rotation of data axes to provide an alternative presentation of the data.

Data compression module provides compression of OLAP cubes data using one or more compression techniques. The data compression can be implemented using one of the following techniques called B-Tree or Bitmap. The compression of data facilitates not only the reduction of data size but also faster data access, faster query response with large volumes of data.
C. Analytics engine

Analytics engine is an important tool in Operational BI system. The various functional blocks of analytics engine are shown in figure 5. The major functionality of analytics layer is to extract knowledge from the operational data and operational business process. This engine also works as per the predefined schedule to extract knowledge from historical data that is from data warehouse. The functionality of analytics on data warehouse is currently out of scope in this paper.

Analytics engine builds quantitative processes for a business to arrive at optimal decisions and to perform business knowledge discovery from historical data and as well as operational business process. This functionality includes various data mining algorithms, mining models, decision models and other analytic services. Other analytical services include statistical analysis, process mining, predictive analytics, predictive modeling, event processing and business process modeling. Data mining engine consists of a set of functional modules such as characterization, association and correlation analysis, classification, prediction and cluster analysis.

D. Metadata management

Metadata is often defined as "data about data". This is one of the important functional blocks in BI/Operational BI system. Metadata holds the data that describes and controls the system overall. The details of metadata include data definitions, data models, data mapping, tables -records, segments and entities, columns, keys, indexes, cubes and reports. In addition to this it controls handling of data and describes: rules, transformations, aggregations and mappings. It also describes the data services, control of operation, data warehousing, OLAP, analytics, operational system and reports. Metadata management includes the life cycle of activities covering collection, storing, querying, reporting and maintaining metadata repository for future use.

E. Business services

The business services layer covers the functionality of various modules such as workflow, KPIs/SLAs management, business rules definition, logging and monitoring. Figure 6 shows business services layer of the proposed system.

KPIs/SLAs module includes configuration of various key performance indicators and service level agreements of business operational measurements. The functionality of workflow includes configuration of business processes and execution. Business rules module facilitates configuration of business rule definitions and change of these configurable values from time to time. Logging modules facilitate to record various login activities of users. Monitoring module provides measurement of various operational parameters configured in the system against dynamic computed values obtained from operational data sources in the form of reports and dashboards.

F. Alert engine

Alert engine is one of the most important functional blocks of Operational BI system. It generates alerts on real time basis from operational business processes and operational data sources. Alert engine receives input messages from various functional blocks of the system. The message formats are predefined XML templates. The structure of message template contains header and body. The message header contains attributes like template id, source id, and message id. The body of the message contains contents of alert information.

Alert engine parses the incoming messages, fetches data from the configuration files and validates the configured parameters limits with computed aggregate values from incoming data. Alert engine selects the message and dynamically composes as per the operational parameters configured in KPIs/SLAs modules. The output of an alert engine is dynamically composed message (or alert) and it will send to the dispatcher. Dispatcher will push these alerts or messages to a web based persistent communications channel and finally deliver to the right decision maker to notify the issue.

G. Report engine

The report engine includes customizable reports which can present high-level findings as well as enable a user to drill down to find specific details. The major functional modules in reporting engine are shown in figure 7. This module includes standard report templates that provide the user to create customizable reports.

The reports module consist infrastructure for strategic reporting to serve the strategic management as well as operational reporting for low level decisions of business operations on day to day basis. Dashboard is a panel where all information is presented via graphical display in the simplest way possible. Specifically, it reports key organizational performance data on a near real time basis. There may be multiple reports on a single dashboard with this users can gain at-a-glance understanding of key trends and metrics. The users can manage dashboard reports in a user friendly environment.
Data visualization module provides visualization and analysis of real-time data. The functionality of this module includes detection, representation, and transformation of data. This module supports production, presentation, and dissemination of the results of an analysis to communicate information in the appropriate context to a variety of audiences as described [15]. Data visualization facilitates knowledge discovery through information synthesis, which is the integration of data based on their meaning rather than the original data type. Visual analytics is more than just visualization and can rather be seen as an integral approach to decision-making, combining visualization, human factors, and data analysis as described [20].

H. User and Security

All users of the system are defined and maintained in the database. User groups will be created based on broad category of functional usage and access on various sub-modules functionality. Users will be mapped to one or more user groups. Users are assigned to roles. Roles are mapped to user groups. So each user can access application modules based on their role. The functionality of this module includes user creation, group creation, assigning application functionality to the groups, role creation, assigning roles to groups, and activation and deactivation of users. User access on application functionality is controlled in a data driven manner that uses Role based access control. A two factor authentication will be implemented for application login inorder to provide better security to the application.

I. Portal

Portal provides a common interface to various users of the system and acts as a single entry point for incoming requests. It acts as an information dissemination tool. The users can login and access resources such as reports, dashboards, and other business applications and services as per their access privileges.

VI. DISCUSSION

Early days most of the organizations find the use of BI for their strategic decision making and later BI services are extended into tactical level. Business is dynamic in nature that runs on operations. It mainly requires BI for daily operations for effective and efficient business operations. Nowadays organizations require tools for low level decision making for smooth running of the business operations on daily basis. The number of decisions to be taken at operational level is very large as compared to tactical and strategic levels and it varies based on the size of organization. Collectively the value of low level decisions over a period will be large. Therefore, there is a great need of operational decision making tools for daily operations of business.

The full functional architecture of Operational BI is shown in figure 8. The proposed system envisages all individual sub-systems that are described in the previous sections. It is based on the key features of the system as described [19]. The proposed system in this paper covers all required functionality to the users of the organization for low level decision making for smooth running of business operations on day to day basis. Real time data integration with operational data will reduce data latency of action time. The other modules identified in data services layer such as data compression and data storage in main memory (in-memory) and adhoc querying will provide faster query response and ad hoc query capabilities. In-Memory analytics will provide reduced analysis latency.

Streaming SQL provides analysis of data in motion. OLAP cubes and aggregates will provide lowest granularity data access. The measurement of operational parameters can be obtained from KPIs/SLAs functional module.

Logging and monitoring modules will facilitate not only login activities of various events but also monitoring the same. The notification to the business users can be provided by the use of alert engine functionality. It works on near real time/real time basis. The alert engine provides the right information to the right user in right time which reduces decision latency time. The report engine will provide ready to use industry specific reports to the business users and these can be managed on self service basis. Analytics engine will provide all required analytics for operational perspective. The portal provides collaboration between OLTP and OLAP systems. It also acts as information dissemination.

The various users of the system are managers, analysts, IT users, Line of business (LOB) managers, operational users, and financial managers. The proposed functional architecture of Operational BI system will suit almost all types of business domains. The proposed system is based on highly modular structure. Additional functionality can be introduced to the proposed system without affecting the existing functionality. However, new report templates, dashboards, message templates can be added to the existing system that helps to the user not only to operate on self services basis but also to support multiple industry verticals.

VII. CONCLUSION AND FEATURE DIRECTION

Operational BI system and its typical context diagram. In addition to this we proposed the functional architecture of Operational BI system that is based on the key features of the system [19]. The proposed functional architecture of Operational BI system is divided into various sub-systems that are represented as abstract layers. The functionality of each abstract layer is decomposed further into modules and sub-modules. The functionality of modules is explained briefly.

As further work, we are working on technical architecture and design of major functional modules of the proposed system as a prototype implementation. The same will be further extended into enterprise architecture by integrating these individual functional modules to build a full fledged Operational BI system. Furthermore, we will test and implement the designed Operational BI system for one or more business domains and will extend the same as a generic product.
There are few open areas in the proposed functional architecture of Operational BI system for further development as identification and standardization of data exchange formats between various modules. Identification of message templates for alert notification that is specific to industry verticals. Another interesting direction of future work is implementation of industry specific reports and dashboards that operate on self service basis to various users of the system.

REFERENCES


Design of Operational Business Intelligence System
Using Model View Controller-Model 2 Architecture

A.D.N.Sarma  
Department of Computer Science  
Acharya Nagarjuna University  
Guntur- 522 510, Andhra Pradesh, India  
adnsarma@yahooo.com

Dr. R.Siva Rama Prasad  
Department of International Business Studies  
Acharya Nagarjuna University  
Guntur- 522 510, Andhra Pradesh, India  
raminisivaram@yahoo.co.in

Abstract—Operational Business Intelligence (Operational BI) is a new domain of Business Intelligence (BI). This acts as a low level decision making tool for front line managers of the business for day to day business operations. It works on near real time to provide alerts to the users from operational data sources. In this paper we present the design of Operational BI system using Model View Controller (MVC) Model 2 architecture of Java 2 Platform, Enterprise Edition (J2EE). The tiered architecture of the Operational BI system is presented and explained the function of various components present in each tier. Working of MVC - Model 2 architecture is explained in the context of proposed system. Generic object flow of the proposed system is presented. Finally system architecture of the proposed system is envisaged.

Keywords- Business Intelligence, J2EE, Low level Decision Making tool, Model View Controller, Operational Business Intelligence.

I. INTRODUCTION

Operational Business Intelligence (Operational BI) is a new domain of Business Intelligence (BI). This acts as a low level decision making tool for front line managers of business for day to day business operations. It works on near real time basis. Operational BI is an event based, work on near real time/ real time and low data latency where as traditional BI is data driven, high data latency and historic in nature. Traditional BI is mainly used for strategic and as well as tactical decision making tool whereas Operational BI is mainly useful for low level decision making tool. In this paper we present the design of Operational BI system using Model View Controller (MVC) Model 2 architecture of Java 2 Platform, Enterprise Edition (J2EE).

The aim of the paper is to present design of Operational BI system using MVC Model 2 architecture that supports low data latency and reduced action time and is browser based. The proposed design of Operational BI system does not discus about data warehouse system. The rest of the paper is organized as follows. Section 2, we discusses about relevant work. Section 3 covers the design of Operational BI system using MVC architecture. Section 4 discusses the proposed MCV architecture and its advantages. Section 5 covers conclusion and further work.

II. RELEVANT WORK

Many researchers [3], [4], [5] reported work on architecture of BI system. There is very limited work has been attempted [6], [7], [10], [13] on Operational BI system. MVC is an architectural design pattern. It is well proven, standard and industry accepted architecture. Many enterprise applications use the design principles of MVC architecture as described [1], [12], [16], [17], [18]. The very advantage of this architecture is separations of layers and supports multi-layer architecture. This provides change in one layer can be accomplished without altering the other layers of the system.

As described [3] Service Oriented Architecture (SOA) for BI makes possible a seamless integration of technologies into a coherent BI environment, thus enabling simplified data delivery and low-latency analytics. It is also mentioned that SOA based approach appears to be the best way to reduce the total development and maintenance cost, and to minimize the risk and impact across an entire enterprise when introducing business intelligence solutions. As described [4] the architecture of process driven business intelligence decision system contains four major constructs such as the data construct, the operation construct, the information construct and knowledge construct. The data construct includes external and internal data sources and process audit log databases. The operational construct includes operational data in Operational Data Store (ODS) and process data. The information construct includes data warehouse and process warehouse. The knowledge construct includes real time business intelligence.

A low cost BI system proposed [5] using self organized multi agent technology. As described [7] a layered methodology for designing ETL process in operational business intelligence systems for modeling the operational business processes for generating the end to end information views required by operational decision making. As described [10] the functional architecture of the Operational BI system that consists of six abstract layers namely data sources, data services, analytics engine, business services, alert engine, reporting engine and portal. Enterprise architecture of Operational BI system is provided [10] that support high availability and scalability. In addition to this various architectures that include functional, layered, technology and deployment architectures of the proposed
system. In [15] the nine key features of the proposed system are presented, mapping between the key features of the system in terms of its functional modules presented. In addition to this the holistic view of Operational BI system is presented.

III. DESIGN OF OPERATIONAL BI SYSTEM

In this section, we present tiered architecture, MVC Model 2, sequence diagram, structure of web application, and system architecture of the proposed system. The designing of Operational BI system is different from traditional BI system. As mentioned [13] Traditional BI system are static, data driven, high data latency as compared to dynamic, event driven and low data latency. MVC architecture is a well proven and industry accepted architecture. This architecture is highly scalable, extendable, secured and support for even complex systems that includes like Operational BI system.  MVC design pattern not only supports event centric but also web based applications.

A. Tiered architecture

The tiered architecture of Operational BI system is shown in figure 1. The proposed Operational BI is considered as stack of tiers. A tier is a logical representation of concerns in the system. Each tier is assigned its unique responsibility in the system. Each tier is logically separated from another and is loosely coupled with the adjacent tier.

<table>
<thead>
<tr>
<th>Client Tier</th>
<th>Application client, Device and GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation Tier</td>
<td>Java Server Pages, Servlets and UI elements</td>
</tr>
<tr>
<td>Business Tier</td>
<td>Workflow and Business Objects</td>
</tr>
<tr>
<td>Integration Tier</td>
<td>JMS, JDBC, SOA, Connectors and Legacy</td>
</tr>
<tr>
<td>Resource Tier</td>
<td>Databases, external systems, and legacy applications</td>
</tr>
</tbody>
</table>

Figure 1. Tiered architecture of Operational BI System.

Client tier represents all devices or system clients accessing the system resources. A client can be a Web browser, a Java application, or a device and Graphical User Interface (GUI).

Presentation tier encapsulates all presentation logic required to service the clients that access the system. The presentation tier intercepts the client’s requests, controls access to business services, construct the responses and finally deliver the response to the clients. The presentation layer contains Servlets and JSPs that produce User Interface (UI) elements.

Business tier provides the business services required by the client. This tier contains the business data and business logic. Typically, most business processing for the application is centralized in this tier as described [14]. Mostly workflow, Java bean components or Enterprise beans are used as business objects. As described [9] Java bean components are Java classes that can be easily reused and composed together into applications. JSP technology directly supports using JavaBeans components with JSP language elements. You can easily create and initialize beans and get and set the values of their properties. The various business services of Operational BI includes login, logging, master data management, analytical, Metadata, Key Performance Indicators (KPIs), business rule engine, alert engine, reports and dash boards as mentioned in [10].

Integration tier The components in integration tier can be JDBC, JMS, SOA, J2EE connectors, or middleware or any legacy application. Resource tier contains the business data and external data or application sources, legacy systems and data warehouse system. The design of data warehouse system is out of scope of this paper and considered as an external system to the proposed architecture. Integration tier is responsible for communicating with external resources and systems. The business tier is coupled with the integration tier whenever the business objects require data or services that reside in the resource tier.

Resource tier contains database servers, legacy systems, other transaction application and ftp servers, and directory servers.

B. Model View Controller(MVC) Model 2 Architecture

The MVC architecture has its roots in Smalltalk, where it was originally applied to map the traditional input, processing, and output tasks to the graphical user interface model as mentioned [12]. MVC is one the architectural design patterns. A typical MVC architecture is shown in figure 2.

The client will access the application through browser. All user requests are handled via controller and responses through JSP. Internally, model (Java bean) will connect to
the enterprise servers/ databases inorder to access the required data.

Model encapsulates the core data and functionality. The model represents enterprise data and the business rules that govern access to and updates of this data.

View encapsulates the presentation of the data. The view renders the contents of a model. It accesses enterprise data through the model and specifies how that data should be presented. It is the view's responsibility to maintain consistency in its presentation when the model changes. This can be achieved either by using a push model or pull model. In push model the view registers itself with the model for change notifications whereas in pull model the view is responsible for calling the model when it needs to retrieve the most current data.

Controller accepts inputs from the user and makes request from the model for the data to provide a new view. The controller translates interactions with the view into actions to be performed by the model. In a stand-alone GUI client, user interactions could be button clicks or menu selections whereas in a Web application, they appear as GET and POST HTTP requests. The actions performed by the model include activating business processes or changing the state of the model. Based on the user interactions and the outcome of the model actions, the controller responds by selecting an appropriate view.

The request and response flow of the system is envisaged as follows:
1. User requests are directed to the controller servlet.
2. The controller servlet accesses required data and builds the model, possibly delegating the processing to helper classes.
3. The controller servlet (or the appropriate subordinate task) selects and passes control to the appropriate JSP responsible for presenting the view.
4. The view page is presented to the requesting user.
5. The user interacts with the controller servlet (via the view) to enter/modify data, traverse through results.

C. Sequence Diagram

The sequence diagram of MVC with request dispatcher (or manager) is shown in figure 3. It shows the interaction between various objects such as user, request manager, controller, JSP and bean.

D. Structure of Web Application

The generic object flow of design of the proposed system is envisaged in figure 4. Java Server Pages Model 2 is Sun's attempt to wrap JSP within the MVC paradigm [12].
The general structure of a Web application using the Java Server Page Model 2 architecture has front controller, data access and application logic, Service-To-Worker and Dispatcher View, Intercepting Filter, Value List Handler and Data Access Objects (DAOs).

**Front Controller** is a Servlet that acts as the centralised entry point into a Web application, managing request processing, performing authentication and authorization services, and ultimately selecting the appropriate view.

**Data access and application logic** contain entirely within the controller servlet and its helper classes. The controller servlet (or the helper class) should select the appropriate JSP page and transfer control to that page object based on the request parameters, state and session information. One of the major advances that come with JSP Model 2 is Sun’s specification of the Java Standard Tag Library (JSTL). It specifies the standard set of tags for iteration, conditional processing, database access and many other formatting functions. The guidelines associated with JSP Model 2, Sun also provided a set of blueprints for building application using the MVC paradigm and these blueprints renamed the J2EE Core Patterns.

**Service-To-Worker and Dispatcher View** strategies for MVC application where the front controller module defers processing to a dispatcher that is selected based on the request context. In the Dispatcher View pattern, the dispatcher performs static processing to select the ultimate presentation view. In the Service-To-Worker pattern, the dispatcher’s processing is more dynamic, translating logical task names into concrete task module references, and allowing tasks to perform complex processing that determines the ultimate presentation view.

**Intercepting Filter** allows for pluggable filters to be inserted in the “request pipeline” to perform pre and post processing of incoming requests and outgoing responses. These filters can perform common services required for all or most application tasks, including authentication and logging.

**Value List Handler** is a mechanism for caching results from database queries, presenting discrete subsets of those results, and providing iterative traversal through the sequence of subsets.

**Data Access Object (DAO)** is the centralised mechanism for abstracting and encapsulating access to complex data sources, including relational databases. The DAO acts as an adapter, allowing the external interface to remain constant even when the structure of the underlying data sources changes.

### E. System Architecture of Operational BI System

The system architecture of the proposed Operational BI is shown in figure 3. Data resource layer contains all the legacy application data, data sources, data warehouse and Metadata. OBI engines with admin services, user management and other resources code are deployed on Application server. Operational BI engines have various sub-engines that include ETL, Real Time ETL, business rules, OLAP, data compression, reporting, dashboards, alert notification and monitoring, analytics and SQL streaming as mentioned in [15]. In addition to this admin services include managing various business services, configuration of parameters to be measured, and user management.

![System architecture of Operational BI](image)

Web server contains various UI components and as well as portal. All the users of the system will access the resources through the browser.

The portal acts as singe entry point to the users to access the applications resources that are available in the system. This also acts information dissemination and collaboration tool.

### IV. DISCUSSION

MVC is an architectural design pattern. The goal of the MVC design pattern is to separate the application object (model) from the way it is represented to the user (view) from the way in which the user controls it (controller). This pattern contains two models MVC 1 and MVC 2 for implementation. MVC 1 architecture is page centric and tightly coupled between page and model.

The proposed design of Operational BI system uses MVC 2 model that removes page centric property of MVC 1 and separates presentation logic and application logic. The controller receives all requests for the application and is responsible for taking appropriate action for each request. It is well proven and industry accepted architecture. Model contains the state (data), view displays model to user (presentation) and controller modifies model (business logic). The proposed design of the system will support the 9 Key features of the Operational BI as described in [15].

The MVC 2 architecture has the following advantages:

- Clear separation between presentation logic and business logic:- Each object in MVC has distinct responsibilities. All objects and classes are
independent of each other. So change in one class does not require alternation in other classes.

- Multiple views using the same model:- The separation of model and view allows multiple views to use the same model. This is not only facilitates easier implementation of enterprise model but also easier to test, and maintaining of the enterprise application.

- Efficient modularity:- This architecture highly supports modular development of the application either by the use of different controllers for each module or single controller with different action classes.

- Easier support for new types of clients:- This model is easier to support for new types of clients. We need to view and controller for each type of client and wire them into the existing enterprise model.

- Support for web applications: This model is often seen in web applications.

- High scalability:- Controllers and views can grow as the model grows; and older versions of the views and controllers can still be used as long as a common interface is maintained.

- This model supports easier maintenance of the code and future improvements of the application.

V. CONCLUSION AND FUTURE WORK

In this paper, we presented the design of Operational BI system using MVC Model 2 architecture of J2EE. It is well proven and industry accepted architectural pattern. The tiered (or layered) architecture of the proposed system presented and explained briefly the major components associated in each tier. Explained how MVC Model 2 architecture works. The flow of user request and system response is explained. The interaction between various objects in MVC architecture such as user, request manager, controller, JSP and bean is explained with the help of sequence diagram. The structure of web application of the proposed system is explained. The generic object flow of the proposed Operational BI system is presented. The system architecture of the proposed system is presented and explained.

In the future, the design of proposed Operational BI system will be implemented as a prototype implementation. The functionality of the proposed system will be tested for one of the business verticals.

REFERENCES


Abstract—Business Intelligence (BI) becomes an essential element in decision making system. BI provides decision making information for strategic and tactical users. Operational Business Intelligence (Operational BI) is an extension of BI functionality into operational level of the business. Thus, Operational BI provides decision making information not only for strategic and tactical users but also operational users. The characteristics of Operational BI system is different from traditional BI system in terms of low latency, reduced access time, real time alerts and notification, large number of users, process oriented and event driven. So there is a great need for enterprise architectural framework that supports the characteristics of Operational BI system. In this paper, we present an architectural framework for Operational BI system that uses Model View Controller (MVC) framework which is a well proven design pattern of Java 2 Platform, Enterprise Edition (J2EE). The proposed framework of the system is presented in terms of multi-tiered architecture, MVC Model 2 architecture, generic objects flow and sequence diagram. Finally, the system architecture of the proposed system is envisaged. The proposed architectural framework is highly scalable and supports for enterprise Operational BI applications and systems.

Index Terms—Architectural framework, Business Intelligence, J2EE, Model View Controller, Model 2 Architecture, Operational Business Intelligence.

I. INTRODUCTION

Business Intelligence (BI) becomes an essential element in decision making system. Decision making is a major part of the modern business and is essential in all most all business organizations. BI software aims [19] to enable business users to easily access and analyze relevant enterprise information for timely and fact-based decisions. The use of BI is gaining in all most all organizations irrespective of business size and functionality which includes small, medium and large in domestic, multi-national and even transnational organizations for strategic and tactical decision making purposes. Today, it is difficult to find a successful enterprise that has not leveraged BI technology for its business [19]. The advancement of technology and tools in today’s world provide us to extend the functionality of BI into operational level decision making of the business in addition strategic and tactical decision making that evolves as new area of BI which is known as Operational Business Intelligence (Operational BI). Operational BI is one of the fastest growing areas of BI [6]. Operational BI works on near real time basis and provides decision making information in current time. Thus, Operational BI systems are also known as dynamic BI, real time BI, operational intelligence and operational analytics. Operational BI provides low level decision making information to front line managers of business for day to day business operations in addition to tactical and strategic decision making information as opposed to traditional BI.

Traditional BI systems are static, historic in nature, non-process oriented and highly data driven. In addition, they have very limited user access and limited view of decision making information, what is happening in current time of the business is not known. Moreover, traditional BI systems are monolithic, client server and non-web based architectures. The characteristics of Operational BI system are low latency, reduced access time, real time alerts and notification, large number of user access, event driven, process-orient and decision making information in current time those are different from traditional BI system. Thus, traditional BI systems suffer many drawbacks as compared with Operational BI systems internals of providing decision making information in current time, low scalability, non-monitoring of business performance measurements, dynamic configuration of business performance parameters, real time alerts and limited user access. So there is a great need to develop an architectural framework for Operational BI systems which is found to be an open research problem. This motivates us to develop an architectural framework that support for operational business of an organization to provide decision making information not only in present time but also strategic and tactical users.

The goal of this paper is to propose an architectural framework for Operational BI system that supports multi-tier architecture, decision making information in current time, low response time, large number of user access, business process monitoring information and real time alert notification. The proposed architectural framework uses Model View Controller (MVC) Model 2 architecture of Java 2 Platform, Enterprise Edition (J2EE). MVC architecture supports multi-tier architecture, faster response and highly scalable. The proposed architectural framework of the system is based on the functional architecture of the system as described [10]. The proposed architectural framework is browser based and is highly suitable for real time or near real time applications on day to day business environment. The proposed architectural framework supports to meet the characteristics of Operational BI system. Discussion on data warehouse system and
A layered methodology for designing ETL processes in Operational BI systems was proposed [7] that follow in successive, stepwise refinements from high level business requirements, through several levels of more concrete specifications and down to execution models. The key feature of this layered methodology is a unified formalism for modeling the operational business processes of the enterprise as well as the processes for generating the end-to-end information views required by operational decision-making. This layered methodology starts with a conceptual specification from which the logical definition and physical implementation are systematically derived. Included in the conceptual model is the specification of quality objects (or QoX objectives) which drive the design and optimization at the logical and physical levels.

An Operational BI system essentially consists of two architectural entities namely Corporate Information Factory (CIF) and Operational Data Store (ODS) that was described [7] which is based on the concepts of W.H. Inmon. These two components CIF and ODS will play a dual role interns of supporting both decision support and operational transaction processing. ODS is used as an intermediate layer between operational systems and a data warehouse that has three properties such as volatile, detailed and current valued. The concept of integrated ODS was proposed for large scale architecture system.

The impact while shifting from strategic BI to Operational BI was discussed [21] in terms of increased number of users, the data volumes needed to support operational BI for handling of a mixed workload including operational response time, short tactical queries, massive analytical queries, thousands of concurrent users and large volume of data.

The different levels of Operational BI were described [20] that are analyze, monitor, facilitate and execute. In the first level of operational BI, users analyze operational process using traditional reports. The next level occurs when user monitor process on a just-in-time basis using graphical key performance indicator. In the next level, IT developers facilitate processes by embedding BI into operational applications using SOA to merge operational and analytical processes into a single application. Finally, the culmination of Operational BI is when organization executes the process using event driven analytic engines, predictive models and other techniques that monitor events and trigger rules to automate or guide actions.

The key features of Operational BI system were presented [15] which are namely low latency and reduced action time, access to lowest granularity data, real-time alerts, faster query response time, more ad hoc querying capability, support for Streaming SQL, flexible to integrate the existing business processes and workflows, performance measurements of configurable parameters and timely information to the users of the system. Moreover, mapping between the key features of the system were presented with their equivalent functional modules. A holistic view of Operational BI system was presented [15] that consists of set of abstract layers namely data sources, data services, Operational BI engines and support services, service delivery and delivery channels.

The functional architecture of Operational BI system was presented [10] which are based on the key features [15]. The various functional layers of the system are envisaged as data sources, data services, analytics engine, reporting engine and portal. In addition, alert engine, business services, metadata management and user and security. The data sources consist of set of operational databases which acts as an input to the system. The functionality of data services layers includes data integration, storage, compression, OLAP, querying and streaming. The functionality of analytic engine is not only to provide decision making information in current time but also form historical systems. The business services layer covers the functionality of various modules such as workflow, key performance indicators (KPIs) and service level agreements (SLAs), business rules definition, logging and monitoring. Alert engine generates alerts on real time basis to the user for
timely information. Reporting engine will provide various reporting tools that include data visualization, dashboards and linking between operational and strategic reports. The portal will act as a single point of contact for information dissemination to the various users of the system.

III. OPERATIONAL BUSINESS INTELLIGENCE FRAMEWORK

In this section, Operational BI architectural framework is presented which covers multi-tiered architecture and various components associated in each tier, MVC framework, MVC Model 2 architecture, generic objects flow. Moreover, the sequence diagram and deployment diagrams of the proposed framework of the system are presented.

A. Multi-tiered Architecture

The proposed Operational BI is considered as stack of tiers. A tier is a logical representation of concerns in the system. Each tier is responsible for set of tasks that are performed by the components associated with in the tier. Each tier is logically separated from another and is loosely coupled with the adjacent tier. The multi-tiered architecture of Operational BI system framework is shown in figure 1.

<table>
<thead>
<tr>
<th>Client Tier</th>
<th>Application client, Device and GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation Tier</td>
<td>Java Server Pages, Servlets and UI elements</td>
</tr>
<tr>
<td>Business Tier</td>
<td>Workflow, Business rule engine and Business Objects</td>
</tr>
<tr>
<td>Integration Tier</td>
<td>IMS, JDBC, SOA, Connectors and Legacy</td>
</tr>
<tr>
<td>Resource Tier</td>
<td>Databases, external systems and legacy applications</td>
</tr>
</tbody>
</table>

Fig. 1. Multi-tiered architecture of operational business intelligence.

Resource tier consists of different data sources that includes operational data sources, external systems and legacy applications, email and web repositories. In addition, this may include other transaction applications, ftp servers, directory servers, back-office systems like enterprise resource planning (ERP), supply chain management (SCM) front-office systems like customer relationship management (CRM) and custom business applications. Resource tier will act as input to the proposed system. In addition, resource tier contains the business data and external data or application sources, legacy systems and data warehouse system. The design of data warehouse system is out of scope of this paper and considered as an external system to the proposed architecture.

Integration tier consists of components like Java Database Connectivity (JDBC), connectors, SOA, Java Messaging Service (JMS), other legacy applications and real-time integration. Integration tier is responsible for communicating with external resources and systems. The business tier is coupled with the integration tier whenever the business objects require data or services that reside in the resource tier.

In addition, this tier consists of real-time ETL, enterprise information integration (EII) and enterprise application integration (EAI) frameworks.

Business tier provides the business services required by the client. This tier contains the business data and business logic. Typically, most business processing for the application is centralized in this tier as described [14]. Mostly workflow, Java bean components or Enterprise beans are used as business objects. As described [9] Java bean components are Java classes that can be easily reused and composed together into applications. JSP technology directly supports using Java bean components with JSP language elements. Java beans are easy to create and initialize parameter’s values using get and set methods. The various business services of Operational BI includes login, logging, master data management, analytical, Metadata, key performance indicators (KPIs), Service level agreements (SLAs) monitoring, workflow, business rule engine, monitoring engine, alert engine, reports and dashboards as described in [10].

Presentation tier encapsulates all presentation logic required to service the clients that access the system. The presentation tier intercepts the client’s requests; controls access to business services, construct the responses and finally deliver the response to the clients. The presentation layer contains Servlets and Java Server Pages (JSPs) that produce User Interface (UI) elements.

Client tier represents all devices or system clients accessing the system resources. A client can be a Web browser, a Java application, or a device and Graphical User Interface (GUI).

B. Model View Controller (MVC) Architectural Framework

MVC is software architectural design pattern which is a well-proven, standard and industry accepted architecture. The MVC architecture has its roots in Smalltalk, where it was originally applied to map the traditional input, processing, and output tasks to the graphical user interaction model as mentioned [12]. Many enterprise applications use the design principles of MVC architecture as described [1], [12], [16], [17], [18]. The very advantage of MVC architecture is separations of layers which supports multi-layer architecture. This provides change in one layer can be accomplished without altering the other layers of the system. The MVC pattern can be implemented with programming languages such as Smalltalk, Java, C, C++ and Microsoft .NET. The MVC Model 1 architecture is shown in figure 2.

Fig. 2. MVC Model 1 architecture.
In order to develop web applications, Servlets and JSPs technologies are commonly used in J2EE. The advantages of Servlets over Common Gateway Interface (CGI) are easier to develop web applications, faster to run, platform independent, handle multiple requests concurrently and synchronize requests, improved performance, scalability, reusability, safety and secure.

Servlets create threads to handle requests instead of creating process results so no separate memory area is required. Thus many subsequent requests can be easily handled by the servlets. However, there are limitations of Servlets in terms of recompilation of code as and when there is change. Moreover, Servlets do not provide separation of business logic from presentation logic.

JSP overcomes the limitations of Servlets which provides separation between presentation and business logic. There is no need to recompile the application if JSP code is modified unlike Servlets. Java beans are used in JSPs for developing applications. Custom tags and Java Standard Tag Library (JSTL) will provide reuse of components in JSP pages which results in changes in the system can be handled more easily.

The logical flow of request and response in Model 1 architecture is summed up as follows:

1) Browser sends request for the JSP page.
2) JSP page accesses Java bean and invokes business logic.
3) Java bean connects to the database and gets or saves data into database.
4) Response sends to the browser which is generated by JSP.

In MVC Model 1 architecture, JSP alone acts as view and controller this result in no separation of presentation logic from business logic, decentralized navigation control and support for small and medium size applications.

Figure 3 shows MVC Model 2 architecture. The client will access the application through browser. All user requests are handled via controller and responses through JSP. Internally, model (Java bean) will connect to the enterprise servers/databases inorder to access the required data.

**View** encapsulates the presentation of the data. The view renders the contents of a model. It accesses enterprise data through the model and specifies how that data should be presented. It is the view's responsibility to maintain consistency in its presentation when the model changes. This can be achieved either by using a push model or pull model. In push model the view registers itself with the model for change notifications whereas in pull model the view is responsible for calling the model when it needs to retrieve the most current data.

**Controller** accepts inputs from the user and makes request from the model for the data to provide a new view. The controller translates interactions with the view into actions to be performed by the model. In a stand-alone GUI client, user interactions could be button clicks or menu selections, whereas in a web application, they appear as GET and POST HTTP requests. The actions performed by the model include activating business processes or changing the state of the model. Based on the user interactions and the outcome of the model actions, the controller responds by selecting an appropriate view.

The request and response flow of the system is envisaged as follows:

1) User requests are directed to the controller servlet.
2) The controller servlet accesses required data and builds the model, possibly delegating the processing to helper classes.
3) The controller servlet (or the appropriate sub-ordinate task) selects and passes control to the appropriate JSP responsible for presenting the view.
4) The view page is presented to the requesting user.
5) The user interacts with the controller servlet (via the view) to enter/modify data, traverse through results.

MVC Model 2 architecture has the following advantages over MVC Model 1. In MVC Model 2 Servlets acts as controller, JSP as view and Java bean as model. Hence, there is a clear separation between all the three layers as compared with MVC Model 1 architecture, centralized navigation control, easy to maintain, easy to test and highly scalable for enterprise applications.

**C. Generic Objects Flow**

The general structure of a web application using MVC Model 2 architecture is Sun's attempt to wrap JSP within the MVC paradigm [12] which has front controller, data access and application logic, Service-To-Worker and Dispatcher View, Intercepting Filter, Value List Handler and Data Access Objects (DAOs). The generic object flow of the proposed architectural framework is envisaged in figure 4.

Front Controller is a Servlet that acts as the centralized entry point in a web application. This performs managing request processing, authentication, authorisation services and ultimately selecting the appropriate view.

Data access and application logic contain entirely within the controller servlet and its helper classes. The controller servlet (or the helper class) should select the appropriate JSP page and transfer control to that page object based on the request parameters, state and session information. One of the major advances that come with JSP Model 2 is Sun's specification of the JSTL which specifies the standard set of tags for iteration, conditional processing, database access and many other formatting functions.
The guidelines associated with JSP Model 2, Sun also provided a set of blueprints for building application using the MVC paradigm and these blueprints renamed the J2EE Core Patterns.

Service-To-Worker and Dispatcher View strategies for MVC application where the front controller module defers processing to a dispatcher that is selected based on the request context. In the Dispatcher View pattern, the dispatcher performs static processing to select the ultimate presentation view.

In the Service-To-Worker pattern, the dispatcher’s processing is more dynamic, translating logical task names into concrete task module references and allowing tasks to perform complex processing that determines the ultimate presentation view.

Intercepting Filter allows for pluggable filters to be inserted in the “request pipeline” to perform pre and post processing of incoming requests and outgoing responses.

D. Sequence Diagram

The sequence diagram of MVC with request dispatcher (or manager) is shown in figure 5. This shows the interaction between various objects such as user, request manager, controller, JSP and Java bean.

E. System Architecture

The system architecture of the proposed Operational BI is shown in figure 6. Data resource layer contains all the legacy application data, data sources, data warehouse and Metadata. Operational BI engines with admin services, user management and other resources code are deployed on application server. Operational BI engines have various sub-engines that include ETL, Real Time ETL, business rules, Online analytical processing (OLAP), data compression, reporting, dashboards, alert notification and monitoring, analytics and SQL streaming as mentioned in [15].

In addition to this admin services include managing various business services, configuration of parameters to be measured and user management. Web server contains various UI components and as well as portal. All the users of the system will access the resources through the browser. The portal acts as single entry point to the users to access the applications.
resources that are available in the system. This also acts information dissemination and collaboration tool.

MVC is an architectural design pattern of J2EE which is well proven and industry accepted architecture. The main advantage of the MVC design pattern is to separate the application object (model) from the way it is represented to advantage of the MVC design pattern is to separate the well proven and industry accepted architecture. The main resources that are available in the system. This also acts controller modifies model (business logic). The proposed that removes page centric property of MVC 1 and separates (controller). This pattern contains two models MVC 1 and the user (view) from the way in which the user controls it MVC 2 for implementation. MVC 1 architecture is page centric and tightly coupled between page and model. The proposed design of Operational BI system uses MVC 2 model that removes page centric property of MVC 1 and separates presentation logic and application logic. The controller receives all requests for the application and is responsible for taking appropriate action for each request. Model contains the state (data), view displays model to user (presentation) and controller modifies model (business logic). The proposed design of the system will support the 9 Key features of the Operational BI as described [15] and functional architecture as described [10]. The proposed architecture framework supports the characteristics of Operational BI systems intern of multi-tier architecture, reduced data latency, supports real time applications, access for large number of users and can be used for enterprise Operational BI applications and systems.

The proposed architectural framework of Operational BI has the following advantages:

1) Clear separation between presentation logic and business logic: - Each object in MVC has distinct responsibilities. All objects and classes are independent of each other. So change in one class does not require alteration in other classes.

2) Multiple views using the same model: - The separation of model and view allows multiple views to use the same model. This is not only facilitates easier implementation of enterprise model but also easier to test, and maintaining of the enterprise application.

3) Efficient modularity: - This architecture highly supports modular development of the application either by the use of different controllers for each module or single controller with different action classes.

4) Easier support for new types of clients: - This model is easier to support for new types of clients. We need to a view and controller for each type of client and wire them into the existing enterprise model.

5) Support for web applications: This model is often seen in web applications.

6) High scalability: - Controllers and views can grow as the model grows; and older versions of the views and controllers can still be used as long as a common interface is maintained.

7) This model supports easier maintenance of the code and future improvements of the application.

IV. DISCUSSION

MVC is an architectural design pattern of J2EE which is well proven and industry accepted architecture. The main advantage of the MVC design pattern is to separate the application object (model) from the way it is represented to the user (view) from the way in which the user controls it (controller). This pattern contains two models MVC 1 and MVC 2 for implementation. MVC 1 architecture is page centric and tightly coupled between page and model. The proposed design of Operational BI system uses MVC 2 model that removes page centric property of MVC 1 and separates presentation logic and application logic. The controller receives all requests for the application and is responsible for taking appropriate action for each request. Model contains the state (data), view displays model to user (presentation) and controller modifies model (business logic). The proposed design of the system will support the 9 Key features of the Operational BI as described [15] and functional architecture as described [10]. The proposed architecture framework supports the characteristics of Operational BI systems intern of multi-tier architecture, reduced data latency, supports real time applications, access for large number of users and can be used for enterprise Operational BI applications and systems.

The proposed architectural framework of Operational BI has the following advantages:

1) Clear separation between presentation logic and business logic: - Each object in MVC has distinct responsibilities. All objects and classes are independent of each other. So change in one class does not require alteration in other classes.

2) Multiple views using the same model: - The separation of model and view allows multiple views to use the same model. This is not only facilitates easier implementation of enterprise model but also easier to test, and maintaining of the enterprise application.

3) Efficient modularity: - This architecture highly supports modular development of the application either by the use of different controllers for each module or single controller with different action classes.

4) Easier support for new types of clients: - This model is easier to support for new types of clients. We need to a view and controller for each type of client and wire them into the existing enterprise model.

5) Support for web applications: This model is often seen in web applications.

6) High scalability: - Controllers and views can grow as the model grows; and older versions of the views and controllers can still be used as long as a common interface is maintained.

7) This model supports easier maintenance of the code and future improvements of the application.

V. CONCLUSION

Traditional BI is static, historic in nature and mainly used for strategic and tactical decision making whereas Operational BI is dynamic, current in time and provides decision making information not only in current time but also strategic and tactical levels. Operational BI works on near real time/real time, event based and low data latency whereas traditional BI based on data driven and high data latency. The characteristics of Operational BI system is different from traditional BI system in terms of low latency, reduced access time, real time alerts and notification, large number of users, process oriented and event driven. In this paper, the architectural framework for Operational BI system is presented using MVC Model 2 architecture of J2EE which is well proven and industry accepted architectural pattern. The tiered (or layered) architecture of the proposed system is presented and explained the major components associated in each tier. Explained MVC Model 2 architecture works and how this is different form MVC Model 1. The flow of user request and system response is explained. The interaction between various objects in MVC architecture such as user, request manager, controller, JSP and bean is explained with the help of sequence diagram. The structure of web application of the proposed system is explained. The generic object flow of the proposed Operational BI system is presented. The system architecture of the proposed system is presented and explained.

In the future, the proposed architectural framework of Operational BI system can be implemented as a prototype implementation. The functionality of the proposed system will be tested for one of the business verticals. The proposed architecture can be further implemented using one of the programming languages such as Small talk, Java, Microsoft .NET. Further, the proposed architectural framework can be extended into hierarchical MVC architecture for multiple client tiers.

REFERENCES


A.D.N.Sarma is the first author of this paper originally belongs to Vijayawada, Andhra Pradesh and born in 1964. He earned B.Sc and M.Sc (Electronics) degrees from Acharya Nagarjuna University in 1984 and 1998 respectively. He received M.Phil degree from Andhra University in 1993. He got M.S. (Software Systems) from B.I.T.S., Pilani in 1998 and received M.B.A from Pondicherry University in 2007. Currently the author is pursuing Ph.D in Computer Science and Engineering from Acharya Nagarjuna University.

Sarma has more than 24 years of experience that includes research, Information technology and academic. His interesting area includes Information Systems, Information Technology Management, Software Engineering, Software Architecture, Algorithms, Data Mining, Business Intelligence and E-Commerce. He is a Global Member of Internet Society since 2009, Member of International Association of Engineers (IAENG) and Member of the Computer Science Teachers Association (CSTA). He has published and presented more than 15 papers in various reputed international conferences and Journals.

Dr. R. Siva Rama Prasad is working as Research Director in Computer Science and Engineering and Coordinator for International Business Studies, Acharya Nagarjuna University. He earned M.B.A and M.C.A degrees and Ph.D from Acharya Nagarjuna University.

Prasad has more than 20 years of experience that includes research, Information technology and academic. His interesting area includes E-Commerce, Information Systems, E-Governance, HRD and Finance.

He has published more than 33 research papers, attended 80 National and International seminar, delivered 104 extension lectures and published 5 books.
A Novel Algorithm for Incremental Frequent Itemsets Mining

A.D.N.SARMA
Department of Computer Science & Engineering
Acharya Nagarjuna University, Guntur- 522 510, India
adnsarma@yahoo.com

Dr.R.Siva Rama Prasad
Department of International Business Studies
Acharya Nagarjuna University, Guntur- 522 510, India
raminenisivaram@yahoo.co.in

Abstract—Frequent Itemsets mining is very important research problem in knowledge discovery and data mining areas. Most of the proposed algorithms in the literate for finding frequent itemsets discuss about the full scan of the data warehouse database. Thus, for each update of data warehouse database demands to rerun of data mining algorithms in order to extract frequent itemsets which is a monotonous task. In this paper, we present a novel algorithm for incremental frequent Itemsets mining which eliminates to rerun of data mining algorithm from scratch. The proposed algorithm updates database of data warehouse dynamically from the incremental database which are generated from operational data sources. The very salient feature of the proposed algorithm is to update the extracted knowledge database from dynamically generated frequent itemsets of incremental transaction database.

Keywords—Association rules, data mining, frequent itemsets, frequent pattern, incremental mining, knowledge discovery, large databases, maintenance.

I. Introduction

Association mining is one of the most important research problems in knowledge discovery and data mining areas which facilitates to extract association rule, frequent patterns in data. The frequent patterns bring out the combination of events that occur at the same time which includes frequent itemsets in large databases. Most of the proposed algorithms in the literate for finding frequent itemsets discuss about the full scan of the data warehouse database. In the past researchers assumed database of data warehouse is static in nature inorder to simplify data mining problems. Thus, most of the proposed algorithms focused on batch mining full database scan for every update and did not utilize previously mined information in incrementally growing databases.

Incremental mining means to find delta in the source and extract the knowledge from the delta and update with the extracted knowledge. In most of the real world applications developing a mining algorithm that can suitable for incrementally maintain discovered information as a database grows is quite important and is open research problem. The real-world databases, from which useful patterns and rules are mined, are dynamic in nature. Periodically, the organizational database is updated, and it may become necessary to carry out the mining process again on the updated database. Since mining is a costly activity, typically requiring multiple database scans, process of incremental mining is proposed by researchers to maintain the rules discovered during previous mining processes. The objective of incremental mining is to avoid re-learning of rules from the old data [23] and utilize knowledge that has already been discovered.

In this paper, we proposed a novel algorithm that updates knowledge database from dynamically generated frequent itemsets from incremental transaction database in addition to the update of data warehouse of database. The proposed algorithm is highly scalable and suitable for large volumes of continuously growing databases. This algorithm is highly suitable for improving the decision making not only at strategic, tactical but also at operational level because of dynamic update of knowledge database directly from incremental transaction database.
The remainder of the paper is organized as follows. Section 2 provides a brief review of the related work. Section 3 introduces the structure and description of the proposed algorithm. Section 4 covers conclusion of our work.

II. Relevant Work

Recent studies have shown that there are various algorithms for incremental mining and sequential patterns. Sequence mining finds its application in many different areas in business industrial and medicine. A few efficient algorithms incremental mining frequent sequences have been proposed [1] [2] [4] [6] [10] [13] [19] in the literature.

A few efficient algorithms for mining frequent sequences have been proposed in the literature are AprioriAll [24], GSP [25], SPADE [26], MFS [9], and PrefixSpan[28].

In a typical data mining process, full data is rarely collected in one attempt. The collection of data is a continuous and ongoing process. Typically in the context of business operations the data collection time is as small as weeks, days, hours, minutes and even seconds. In many cases, data collection is carried out in phases. Consequently, the content of the underlying database changes over time. To keep track of the frequent sequences, sequence mining algorithms have to be executed whenever the underlying database changes. We refer to this problem as incremental mining which includes update, insertion and deletion operations in not only data warehouse database but also the corresponding operations extracted knowledge to get timely update of discovering frequent patterns.

A simple approach to the update problem is to mine the new database from scratch, using sequence mining algorithm. This approach, however fails to take advantage of the valuable information obtained from the previous mining results. In order to utilize the previous mining results i.e. extracted knowledge from data warehouse database to efficiently mine the updated database, several incremental mining algorithms have been proposed which include GSP+ [10], MFS+ [10], and ISM [11]. The above three algorithms have been based on GSP, MFS and SPADE respectively. The structures of GSP+ and MFS+ follow those of GSP and MFS.

Toivonen [22] has proposed a sampling based algorithm for mining association rules. They take a random sample of the database, find large itemsets, and then verify the results with the whole database. Chang et al. [3] proposed an incremental approach to buffer the CSTree of the old database. This approach is also capable of mining the closed frequent sequences. Toshi Chandraker et al [2] proposed an incremental mining algorithm based on two support thresholds which further reduces the need for rescanning original databases.

Cheung and his co-workers proposed an incremental mining algorithm, called FUP (Fast UPdate algorithm) [20], for incrementally maintaining mined association rules. It first calculates large itemsets mainly from newly inserted transactions, and compares them with the previous large itemsets from the original database. According to the comparison results, FUP determines whether re-scanning the original database is needed, thus saving some time in maintaining the association rules. Although the FUP algorithm can indeed improve mining performance for incrementally growing databases, original databases still need to be scanned when necessary. FUP can only handle the maintenance problems in the case of insertion.

FUP\textsuperscript{2} is developed [1] further to address the new rules in an update cases including insertion, deletion and modification of transactions and is a complementary algorithm of FUP. FUP\textsuperscript{2} technique updated the association rules when old transactions are removed from the database and new transactions are added to it. It uses information available from a previous mining.
Parthasarathy et al [11] proposed ISM which is based on SPADE, by maintaining a sequence lattice of an old database. The sequence lattice includes the frequent sequences and a negative border which includes sequences which are infrequent but their subsequences are frequent. Pei et al [28] proposed IncSpan which is based on PrefixSpan by buffering a set of semi-frequent sequences.

III. Algorithm

In this section, we describe the proposed incremental frequent itemsets mining algorithm and its salient features.

A. Algorithm

The proposed algorithm of incremental frequent itemsets mining is envisaged below. The inputs to the algorithm are data warehouse database, extracted pattern database or knowledge database, and incremental database. The output will be updated frequent itemsets.

Algorithm: Incremental Frequent Itemsets Mining

Input:

1. Data warehouse database.
2. Knowledge base (or extracted pattern database) of data warehouse database and
3. Incremental database.

Output: Frequent itemsets

1. Read a transaction from incremental database.
2. Find the length (number of items) of the transaction.
3. If (number of items > 1)
4. ItemSetGen (transaction, 1-itemset, n-itemset)
   // Generate 1- Itemset and n-Itemset from the transaction of incremental database.
5. Update counts of 1-Itemset and n-Itemset, count of transactions and extracted knowledge database.
6. Else
7. ItemSetGen (transaction, 1-itemset) - Generates 1 –itemset from the transaction of incremental database.
8. Update counts of 1-Itemset and count of transactions and extracted knowledge database.
9. End If
10. Update the database of data warehouse with transaction of incremental database.
11. CheckSup () - checks the support threshold of 1-Item set of knowledge database
12. GenReqItemsets ()
   // Generate required itemsets from the transaction of incremental database.
13. Update counts of sub-itemsets of knowledge database.
14. Repeat the process until all the transactions of incremental database completes.

Data warehouse database: This is the data of data warehouse which contains the historical data. It is single source of truth of an organization. This database is used for reporting and data analysis. It is a central repository of data which is created by integrating data from multiple disparate sources. Data warehouse database stores current as well as historical data and are commonly used for creating trending reports for senior management reporting such as annual and quarterly comparisons which is used for Strategic, tactical and operational decision making.
Knowledge database: In order to extract the knowledge from the database of data warehouse mining algorithm will be run. The extracted information contains the frequent patterns and other information which is known as knowledge database. In the proposed algorithm the previously mined information will be updated from newly obtained knowledge after processing each individual transaction of incremental database.

Incremental database: This is the data obtained from the operational systems i.e. transaction databases in a given window of time interval and the time interval will vary from case to case basis and typically ranges days, hours or even minutes.

Incremental mining: Each transaction of incremental database is extracted and updated both knowledge database and as well as database of data warehouse.

B. Silent Features
The proposed algorithm has the following salient features:

It works dynamically – each transaction of incremental database is processed and then the corresponding Itemsets will be updated in knowledge database including item set counts, itemsets, transactions count and database data warehouse.

It avoids re-computing large itemsets that have already been discovered. In the proposed algorithm there is no need of multiple scan or even re-scan of the database for computing large itemsets that have been discovered because the system directly used the extracted knowledge from the previously.

The proposed algorithm focuses on newly inserted transactions, thus greatly reducing the number of candidate itemsets generation which intern increases computation time. As a result it improved the efficiency of the proposed algorithm.

It uses a simple check of 1-itemset support count before to update the frequent item counts and then generates dynamically the required frequent itemsets from the transaction of incremental database. This results the optimization of the Itemset generation which intern saves computation time.

Keeping all the above features of the proposed incremental frequent Itemset mining algorithm it is to conclude that the proposed algorithm is not only superior over other algorithms, however experiments to be furnished and this work is in progress, but also highly scalable for any large incremental databases. This algorithm is highly suitable for operational business intelligence systems which provide miniscule details of frequent itemsets information for low level decision making and as well as strategic decision making to the organizations.

II. CONCLUSIONS
We presented a novel algorithm for incremental frequent Itemset mining for the maintenance of frequent pattern mining. This algorithm updates the extracted knowledge database with dynamically generated frequent itemsets from the incremental transaction database. It uses the knowledge available from the previous mining to reduce the amount of work that has to be done to discover the frequent patterns in the updated database of data warehouse.

The performance studies show that the proposed algorithm is significantly faster than mining the updated database of data warehouse from scratch.

The proposed algorithm has the following salient features:

• It works dynamically.
• Avoids re-computing large itemsets that have already been discovered.
• Focuses on newly inserted transactions, thus greatly reducing the number of candidate itemsets which intern increases computation time.
• Uses a simple check of 1-itemset support count and then generates the required frequent itemsets dynamically from the transaction.
The proposed algorithm works well over wide ranges of system parameter values. In particular, it works well for updates of a wide range of insertion sizes and small to moderate deletion sizes. Further researches could be extended to problems of various minimum supports and problems of generalized sequential patterns.

IV. References


[9] Ding Yuan, Kyuhyung Lee, Hong Cheng, Gopal Krishna, Zhenmin Li,Xiao Ma, Yuanyuan Zhou+ and Jiawei Han, "CISpan: Comprehensive Incremental Mining Algorithms of Closed Sequential Patterns for Multi-Versional Software Mining", In Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA, pp. 84-95.


AMKIS: An Algorithm for Association Mining Kth Itemset Frequent Pattern in Large Databases

A.D.N.Sarma
Research Scholar
Dept. of Computer Science & Engineering
Acharya Nagarjuna University
Guntur- 522 510, India
adnsarma@yahoo.com

Dr.R.Siva Rama Prasad
Research Director
Dept. of CSE, Coordinator IBS
Acharya Nagarjuna University
Guntur-522510, India
raminenisivaram@yahoo.co.in

Abstract—Mining frequent items and itemsets is a daunting task in large databases and has attracted research attention in recent years. Generating specific itemset, Kth—itemset having K items, is an interesting research problem in data mining and knowledge discovery. In this paper, we propose an algorithm for finding Kth itemset frequent pattern generation in large databases which is named as AMKIS. AMKIS algorithm uses no candidate generation and minimum support criteria’s for generating Kth itemset frequent pattern. The structure and functionality of AMKIS is different from Apriori and FP tree based algorithms. Further, this algorithm does scan transaction database once. AMKIS performance is compared with Apriori algorithm. Our extensive performance study shows that AMKIS algorithm has higher performance as compared with Apriori. The proposed algorithm, AMKIS, is highly scalable for mining not only small but also large Kth itemset frequent patterns and is linearly scalable in terms of the database size.

Keywords— association mining, association rules; business intelligence; data mining; frequent itemset; frequent patterns; knowledge discovery; Kth itemset and mining methods.

I. INTRODUCTION

Frequent patterns mining is essentially one of the most important concepts in knowledge discovery and data mining. From frequent pattern mining concepts many other data mining tasks and theories are evolved that includes sequential pattern mining, structured pattern mining, correlation mining, associative classification, link mining and frequent pattern-based clustering. Abundant research has been reported in this area of frequent pattern mining and presenting new algorithms and improvements on the existing algorithms to solved mining problems more efficiently and highly scalable.

Agrawal et al. [25] reported frequent itemset mining and association rule mining first time in 1993. Since then frequent itemset and association rule mining problems have received a great deal of attention in research. One of the first algorithms proposed for association rules mining was the AIS algorithm [25]. The problem of association rules mining was introduced in [25] as well. This algorithm was improved later to obtain the Apriori algorithm [2]. The Apriori algorithm employs the downward closure property that is if an itemset is not frequent, any superset of it cannot be frequent either.

Most of the previous research work is based on Apriori, (FP) Frequent Pattern growth, sampling and prefix tree algorithms. Apriori algorithm suffers with generation of huge number of candidate itemsets and performs repeated passes and multiple scans of database for finding frequent itemsets. The existing mining association algorithms have some drawbacks. First, scanning of database multiple times that result overhead on input and output devices. For large database systems the IO overhead become more and demands large main memory to store whole data. This is very inefficient and time consuming because big overhead of reading the large database even though partial items are interested. Second, large number of candidate generation. Third, the association rules are sometimes very large, often thousands or even millions. So, it is very difficult to analyze all association rules available in the system. Fourth, there are no thumb rules available to the users to choose proper values for support and confidence parameters to limit association rule discovery. The users have to follow trial and error approach to get suitable number association of rules. However, in real time basis, it is difficult to provide an appropriate minimum support threshold and will vary from item to item. This is very inefficient in selecting and filtering large number of association rules to pick-up small number of items in large database and time consuming. Fifth, there is missing knowledge during pruning phase while choosing system parameters such as minim support and count values for limiting association rule discovery. Sixth, the existing algorithms demand large main memory resource for storage. Prefix-tree based algorithms may suffer from the limitation of memory size when it tries to hold whole database information.

The aim of this paper is to develop a novel, scalable and memory efficient algorithm for association mining Kth itemset frequent pattern generation in large databases that addresses the problems such as to eliminate multiple scans of database, use of support criteria, excess generation of candidates and missing knowledge while extraction and complex data structures. In this paper, we propose an efficient association mining algorithm for finding Kth frequent itemset directly from large transaction databases. This algorithm is referred as Association Mining Kth ItemSet and is called AMKIS from
Apriori algorithm \[3\] generates a set of candidate large itemsets frequent pattern in large databases. It performs a breadth-first search in the search space by generating candidate \(K\)th itemset frequent pattern by recursively finding all frequent 1-item sets in the current transaction database, no massive candidate generation, it extracts the missing knowledge that is lost during the pruning process during support count thresholds, uses limited main memory resources, uses simple data structure, and does not use any support criteria. The Apriori algorithm is fundamentally different from Apriori and FP tree algorithms. We present experimental results to find scalability of AMKIS algorithm for different groups of datasets. From our experiments it is found that the AMKIS algorithm always outperforms Apriori.

The remainder of the paper is organized as follows. Section 2 deals related work. Section 3 describes problem definition. Section 4 presents the proposed algorithm, AMKIS, for mining \(K\)th itemset frequent pattern in large databases. Section 5 presents performance study of the algorithm. Section 6 covers discussion on the proposed algorithm. Finally, Section 7 concludes the paper with further work.

II. RELATED WORK

Frequent pattern mining is an interesting task and found lot of research work in the area of data mining. Frequent itemset mining has been studied extensively in literature [1], [3], [4], [5], [6], [9], [11], [14], [16], [17], [22], [24], [25]. The existing approaches for mining frequent pattern algorithms can be broadly classified into the following two categories as Apriori, and pattern growth base. In this section, we describe the major functionalities of these algorithms and recent work reported on these algorithms in the literature.

A. Apriori based algorithms

Apriori algorithm was proposed by Agrawal in 1993 for finding frequent itemset and association mining. A number of Apriori based algorithms [2], [3], [15], [17], [23], [24] have been proposed in literature to improve the performance in terms of scalability, memory efficient and computational efficiency.

The Apriori algorithm performs a breadth-first search in the search space by generating candidate \(K\)th frequent itemsets from frequent \(k\)-itemsets. The frequency of an itemset is computed by counting its occurrence in each transaction. Apriori algorithm [3] generates a set of candidate large itemsets whose lengths are \((K+1)\) from the large \(K\) itemsets where \(K \geq 1\) and eliminates those candidates, which contain not large subset. Then, for the rest candidates, only those with support over minsup threshold are taken to be large \((K+1)\) itemsets. The Apriori generate itemsets by using only the large itemsets found in the previous pass, without considering the transactions.

Many variants of the Apriori algorithm have been developed such as AprioriTid, AprioriHybrid, direct hashing and pruning (DHP), dynamic itemset counting (DIC), and Partition algorithm.

Ravi et al [3] proposed an algorithm to find frequent \(K\)-itemsets based on Apriori such that itemsets whose length is less than \(K\) will be pruned from the database. In addition to this, it generates 1-itemset as a data pre-processing step which makes execution fast for generation of \(k\) itemset as compared to Apriori.

Ming Yen Lin et al [24] proposed three algorithms, named SPC, FPC, and DPC, to investigate effective implementations of the Apriori algorithm in the MapReduce framework. DPC features in dynamically combining candidates of various lengths and outperforms both the straight-forward algorithm SPC and the fixed passes combined counting algorithm FPC. Extensive experimental results show that all the three algorithms scale up linearly with respect to dataset sizes and cluster sizes.

B. Pattern Growth algorithms

FP-growth [1] is a well-known algorithm that uses the FP-tree data structure to achieve a condensed representation of the database transactions. FP growth employs a divide-and-conquer approach to decompose the mining problem into a set of smaller problems. In essence, it mines all the frequent itemsets by recursively finding all frequent 1-itemsets in the conditional pattern base that is efficiently constructed with the help of a node link structure. FP growth algorithm uses FP tree that has been widely studied [1], [4], [9], [10], [11], [14], [20], [22] for frequent pattern mining because it can give a great performance improvement as compared with Apriori algorithm in terms of the candidate generation and test paradigm. However, FP-growth still requires two database scans which is not recommended for extraction of knowledge from operational database.

A variant of FP-growth is the H-mine algorithm [14]. It uses array-based and trie-based data structures to deal with sparse and dense data sets, respectively. Andrea Pietracaprina et al [13] proposed PatriciaMine algorithm for finding frequent itemsets. This algorithm is main-memory based and employs a Patricia trie to represent the dataset, which is space efficient for both dense and sparse datasets. The FPGrowth* [22] algorithm, which extends original FPGrowth method, also uses the novel array technique to mine all frequent itemsets. FP growth based algorithm greatly reduces the time spent traversing FP-trees, and works especially well for sparse datasets.

Jianyong Wang et al [9] proposed an algorithm TFP (Top-K Frequent) closed itemsets without minimum support using FP tree. FP-tree can be can be pruned dynamically both during and after the construction of the tree using two methods: the closed node count and descendant sum method. TFP algorithm uses both top-down and bottom-up FP-tree traversing strategy to speedup mining process. Mining frequent patterns with an FP-tree avoids costly candidate generation as defined in [10] and repeatedly occurrence frequency checking against the support threshold. It therefore achieves better performance and efficiency than Apriori-like algorithms. However, the database still needs to be scanned twice to get the FP-tree. This can be very time-consuming when new data are added to an existing database because two scans may be needed for not only the new data but also the existing data.

William Cheng et al [7] proposed CATS (Compressed and Arranged Transaction Sequences) tree algorithm which
extends the idea of FP-tree. It improves storage compression and allows frequent pattern mining without generation of candidate itemsets. It scans database only once. CATS tree is a prefix tree. CATS trees can be used with incremental updates of the transaction databases.

Syed Khairuzzaman et al [4] proposed CP (Compact Pattern) tree to generate frequent pattern mining that captures database information with one scan. CP-tree algorithm supports the functionalities for both interactive and incremental mining.

A new method was proposed [16] for mining dynamical frequent itemset called AC-MFI using the theory of ant colony algorithm. The method considers the item of transaction as a frequent itemset called AC-MFI using the theory of ant colony incremental mining.

Extends the idea of FP-tree. It improves storage compression nonempty subset of transaction identifier and T is an itemset. A transaction l items contained in T, and T is called an frequent itemset.

Let I = {i1, i2, ..., in} be a set of items. An itemset T is a nonempty subset of I. The length of itemset l is the number of items contained in T, and T is called an l-itemset if its length is l. A tuple <TID, T> is called a transaction, where TID is a transaction identifier and T is an itemset. A transaction database TDB is a set of transactions TDB = {t1,t2,...,tn}. Lk is the frequent itemsets of size k. An itemset T is contained in transaction <TID, Y> if X ⊆ Y. An association rule is an association relationship of the form X => Y, where X ⊆ I, Y ⊆ I, and X ∩ Y = φ. Our task is to mine kth itemsets in a large transaction database TDB. The input to the algorithm is a binary representation of transaction data as shown in Table 1.

### III. PROBLEM DEFINITION

Let I = {i1, i2, ..., in} be a set of items. An itemset T is a nonempty subset of I. The length of itemset l is the number of items contained in T, and T is called an l-itemset if its length is l. A tuple <TID, T> is called a transaction, where TID is a transaction identifier and T is an itemset. A transaction database TDB is a set of transactions TDB = {t1,t2,...,tn}. Lk is the frequent itemsets of size k. An itemset T is contained in transaction <TID, Y> if X ⊆ Y. An association rule is an association relationship of the form X => Y, where X ⊆ I, Y ⊆ I, and X ∩ Y = φ. Our task is to mine kth itemsets in a large transaction database TDB. The input to the algorithm is a binary representation of transaction data as shown in Table 1.

### IV. DEVELOPMENT OF Kth FREQUENT ITEMSETS

In this section, we define the various symbols used in this paper and explain step by step analysis of the proposed algorithm, AMKIS, for mining Kth frequent itemsets in a large database.

#### A. Notation

TBD stands transaction database, LUT stands lookup table, tT stands set of items in the given transaction whose item value is unity, tR is an encoded items for the given transaction tT. The length of the transaction is l. Let k is an integer whose itemsets frequent patterns to be find in the give transaction database. Lk is list of K-items in a given transaction, Kth itemset means that an itemset having K- items.

#### B. Algorithm

AMKIS (Association Mining Kth Itemset) algorithm is shown in Algorithm 1. The inputs to the algorithm are binary representation of transaction database and the value of the itemset to be found. The algorithm mines the sets of Kth frequent items in a given large transactional database. The output of the algorithm is Kth itemsets frequent pattern.

### Algorithm 1: AMKIS-Kth Itemsets frequent pattern generation

**Inputs:** Transaction database and itemset to be find (K)

**Output:** Frequent items generation

1. Transaction database TDB = {t1,t2,...,tn}
2. LUT = {{i1,i2,...,ik}, {i1,i2,...,ik}, {i1,...,ik}}
3. COUNT = {x l x is a itemsets of Lk}
4. repeat
5. for each transaction t ∈ TDB do
6. tR = itemsInTransaction(t)
7. tR = itemEncode (tR)
8. if (length of tR >= K) Lk = itemsetGen (tR)
9. Lk = itemsetGen (tR)
10. for each itemset element of L do
11. lookup of each itemset element of Lk in LUT
12. count (itemset of Lk) ++
13. end for
14. end if
15. end for
16. itemCountFiltering()
17. itemDecoding()
18. until TDB = φ

Lookup Table (LUT): a lookup table is a data structure, usually an array or vector often used to replace a runtime computation with a simpler array indexing operation. The lookup table is created dynamically based on the itemset to be generated which holds all possible itemsets. Figure 3(a) shows a typical 3-itemset lookup table and figure 3(b) shows 3 itemset counter initialization.

![Figure 1. Kth Itemset lookup table and counter initialization](image)
The items in the lookup table are created in logographical order as per the binary items represented in a transaction database. In the present implementation of algorithm lookup table is employed using vectors in Java programming language. The use of LUT in the proposed algorithm greatly reduces processing time and retrieving a value from memory is often faster than undergoing an expensive computation or input/output operation. The table is loaded with all possible itemsets that are belonging to Kth itemsets to find the interesting frequent item pattern to be found in a large transaction database.

The proposed algorithm uses the following five functions itemsInTransaction(), itemEncode(), itemsetGen(), itemCountFiltering() and itemDecoding() for generating Kth itemset frequent pattern in large databases. These functions are self-explanatory. itemsetGen() is an important function among all other functions. The algorithm of itemsetGen() is explained in next subsection.

The function itemsInTransaction() will count the number of items whose value is unity for a given transaction. The items in a transaction are represented in a binary vector as shown in table 2. Binary zero indicates the items is not present where as one indicates the item presence.

itemsEncode(): This function converts the given transaction into an encoded items as shown in figure 2. For example items i_1, i_2, i_3 are encoded as A, B, C. This is only a sample. Thus each item is encoded into corresponding symbol. A symbol may be combination of alphabets or numbers. In the same way large items of transaction can be encoded. Discussion on item mapping schemes or encoding algorithms is out of the scope of this paper. However, a typical itemset encoding scheme is envisaged in figure 2. Similarly itemDecode() performs reverse functionality of itemEncode() function. That is the encoded symbols for each item is decoded into the corresponding item names. The itemCountFiltering() function filters the frequent item that is based on item count threshold value.

### C. Generation of Kth Itemset Frequent Pattern

Generation of Kth itemset frequent pattern in large databases is shown in figure 3. The high level functionality of the proposed algorithm, AMKIS, is explained as follows:

1) Scans database and item encoding: Primarily it scans each record of transaction database. And then each transaction is encoded. During the process of encoding each binary item in the transaction whose value is unity replaced with the corresponding encoded symbol. Thus the encoded transaction contains only the symbols that are mapped against each item.

<table>
<thead>
<tr>
<th>ItemName</th>
<th>Encoded Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread</td>
<td>A</td>
</tr>
<tr>
<td>Milk</td>
<td>B</td>
</tr>
<tr>
<td>Beer</td>
<td>C</td>
</tr>
<tr>
<td>Eggs</td>
<td>D</td>
</tr>
<tr>
<td>Cola</td>
<td>E</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TID</th>
<th>Encoded Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T100</td>
<td>A, B, C</td>
</tr>
<tr>
<td>T200</td>
<td>A, B, C, D, E</td>
</tr>
<tr>
<td>T300</td>
<td>A, B, C, D</td>
</tr>
<tr>
<td>T400</td>
<td>A, B, C, D</td>
</tr>
</tbody>
</table>

Figure 2. Item encoding table

Figure 3. Generation of Kth Itemset frequent pattern
2) **Pruning and \(K^h\) itemset generation**: If the length of the encoded transaction is less than itemset to be generated then all such transactions are pruned. This way the computation time of algorithm is improved. For all other transactions possible \(K^h\) itemset will be generated. The algorithm of itemsetGen() is given in Algorithm 2. The inputs to the algorithm are \(K^h\) itemset to be determined and mapped transaction. The output will be all possible itemsets whose length is equal to \(K\). The generated itemsets are in lexicographical order as per the binary items represented in transaction database.

**Algorithm 2: Itemsets Generation**

**Inputs**: \(K^h\) itemset and mapped transaction  
**Output**: Itemsets generation

1. Declare an array of *items*  
2. Declare loopcounter  
3. Initialize the variable \(K\) itemset  
4. Initialize mapped transaction length \(l\)  
5. for 1 to loopcounter  
6. Generate binary of loopcounter  
7. if binary count of loopcounter equal to \(k\)  
   Add itemset to *items* array  
8. end if  
9. end for  
10. return *items* array  
11. end for

3) **Lookup and count increment**: Each element of the generated \(K^h\) itemset is lookup into LUT and finds its address and then the corresponding count of the itemset counter is incremented.

4) **Filtering**: The generated itemset by itemsetGen() will be filtered based on item count threshold set by the user. If the item count threshold value is unity means it filters all \(K^h\) itemset whose occurrence is at least once in the TDB.

5) **Item decoding**: Finally, the filtered items will be decoded. The functionality of decoding is reverse function of item encoding. The user can easily understand the generated \(K^h\) itemset frequent patterns from transactional database when symbols are decoded into the corresponding item name as specified in figure 2. Generally this step will be a part of data presentation to the user. Hence, the functionality of decoding can be ignored in the experimental evaluation of algorithm.

**V. EXPERIMENTAL EVALUATION**

In this section, we describe experimental setup, specifications of datasets being used and performance study of AMKIS algorithm over variety of datasets. We present performance results of AMKIS algorithm with Apriori for a given very large transaction database.

**A. Experimental Setup**

We conducted a set of experiments to test the performance of the AMKIS algorithm. The experiments were on the Compaq 420 system with Core 2 Duo T6570 CPU, Clock speed 2.10Gz, System Memory 2 GB, Storage HDD Capacity 500 GB, Hardware Interface SATA and RPM 5400. The Operating system on the system is Windows 7 Professional. AMKIS algorithm is developed and implemented using Java programming language.

**B. Data Sets**

Two groups of datasets considered to study the applicability and scalability of AMKIS algorithm for finding \(K^h\) itemset frequent patterns. Both groups of datasets are synthetic. These data sets support large range of data characteristics. The data characteristics include volumes of transactions range from tens of hundreds to several millions, average number of items per each transaction and guaranteed minimum number of items per each transaction.

The specifications of first group of datasets are summarized in the Table 2. There are eleven canonical sets of data for this dataset whose transactions ranging from 1000 to 750 thousands.

<table>
<thead>
<tr>
<th>Name of dataset</th>
<th>No. of transactions</th>
<th>Size, KB</th>
<th>No. of items</th>
<th>Average items</th>
<th>Minimum items</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS#01</td>
<td>1000</td>
<td>11</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>DS#02</td>
<td>2000</td>
<td>22</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>DS#03</td>
<td>5500</td>
<td>60</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>DS#04</td>
<td>10K</td>
<td>108</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>DS#05</td>
<td>33K</td>
<td>355</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>DS#06</td>
<td>50K</td>
<td>538</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>DS#07</td>
<td>75K</td>
<td>806</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>DS#08</td>
<td>100K</td>
<td>1075</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>DS#09</td>
<td>250K</td>
<td>2686</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>DS#10</td>
<td>500K</td>
<td>5372</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>DS#11</td>
<td>750K</td>
<td>8057</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

**C. Mining Very Large Databases & Scaleup Experiments**

To test the efficiency and scalability of AMKIS algorithm he proposed mining algorithm requires testing on very large databases. In our scale-up experiments we generated large datasets (LDS) whose transactions volume ranges from one million to seven millions. The specifications of this group of datasets are shown in table 3.

<table>
<thead>
<tr>
<th>Name of dataset</th>
<th>No. of transactions</th>
<th>Size, MB</th>
<th>No. of items</th>
<th>Average items</th>
<th>Minimum items</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDS#01</td>
<td>1M</td>
<td>10.743</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>LDS#02</td>
<td>2M</td>
<td>21.485</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>LDS#03</td>
<td>3M</td>
<td>32.227</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>LDS#04</td>
<td>4M</td>
<td>42.969</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

The proposed algorithm is experimentally evaluated with respect to Apriori algorithm for large datasets. The experimental results of AMKIS and Apriori algorithms with large database are shown in figure 6. This clearly shows AMKIS outperforms Apriori algorithm. AMKIS algorithm is highly scalable for mining not only small but also large \(K^h\) itemset frequent patterns and is linearly scalable in terms of the database size.
Figure 4. AMKIS: Execution time versus number of transactions

Figure 5. Apriori: Execution time versus number of transactions
D. Performance Results

The performance results of AMKIS algorithm with Apriori are shown in figure 8.

VI. DISCUSSION

The proposed algorithm is compared with one of the most popular frequent pattern mining algorithms called Apriori. Apriori generates Kth itemset iteratively from the frequent itemsets with cardinality from 1-itemset to K-itemset with minimum support criteria at each itemset level. Most of the previous published literature deals with database sized around 100k [2], [3], [10], [17], [25]. In our experiments, our database size is over a million transactions, which is a reasonable size for a respectable department store-like transactional database. The performance and scalability of the algorithm will be known when testing at large databases.

We generated synthetic transactions to evaluate the performance of the algorithms over a large range of data characteristics. A transaction may contain more than one large itemset. Transaction sizes are typically clustered around a mean and a few transactions have many items. Typical sizes of large itemsets are also clustered around a mean, with a few large itemsets having a large number of items.

Figure 4 shows AMKIS execution time versus transactions. Figure 4(a) shows below 33K transactions the execution time of AMKIS is steady for itemsets range from 1 to 5. Figure 4(b) shows that as transactions volume increases then the execution time decreases. In figure 4(c) shows similar trend for large volume of transactions that ranges from 1 million to 4 million. The execution time of Apriori algorithm for multiple support counts say 2%, 10%, and 20% are shown in figures 5(a) to 5(f). From figure 4 and 5 we can say that AMKIS execution time is steady and is linearly increasing as volume of transactions increases and decreasing as itemset value increases. Figure 6 shows the scaleup experiments results of large datasets ranging from thousand transactions to several million transactions. As the minimum support decreases, the execution times of Apriori algorithm increases because of increase in the total number of candidate and large itemsets. The proposed algorithm, AMKIS, for generation of Kth itemset frequent patterns in large databases is shown in figure 7. The performance results of AMKIS with Apriori are shown in figure 7. From our experimental results it is observed that as increasing itemset the execution time of AMKIS algorithm decreases as compared with Apriori. This clearly shows that AMKIS algorithm beats Apriori for generation of Kth itemset frequent patterns in large bases.

VII. CONCLUSIONS

In this paper, we have presented a new algorithm named as AMKIS for mining Kth itemset frequent pattern in large databases. We systematically explore mining of Kth itemset frequent patterns in large databases without the use of massive candidate generation, support criteria and multiple scans of database. Based on this approach, a novel algorithm for discovering the set of all Kth frequent items sequences is presented which can reduce the search space and minimize cost of computation efficiently by using generating all possible Kth itemset frequent patterns from the given large database. The structure and functionality of the proposed algorithm, AMKIS, to find Kth itemset frequent patterns is different from Apriori and FP tree based algorithms. We compared the performance of AMKIS algorithm with Apriori which is one the most popular frequent itemsets mining algorithms found in literature. The findings from different experiments have confirmed that our proposed AMKIS
algorithm is efficient for mining Kth itemset frequent pattern in large databases. It can speed up the data mining process significantly as demonstrated in the performance comparison.

The proposed AMKIS algorithm has the following salient features. This scans database only once. Hence, the high repeated disk overhead incurred in other mining algorithms can be reduced significantly. Furthermore, it provides missing intelligence which is lost during the support count pruning with other algorithms. The main memory required to run this algorithm is mainly to store look up table (LUT) data, mapping table data, item count data, and the generated Kth itemset of single transaction. Thus, AMKIS uses less main memory as compared to Apriori and FP tree based algorithms. The proposed algorithm gives a better performance for mining both short and long length frequent patterns.

We demonstrated the effectiveness and scalability of the proposed algorithm, AMKIS, using synthetic datasets. The volumes of transactions datasets ranges from thousands to several millions. It is shown to be efficient and scalable to large amount of transactions and out performs. Currently, there is no known and published algorithm that can provide the same functionalities efficiently. This makes the proposed algorithm is not only suitable for frequent pattern mining from large historical databases but also incremental transaction database of operational data sources. The $K^d$ itemset-based extension approach opens several research opportunities and future work will be done in various directions including finding TOP – K frequent itemsets, to find maximal or closed sequential patterns, multilevel association rules, Further, the proposed algorithm can be extended to mine frequent pattern from multilevel rules, clustering the association rules, constraint based association mining.

REFERENCES
[26] Jiawei Han, Jian Pei, and Yiwen Yin., “Mining frequent patterns without candidate generation,” In Proceedings of the 2000 ACM SIGMOD international conference on Management of data, Volume 29 Issue 2, pp.1-12, June 2000.
INFRIM: A Novel Algorithm for Incremental Frequent Itemsets Mining

A.D.N.Sarma, R.Siva Rama Prasad
Research Scholar, Research Director, Coordinator IBS
Dept. of CSE
Acharya Nagarjuna University, Guntur- 522 510, Andhra Pradesh, India
adnsarma@yahoo.com, raminenisivaram@yahoo.co.in

Abstract— Frequent Itemsets mining is very important research problem in knowledge discovery and data mining areas. Most of the proposed algorithms in the literate for finding frequent itemsets discuss full scan of database. Thus, for each update of database demands to rerun of data mining algorithms in order to extract frequent itemsets that is a monotonous task. In this paper, we present a novel algorithm for Incremental Frequent Itemsets Mining (INFRIM) for updating data. The proposed algorithm uses previous mining results and generates not only frequent itemset patterns of incremental database but also updated database. The basic functionality of the proposed algorithm is envisaged as five step process. In the first step, frequent 1-itemset count is computed during preprocessing of incremental database. Second, it generates intermediate mining results from previous mining results and extracted 1-itemset counts of original and incremental database respectively. Third, based on support threshold the system will generate interesting frequent patterns to be studied from intermediate mining results. Fourth, scans incremental database and generates frequent itemsets. Fifth, update the frequent itemsets of updated database. The very salient feature of the proposed algorithm is to scan database once, avoids re-computing of large itemsets generation, dynamic update of previously extracted mining results. The experimental results of proposed algorithm is compared with Apriori and found higher speedup ratio. The proposed algorithm is highly suitable for Operational Business Intelligence systems because it provides timely update of mining results. The proposed algorithm not only supports incremental mining results in maintenance of data warehouse but also business intelligence projects.

Keywords—Association rules; data mining; frequent itemsets; frequent pattern; incremental mining; knowledge discovery; maintaining mining association rules;

I. INTRODUCTION

In recent years, data mining has attracted much attention in database research [6]. Association mining is one of the most important research problems in knowledge discovery and data mining areas. Association mining facilitates to extract association rule, frequent patterns in data which discovers strong associations or correlation relationships among data [20]. Most of the proposed algorithms in association mining focus on batch mining that do not permit full scan of database for every update and does not utilize previous mining results. In most of real world applications the organizational database grows continuously and is updated into data warehouse. Thus for each update demands to rerun of data mining algorithms inorder to main discovered information. Mining is a costly activity and needs to reduce full scan of database for each update. The concept of incremental mining was proposed [4] to maintain the rules of discovered during previous mining process. Incremental mining means to find delta from source, extract knowledge from delta and update previous mining results with extracted knowledge. The objective of incremental mining defined in [23] as to avoid re-learning of rules from the old data and utilize knowledge that has already been discovered.

In this paper, we propose a novel algorithm for Incremental Frequent Itemsets Mining (INFRIM) which extracts frequent itemsets for updating data. The data extraction from operational databases and update of original database are out of scope of this paper. It is assumed previous mining results are available in main memory as lookup to the proposed algorithm. The INFRIM algorithm will provide not only extracted mining results of incremental database but also updated database. During our experiments it is found that the INFRIM algorithm is highly suitable for maintenance of data warehouse and business intelligence projects. The proposed algorithm is highly scalable and suitable for large volumes of continuously growing databases.

The rest of the paper is organized as follows. Section 2 provides review of related work. Section 3 introduces various symbols being used, the structure and description of the proposed algorithm. Section 4 covers experiment and results. Section 5 covers discussion and salient features of the proposed algorithm. Section 6 covers conclusion of our work.
II. RELATED WORK

Abundant work has done in the field of association mining rules [1], [2], [3], [4], [5], [6], [31]. These efforts are directed at developing efficient and scalable algorithms to mine association rules and sequence patterns in large databases. Recent studies have shown that there are various incremental mining algorithms that have been developed for incremental mining frequent itemsets [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [22] in the literature. In this section, we discuss various incremental mining algorithms report recently in the literate and their functionality. The available incremental mining algorithms in the literature can be broadly classified into two groups known as Apriori and FP growth.

A. Apriori based Algorithms

In 1996, the problem of maintenance of association rules was first studied [5] and proposed Fast Update Algorithm (FUP). The framework of FUP is based on Apriori and adopts the pruning techniques used in the direct hashing pruning (DHP) [21]. FUP algorithm, first calculates large itemsets mainly from newly inserted transactions, and compares them with the previous large itemsets from the original database. According to the comparison results, FUP determines whether re-scanning the original database is needed, thus saving some time in maintaining the association rules. Although the FUP algorithm can indeed improve mining performance for incrementally growing databases, original databases still need to be scanned when necessary. FUP algorithm does not handle the case of deleting transaction from the database.

FUP2 algorithm was developed [7] to address the new rules in an update cases including insertion, deletion and modification of transactions. FUP2 is a complementary algorithm of FUP. FUP2 technique updates the association rules when old transactions are removed from the database and new transactions are added to it. This algorithm uses information available from a previous mining. New Fast Update (NFUP) algorithm proposed [22] for incremental mining of association rules that is based on Apriori. In this, the incremental database is divided into ‘n’ portions and each portion is called a partition. New frequent itemsets are generated from partition. The running time of NFUP rises almost in direct proportion with the transaction number of the incremental database. Accordingly, NFUP is suited frequently updated databases.

MAAP [20] is a low scan incremental association rule maintenance algorithm in the updated database. This algorithm is based on both the Apriori and FUP2 algorithms. This starts computing the high ‘n’ level large itemsets in the new database using the available high ‘n’ level large itemsets in the old database. Thus, it eliminates the need to compute parts of lower level large itemsets and saves rule maintenance time by reducing the number of times the database is scanned. It achieves more benefit when high level large itemsets can be used to generate a lot of low level large itemsets in the first step of applying the Apriori property.

B. FP Growth based Algorithms

Han et al proposed FP growth algorithm [2] to discover frequent patterns using FP tree without candidate generation. FP growth traverses the FP tree in a depth first manner. Since the inception of FP tree several incremental mining algorithms have been proposed [13], [17], [18], [19] in the literature that are based on FP growth.

Adjusting FP-Tree for Incremental Mining (AFPIM) algorithm proposed [18] that update previously constructed FP-tree. The adjusting FP tree contains frequent items based on user specified minimum support threshold minSup, by scanning only the incremental part of the dataset. As items are arranged in descending order of support count based on original dataset, AFPIM re-sorts the items according to new values of support count based on incremental dataset through bubble-sort. There are two major drawbacks of AFPIM algorithm: First, computational expensiveness of sorting process. Second, when new frequent patterns emerge, as a result of scanning of incremental dataset, AFPIM has to construct a new FP-Tree. An incremental approach to buffer the CSTree of the old database was proposed in [6]. This approach is capable of mining the closed frequent sequences. An incremental mining algorithm based on two support thresholds [7] that further reduces the need for rescanning original databases.

Compressed and Arranged Transaction Sequence Tree (CATS tree) [19] addresses the limitations of AFPIM algorithm. Unlike AFPIM, the CATS tree considers all the items in the transactions for representation into tree, regardless of whether items are frequent or not. This allows CATS tree to represent even new emerging frequent patterns from incremental dataset. CATS arrange the nodes based on their local support count, which helps to achieve high compactness of the tree. For incremental mining CATS tree updates the existing tree by considering the transactions of the incremental dataset one by one and merging them with existing tree branches. However, CATS tree has two limitations. First, for each new transaction it is required to find the right path for the new transaction to merge in. Second, it is required to swap and merge the nodes during the updates, as the nodes in CATS tree are locally sorted.

Canonical-order Tree (CanTree) algorithm proposed [17] by Leung et al and its construction is very much similar to CATS tree except that, in CanTree items are arranged according to some canonical order. The canonical order can be determined by the user prior to mining process. Canonical ordering can be lexicographic or based on certain property values of items. Based on this property CanTree can easily maintain when database
transactions are inserted, deleted, and/or modified. The CanTree does not require adjustment, merging, and/or splitting of tree nodes during maintenance. No rescan of the entire updated database or reconstruction of a new tree is needed for incremental updating. Further, though the CanTree takes less time for tree construction it requires more memory and more time for extracting frequent patterns from the generated CanTree.

Batch Incremental Tree (BIT) algorithm [13] to merge the small consecutive duration FP-tree to obtain a FP-tree that is equivalent of FP-tree obtained when the entire database is processed at once from the beginning of the first duration to the end of the second duration. BIT algorithm takes advantage of previously obtained periodical FP-tree, i.e., FP-tree representation of incremental transaction database, for incremental mining. BIT algorithm takes FP tree of the two periodic datasets. It then reads the itemsets of one of the FP-tree (T1) one by one along with their frequency counts and searches for the mergeable prefix path of the other FP-tree (T2). It then merges the itemset of T1 with the mergeable prefix by updating frequency count of the items and inserting remaining non-prefix items (if any) by extending the tree ranch after the last matching prefix item of the mergeable pattern.

III. ALGORITHM

In this section, we describe the various symbols used in this paper, development steps and functionality of the proposed algorithm.

A. Symbols and Notation

The various symbols used in this paper and their meaning are provided in table 1 below.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wdb</td>
<td>Original database</td>
</tr>
<tr>
<td>Idb</td>
<td>Incremental database</td>
</tr>
<tr>
<td>Udb</td>
<td>Updated database</td>
</tr>
<tr>
<td>n</td>
<td>Number of items</td>
</tr>
<tr>
<td>K</td>
<td>Number of itemset</td>
</tr>
<tr>
<td>Sk</td>
<td>Frequent K-itemset</td>
</tr>
<tr>
<td>(Kdb)w</td>
<td>Original database mining results</td>
</tr>
<tr>
<td>(Kdb)u</td>
<td>Intermediate mining results database</td>
</tr>
<tr>
<td>(Kdb)</td>
<td>Incremental database mining results</td>
</tr>
<tr>
<td>(Kdb)u</td>
<td>Updated database mining results</td>
</tr>
</tbody>
</table>

Original database (Wdb): This is most commonly referred as data warehouse. It is a central repository of data which is created by integrating data from multiple disparate sources. This stores historical data that is used for reporting and data analysis. This is commonly used for creating trend reports for senior management reporting for strategic and tactical level decision making. This acts single source of truth of an organization.

Incremental database (Idb): This is the data obtained from the operational systems that is transaction databases in a given window of time interval and the time interval will vary form case to case basis and typically ranges days, hours or even minutes. The extraction of delta from the operational data sources is out of scope of this paper. It is assumed that incremental database is available as one of the inputs to the algorithm.

Updated database (Udb): This database is a combination of original database (Wdb) and incremental database (Idb).

Knowledge database (Kdb): Inorder to extract the knowledge from database, mining algorithm to be run. The extracted knowledge contains the frequent patterns and frequent item counts which is known as knowledge database. The incremental mining system contains original (Kdb), intermediate (Kdb)int, incremental (Kdb) and updated (Kdb)u knowledge databases. Original knowledge database is the database which contains previous mining results of original database. Intermediate knowledge database is formed by the combination of frequent 1-item counts of incremental database and previous mining results of original database. From these intermediate mining results all frequent itemsets will be generated based on support threshold. The proposed incremental mining algorithm extracts knowledge from incremental database based on the itemsets to be generated and all other are pruned from extraction. In the proposed algorithm the previous mining details of original database are updated with incremental mining results that were obtained from incremental database and forms as updated mining results.

Incremental mining: The total functionality INFRIM is summarized as extraction of each transaction of incremental database, generated frequent itemsets dynamically and updated previous extracted knowledge database and as well as database of data warehouse. The output will be frequent itemsets.

B. Algorithms of Incremental Frequent Itemset Mining

The algorithm of proposed incremental frequent itemset mining is envisaged below from algorithm #1 to #6. The inputs to the algorithm are original database (Wdb), previous mining results of original database (Kdb), l-item count of incremental database. The outputs will be frequent itemsets of incremental database (Kdb) and frequent itemsets of updated database (Kdb). The overall functionality of the proposed algorithm is envisaged as “to extract the knowledge from delta and update previous mining results with extracted knowledge.”

Algorithm #1: Infrim: Incremental Frequent Itemsets Mining

Input: Incremental database (Idb), Previous mining results (Kdb)

Output: Frequent itemsets of Idb and Udb

1. begin
2. Extract_Mining_Results()
3. Frequent_Patterns_Under_Study()
4. Incremental_Mining()
5. Update_Database()
6. end
Algorithm #2: Extract_Mining_Results
Input: 1-itemset count of (Idb), Previous mining results (Kdb)
Output: Intermediate mining results
1. begin
2. get Transaction Count of Idb
3. get Transaction Count of Wdb
4. update Transaction Count of Udb
5. for each 1-item ε Idb do
6. get 1-item Count of Idb
7. get 1-item Count of Wdb
8. update 1-Item Count of Udb
9. end for
10. end

Algorithm #3: Frequent_Patterns_Under_Study
Input: 1-itemset count of (Idb), Previous mining results (Kdb)
Output: Intermediate mining results
1. begin
2. get Transaction Count of Idb
3. get Transaction Count of Wdb
4. update Transaction Count of Udb
5. for each 1-item ε Idb do
6. get 1-item Count of Idb
7. get 1-item Count of Wdb
8. update 1-Item Count of Udb
9. end for
10. end

Algorithm #4: Incremental Mining
Input: Incremental database (Idb), Frequent patterns under study
Output: Frequent itemsets of Idb
1. repeat
2. for each t ε Idb do
3. for each frequent pattern do
4. itemSetGen()
5. updateItemCount()
6. end for
7. end for

Algorithm #5: itemSetGen
Input: Itemset to be generated (K), Transaction of Idb.
Output: Specific Itemset
1. ksubsets = {}
2. k, n,
3. set ← {1, 2, 3, …, n}
4. S, getKSubsets ()
5. S = {}
6. generateKSubset (0,0,S)
7. return ksubsets
8. generateKSubset (start, index, ksubset)
9. if (index = k)
10. ksubsets.add (ksubset)
11. else
12. generateKSubset (start +1, index+1, ksubset)

Algorithm #6: updateDatabase (Udb)
Input: Mining results of Idb (Kdb)_I, Previous mining results (Kdb)_W
Output: Frequent itemsets of Idb and Udb
1. repeat
2. for each itemset count of Idb do
3. get item count of (Kdb)_I
4. get item count of (Kdb)_W
5. update item count of (Kdb)_W
6. end for
7. until (Kdb)_I = φ

C. Functionality of Algorithm

The functionality of the proposed algorithm is shown in figure 1 and summarized as five step process. The description of each step is outlined in the subsequent paragraph.

1) Data pre-processing: The extracted data from source systems is available as incremental database. This incremental database is pre-processed and then transformed into a binary vector form. Each row corresponds to a transaction and each column corresponds to an item. An item can be treated as a binary variable whose value is one if the item is present in a transaction and zero otherwise. During this data pre-processing phase count of 1-itemsets are computed.

2) Mining results extraction: In this step, 1-itemset count of incremental database are combined with previous mining frequent itemsets count of original database and forms as intermediate mining results. In addition to this it will also update parameter like transaction counts and support threshold during this phase. Finally, the output of this phase will be an intermediate frequent patterns knowledge database (Kdb)_W.

3) Frequent patterns under study: In this step, as per the user defined suppport count threshold, system will check and generate frequent items to be required for extraction from incremental database.

4) Scanning of incremental database: During this step it scans incremental database transaction by transaction, generates frequent itemsets with their count. It prunes all unwanted frequent patterns that are below support threshold (minimum support) during knowledge extraction. Finally, it generates all frequent itemsets of incremental database.

5) Mining results update: Frequent itemsets results obtained from incremental database are updated with previous mining results of original database and forms as updated frequent mining results of updated database. The outputs contain mining results of both incremental database as well as update database.

III. EXPERIMENTS AND RESULTS

In this section we describe experimental setup, datasets being used, and performance results of the proposed algorithm.
A. Setup

We conducted a set of experiments to compare the performance of the proposed incremental algorithm. The experiments were conducted on a Dual Core CPU with 2.10 GHz configured with 2GB main memory and running Windows 7 Professional. The proposed algorithms are written in Java programming language.

![Functionality of INFRIM (Incremental frequent itemset mining) algorithm](image1)

Fig. 1. Functionality of INFRIM (Incremental frequent itemset mining) algorithm

![Bakery Database Performance Comparison](image2a)

![Retail Store Database Performance Comparison](image2b)

![DS#07 of Retail Store Database](image2c)

Fig. 2. Performance comparison Infrim with Apriori

B. Datasets

Experiments were performed on the following two synthetic databases known as Bakery and Retail stores. The specifications of bakery database consist of four datasets whose specifications are given in table 2.

<table>
<thead>
<tr>
<th>Name of dataset</th>
<th>No. of transactions</th>
<th>Size</th>
<th>No. of items</th>
<th>Average items</th>
<th>Max. no. of items</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS#01</td>
<td>1000</td>
<td>99 KB</td>
<td>50</td>
<td>3.538</td>
<td>8</td>
</tr>
<tr>
<td>DS#02</td>
<td>5000</td>
<td>494 KB</td>
<td>50</td>
<td>3.547</td>
<td>8</td>
</tr>
<tr>
<td>DS#03</td>
<td>20000</td>
<td>1.97MB</td>
<td>50</td>
<td>3.556</td>
<td>8</td>
</tr>
<tr>
<td>DS#04</td>
<td>75000</td>
<td>7.40MB</td>
<td>50</td>
<td>3.549</td>
<td>8</td>
</tr>
</tbody>
</table>

The specifications of retail store database are provided in table 3 that contains seven different datasets.

<table>
<thead>
<tr>
<th>Name of dataset</th>
<th>No. of transactions</th>
<th>Size</th>
<th>No. of items</th>
<th>Average items</th>
<th>Max. no. of items</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS#01</td>
<td>1000</td>
<td>41 KB</td>
<td>20</td>
<td>1.83</td>
<td>6</td>
</tr>
<tr>
<td>DS#02</td>
<td>5000</td>
<td>201 KB</td>
<td>20</td>
<td>2.43</td>
<td>9</td>
</tr>
<tr>
<td>DS#03</td>
<td>30K</td>
<td>1.20MB</td>
<td>20</td>
<td>2.77</td>
<td>9</td>
</tr>
<tr>
<td>DS#04</td>
<td>100K</td>
<td>4.01MB</td>
<td>20</td>
<td>1.82</td>
<td>6</td>
</tr>
<tr>
<td>DS#05</td>
<td>500K</td>
<td>20.02MB</td>
<td>20</td>
<td>2.03</td>
<td>11</td>
</tr>
</tbody>
</table>

![Performance comparison Infrim with Apriori](image3)
The performance of INFRIM algorithm was compared with Apriori [1]. The Apriori was run the database (Wdb) transactions and also on (Idb). The speedup ratio was computed as described in [11]. In our experiments, the speedup ratio ranges from 1.2 to 3.1.

C. Results
The proposed algorithm, INFRIM, is simple in structure and generates frequent itemsets from the intermediate mining results which is obtained from previous mining results and extracted results of incremental database. We conducted testing of proposed algorithm on synthetic databases of various sizes whose specifications are provided in table 1 and 2. The performance of the proposed algorithm is compared with Apriori and few experimental results are provided in figure 2(a) to 2(b). Figure 2(c) shows the speedup ratio of the proposed algorithm with Apriori for dataset (DS#07) of retail store database as specified in table 3.

IV. DISCUSSION AND SAILENT FEATURES
The proposed algorithm is simple in structure which dynamically generates frequent itemsets from incremental database. Inorder to find the performance of the proposed algorithm experiments were conducted on two different databases whose transactions ranges from several thousands to millions. Figure 2(a) shows performance comparison of the proposed algorithm with Apriori of bakery for itemset 2 of bakery database. It is clearly seen that the amount out of time required for the proposed algorithm is much lower. Similarly, figure 2(b) shows the performance comparison for itemset 2 of retail database. From these two graphs it is noticed that as transaction volume increased the run time decreases as compared to Apriori. Figure 2(c) shows the speedup ratio of the proposed algorithm with Apriori for itemsets 2 to 5 and found highly scalable and efficient for even large datasets. During our experiments it is found that the speedup ratio ranges from 1.2 to 3.1. The proposed algorithm not only provides timely knowledge to various operational users of business organization for their decision making but also supports incremental mining in maintenance of data warehouse and business intelligence projects.

The proposed algorithm has the following salient features: It uses previous mining results of original database there by it avoids rescanning of original database. It works dynamically. Each transaction of incremental database is processed and then the corresponding itemsets will be updated in knowledge database which includes itemset counts, itemsets, transactions count and updated database. It avoids re-computing large itemsets that have already been discovered. In the proposed algorithm there is no need of multiple scan or even re-scan of the database for computing large itemsets because it uses previous mining results and scans database only once. The proposed algorithm focuses on newly transactions obtained from incremental database. Thus, it greatly reduces the number of candidate itemsets generation which intern increases computation time. As a result it improves the efficiency of the proposed algorithm. It uses a simple check of 1-itemset support count of extracted knowledge database before to update the frequent item counts and then generates dynamically the required frequent itemsets from each transaction of incremental database. This results optimization of the itemsets generation by eliminating which in turn saves computation time. Keeping all the above features of the proposed incremental frequent Itemset mining algorithm for updating records is highly scalable for any large incremental databases. This is highly suitable for operational business intelligence systems which provide miniscule details of frequent itemsets information for low level decision making and as well as strategic and tactical level decision making to the organizations.

V. CONCLUSION
We presented a novel algorithm for dynamic Incremental Frequent Itemsets Mining for maintenance of frequent pattern mining. This algorithm is named as INFRIM. This algorithm is updates the extracted knowledge database with dynamically generated frequent itemsets from the incremental transaction database. It uses the knowledge available from the previous mining to reduce the amount of work that has to be done to discover the frequent patterns in the updated database of data warehouse. The performance of INFRIM is significantly faster than mining the updated database of original database from scratch. The proposed algorithm has the following salient features: it uses previous mining results, works dynamically, avoids re-computing large itemsets that have already been discovered, Focuses on newly inserted transactions, thus greatly reducing the number of candidate itemsets which intern reduces computation time and increases execution speed of algorithm, uses a simple check of 1-itemset support count of incremental database and then generates the required frequent itemsets to be studied that reduces the computation time of frequent itemset generation.

The proposed algorithm, INFRIM, works well over wide ranges of system parameter values in terms of size of incremental database, number of transactions in incremental transaction database, size of original database, size of knowledge database. In particular, it works well for updates of a wide range of insertion sizes and small to moderate deletion sizes. The INFRIM is not only providing knowledge from time to time to the operational users of business organization for their decision making but also supports incremental mining in maintenance of data warehouse and business intelligence projects. Further researches could extend this work to problems of
various minimum supports and problems of generalized sequential patterns.

REFERENCES


[14] Ding Yuan, Kyuhyung Lee, Hong Cheng, Gopal Krishna, Zhenmin Li,Xiao Ma, Yuanjuan Zhou and Jiawei Han, “ClSpan: Comprehensive Incremental Mining Algorithms of Closed Sequential Patterns for Multi-Versional Software Mining,” In Pr. of the SIAM Int. Conference on Data Mining, SDM 2008, April 24-26, 2008, pp. 84-95.


An Incremental Mining Algorithm for $K^{th}$ Frequent Itemset

A.D.N.Sarma
Research Scholar, Dept. of CSE
Acharya Nagarjuna University
Guntur- 522 510, Andhra Pradesh, India
adnsarma@yahoo.com

Dr. R.Siva Rama Prasad
Research Director, Dept. of CSE, Coordinator - IBS
Acharya Nagarjuna University
Guntur- 522 510, Andhra Pradesh, India
raminenisivaram@yahoo.co.in

ABSTRACT
Frequent Itemsets mining is an important research problem in knowledge discovery and data mining areas. Most of the research is reported on developing highly scalable and efficient algorithms for frequent itemset mining. There is very limited work has been reported in the literature on incremental mining. In this paper, we present an incremental mining algorithm for generating $K^{th}$ frequent itemset for update case. This algorithm uses simple binary principles for generating $K^{th}$ item subsets, where $K > 1$, directly without generating lower itemsets. The inputs to the proposed algorithm are binary representation of transactional data and the value of itemset ($K$) to be generated. The output of the algorithm is $K^{th}$ frequent itemset patterns and their count. In addition to this the functionality of typical incremental mining system is briefly explained. The experimental results of the proposed algorithm are compared with Apriori and found satisfactory performance. The proposed algorithm provides timely update of mining results to the operational users of an organization for their decision making that supports in maintenance of data warehouse and business intelligence projects.

Categories and Subject Descriptors
H.2.8 [Database Applications]:: Data mining

General Terms
Algorithms, Performance.

Keywords
Association rules, data mining, frequent itemsets, frequent pattern, incremental mining, frequent $K^{th}$ itemset, knowledge discovery, maintaining mining association rules.

1. INTRODUCTION
Association mining is one of the most important research problems in knowledge discovery and data mining areas. Abundant research has been reported in the field of association mining rules [1], [2], [3], [4], [5], [6]. Most of the research has confined to find all frequent itemsets, maximal frequent itemsets, closed frequent itemsets, Top - K frequent itemsets and Top - K closed itemset and most of these algorithms are developed based on the concepts of Apriori [1], [2], [4]; FP growth [3] or sampling database [6], [28], [29]. These efforts have been directed at developing an efficient and highly scalable algorithms to mine association rules and sequence patterns in large databases and these algorithms are commonly require multiple scans of database.

In a typical data mining process, full data is rarely collected in single attempt. The collection of data is a continuous and ongoing process. In typical business operations, the data collection time will be as small as in weeks, days, hours, minutes and even seconds from operational data sources. The data collection time mostly depends on the nature of the business and business value. In many cases, data collection is carried out in phases or batches. Consequently, the content of the underlying database changes over time. In order to study the behavior of frequent itemset patterns in continuously growing databases, mining algorithms need to run iteratively or repeatedly. In simple case the mining algorithm has to run on full database that is a monotonous task. Alternatively the mining algorithm that uses previous mining results before to run on incremental database which is known as incremental mining that was first proposed [7]. The concept of incremental mining includes not only to update, insertion and deletion operations on database but also on extracted knowledge to get timely update of frequent patterns. In most of the business applications database grows continuously. New items will include in the catalogs and few items will become obsolete over a period. Hence, there is a need to mine database again to understand association rules that includes earlier discovered rules and also for discovering new rules. According to [13] applying mining algorithms on the updated database (the older database plus the incremental database) may be too costly.

In real time business scenario the users are interested in specific frequent itemset patterns instead of finding all frequent itemset patterns. Most of the existing algorithms in the literature will find all frequent itemsets patterns from transaction database which start from one itemset generation to the specific itemset. So, the generation of specific frequent itemset in continuously growing databases is found to be an open research problem. This motivates us to develop an incremental mining algorithm for $K^{th}$ frequent itemset generation. The objective of this paper is to develop an efficient and scalable incremental mining algorithm for generating $K^{th}$ frequent itemset, where $K > 1$, from the given transactional database directly without generating any previous frequent itemsets. The proposed algorithm uses simple binary concepts for generating $K^{th}$ frequent sub-itemsets.

The rest of the paper is organized as follows. Section 2 provides review of related work. Section 3 covers typical incremental mining system functionality and the proposed incremental mining algorithm for $K^{th}$ frequent itemset. Section 4 covers experiments and result. In Section 5 conclusion of our work.

2. RELATED WORK
Much of the work has reported in the literature on association mining rules [1], [2], [3], [4], [5], [6], [28], [30], [36]. These efforts have been focused at developing efficient and scalable algorithms to mine association rules and sequence patterns in large databases. A few efficient algorithms for mining frequent sequences have been proposed in the literature AprioriAll [2], GSP [2], SPADE [30], MFS [31], and PrefixSpan [32]. A simple
approach to the update problem is to mine the new database from scratch, using sequence mining algorithm. However, this approach does not use previous mining results. In order to utilize the previous mining results, several incremental mining algorithms have been proposed which include GSP+ [31], MFS+ [31], and ISM [18]. The above three algorithms have been based on GSP, MFS, and SPADe respectively. The structures of GSP+ and MFS+ follow those of GSP and MFS. Recent studies have been found various incremental mining algorithms [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [29] in the literature. In this section, we discuss different types of incremental mining algorithms that are reported in the literature recently and their functionality. The available incremental mining algorithms can be broadly classified into two major groups known as Apriori and FP growth.

2.1 Apriori based Algorithms

In 1996, the problem of maintenance of association rules was first studied [7] and proposed Fast Update Algorithm (FUP). The framework of FUP is based on Apriori that adopts the pruning techniques used in the direct hashing pruning (DHP) [36]. FUP algorithm first calculates large itemsets mainly from newly inserted transactions, and compares them with the previous large itemsets from the original database. According to the comparison results, FUP determines whether re-scanning the original database is needed, thus saving some time in maintaining the association rules. Although the FUP algorithm can indeed to improve mining performance for incrementally growing databases, original databases still need to be scanned when necessary. FUP algorithm does not handle the case of deleting transaction from the database.

A new algorithm FUP2 was developed [8] further to address the new rules in an update cases including insertion, deletion and modification of transactions. FUP2 is a complementary algorithm of FUP. FUP2 technique updates the association rules when old transactions are removed from the database and new transactions are added to it. This algorithm uses information available from a previous mining. An incremental mining algorithm was proposed [9] based on the concept of pre-larger itemsets based on two support thresholds which avoids the movements of itemsets directly from large to small and vice versa. Further this algorithm reduces the need for rescanning original database.

MAAP algorithm was developed [34], for incrementally maintaining association rules in the updated database. This algorithm applies an Apriori property to the set of large itemsets in the old database, generates some parts of the lower level large itemsets in the new database. This algorithm uses all previous old large itemsets and that are confirmed to be still large in the new database. Thus, MAAP algorithm eliminates the need to compute parts of lower level large itemsets that saves rule maintenance time by reducing scanning of database number of times. It achieves more benefit when high level large itemsets can be used to generate a lot of low level large itemsets in the first step of applying the Apriori property.

The New Fast Update algorithm (NFUP) was proposed [15] for incremental mining which is Apriori like algorithm. In this the incremental database is logically divided into $n$ partitions according to unit time interval. The latest information is at the last partition of incremental database. Therefore, NFUP scans each partition backward which is suitable for frequently updated database. NFUP does not require scanning of the original database.

2.2 FP Growth based Algorithms

Han et al. was proposed FP growth algorithm [3] to discover frequent patterns using FP tree without candidate generation. FP growth traverses the FP tree in a depth first manner. Since the inception of FP tree several incremental mining algorithms have been proposed [15], [24], [25], [26] in the literature that are based on FP growth. An efficient algorithm Incremental Mining of Closed Sequential Patterns (IMCS) was developed [16] to maintain the Closed Sequence Tree (CSTree) when the sequence database is updated incrementally which is capable of mining the closed frequent sequences.

Adjusting FP-Tree for Incremental Mining (AFPIM) algorithm was proposed [25] which updates previously constructed FP-tree. The adjusting FP tree contains frequent items based on user specified minimum support threshold by scanning only the incremental part of the dataset. As items are arranged in descending order of support count based on incremental dataset, AFPIM re Sorts the items according to new values of support count based on incremental dataset through bubble-sort. There are two major drawbacks of AFPIM algorithm: First, computational expensiveness of sorting process. Second, when new frequent patterns emerge, as a result of scanning of incremental database, AFPIM has to construct a new FP-Tree. Compressed and Arranged Transaction Sequence Tree (CATS tree) [26] addresses the limitations of AFPIM algorithm. Unlike AFPIM, the CATS tree considers all the items in the transactions for representation into tree, regardless of whether items are frequent or not. This allows CATS tree to represent even new emerging frequent patterns from incremental dataset. CATS algorithm arranges the nodes based on their local support count, which helps to achieve high compactness of the tree. For incremental mining CATS tree updates the existing tree by considering the transactions of the incremental dataset one by one and merging them with existing tree branches. However, CATS tree too has two limitations. First, for each new transaction, it is required to find the right path for the new transaction to merge in. Second, it is required to swap and merge the nodes during the updates as the nodes in CATS tree are locally sorted.

Canonical-order Tree (CanTree) algorithm was proposed [24] by Leung et al. Construction of CanTree is very much similar to CATS tree except that, in CanTree items are arranged according to some canonical order. The canonical order can be determined by the user prior to mining process. Canonical ordering can be lexicographic or based on certain property values of items. Since the canonical order is fixed and not based on the support count, CanTree allows easy insertion of nodes. Unlike the CATS Tree, transaction insertions in CanTree require no extensive searching of mergeable paths. CanTree too has some limitations. It generates compact tree if and only if majority of the transactions contain common pattern-base in canonical order. It generates skewed tree with too many branches. Further, though the CanTree takes less time for tree construction, it requires more memory and more time for extracting frequent patterns from the generated CanTree.

Batch Incremental Tree (BIT) algorithm [15] uses to merge the small consecutive duration of FP-trees to obtain a FP-tree. The resultant FP tree is equivalent of FP-tree obtained when the entire database is processed at once from the beginning of the first
duration to the end of the second duration. BIT algorithm takes the advantage of previously obtained periodical FP-tree. BIT algorithm takes FP tree of the two periodic datasets. This reads the itemsets of one of the FP-tree (T1) one by one along with their frequency counts and searches for the mergeable prefix path of the other FP-tree (T2). The itemset of T1 merges with the mergeable prefix by updating frequency count of the items and inserting remaining non-prefix items (if any) by extending the tree branch after the last matching prefix item of the mergeable pattern. BIT algorithm takes less time for tree construction as compared to normal FP tree.

3. INCREMENTAL MINING SYSTEM

Data mining is a collection of tools and technology that helps to analyze datasets, find patterns and relationships in the data. The tools will find patterns in the data with the help of algorithms whereas technology helps to build efficient tools. Incremental mining system or incremental data mining (IMS) will help to analyze more importantly operational data that provides the corresponding changes in current and as well as historical data in terms of insertion, deletion and update operations. The users of an IMS will get timely update of knowledge for their decision making. The IMS will also support for ongoing maintenance of data warehouse or business intelligence applications. The objective of incremental mining was defined [29] to avoid re-learning of rules from the old data and utilize knowledge that has already been discovered. In this section we briefly describe the functionality of typical incremental mining system.

3.1 Typical Incremental Mining System (IMS)

A typical incremental mining system (IMS) is shown in figure 1. The major functional blocks of IMS system are: operational data sources, incremental database, data warehouse, knowledge database, metadata, incremental mining and mining algorithms, reporting and query tools and user access layer.

![Figure 1. Typical Incremental Mining System](image)

3.1.1 Operational Data Sources (Odb)

Operational Data Sources (ODDB): This includes various data sources and applications in the business organization. These data sources will contain day to day business transactions that generally include ERP, CRM, SCM and legacy.

3.1.2 Incremental database (Idb)

This is the data obtained from the operational systems that is transaction databases in a given window of time interval. The data collection time is as small as days, hours or even minutes that depends on the type of business and business value to be measured. The extraction process of delta from the operational data sources is out of scope of this paper. It is assumed that incremental database is available as one of the inputs to the proposed algorithm. The cost of maintaining the rules can be reduced considerably once we know the type of incremental database that can be visualized as three types according to [13] for incremental mining of association rules. (1) The incremental database can be considered as a sample of the original database. In this case, there are no significant differences between the patterns in the original and the updated databases. (2) The incremental database has more or less similar patterns to that of the original database. Essentially the original patterns may still exist but there may be a few new patterns. (3) The incremental database may not necessarily be a good sample of the original database. The patterns found in the incremental and the original database may be entirely different.

3.1.3 Data Warehouse Database (Wdb)

Data warehouse database is a central repository of data which is created by integrating data from multiple disparate sources. This stores current as well as historical data. It is single source of truth of an organization. This database is used for reporting and data analysis purposes for creating trend reports for senior management reporting that includes annual and quarterly comparisons which is used for strategic and tactical level decision making.

3.1.4 Updated Database (Udb)

This database is a combination of original database and incremental database.

3.1.5 Knowledge database (Kdb)

Mining algorithms has to run first time on original database to extract knowledge. This extracted knowledge contains the frequent patterns which are known as previous mining results or knowledge database of original database. The proposed algorithm has to run on the incremental database to extracts the knowledge which is known as incremental knowledge or incremental mining results. These incremental mining results are updated with previous mining results which form as an updated knowledge database. Thus knowledge database contains multiple versions of mining results (or extracted patterns) of incremental and updated databases. So, the users of an IMS can access not only incremental mining results but also updated mining results which provide good visibility of frequent patterns in both short and longer time periods. This inturn gives more business value to the users of the system for their effective decision making.

3.1.6 Incremental Mining and Mining Algorithms

This functional block contains various mining algorithms as well as incremental mining algorithms. The proposed algorithm will reside in this functional block which extracts knowledge from incremental database that uses previously mining results. The extracted knowledge is updated with already available knowledge or previous mining results. So the users of the system can access both incremental knowledge as well as updated knowledge for all the levels of decision making.

3.1.7 Query and Reporting Tools

A good IMS system should include efficient query and reporting modules that include functionalities: ad-hoc querying and reporting dimensional and non-dimensional data, dashboards, and self-configurable reporting. The end user can access frequent
patterns from the updated knowledge database with the help of reporting and query tool. These tools include functionalities for extract, sort, summarize, and present selected data to the user in suitable visual form. In addition to this these tools will provide an additional feature including alerts for presenting the right information from incremental and as well as updated knowledge to the right people at the right time for right decision making.

3.1.8 Metadata
Metadata is one of the most important functional blocks in IMS as similar to data warehouse and business intelligence systems. It describes data of data and is often used to control the handling of data and describes: Rules, Transformations, Aggregations, and Mappings.

3.2 ALGORITHM
The proposed algorithm of incremental Kth frequent itemsets mining is envisaged below in algorithm #1 and #2. The inputs to the algorithm are binary representation of transactional database and the output is Kth frequent itemsets and their count. The proposed algorithm generates frequent itemset, where K > 1, from transactional database directly without generating lower frequent itemsets.

Algorithm #1: Incremental Mining for Kth Frequent Itemset
Inputs: Transaction database and itemset to be generated (K)
Output: Kth Frequent itemsets generation
1. Transaction database $Idb = \{t_1,t_2, ... ,t_N\}$
2. repeat
3. for each transaction in $Idb$ do
4. \hspace{1em} $t_i$ = itemsInTransaction ($t_i$)
5. \hspace{1em} if (length of $t_i$ >= K)
6. \hspace{1em} $L_k$ = ItemsetGen ($t_i$)
7. \hspace{1em} for each itemset element in $L_k$ do
8. \hspace{2em} Item count ++
9. \hspace{1em} end for
10. end if
11. end for
12. until $Idb = \emptyset$

Algorithm #2: Itemset Generation
Inputs: Itemset to be generated (K)
Output: Kth Frequent itemsets generation
1. Declare an vector of $items$
2. Declare $loopcounter$
3. Initialize the variable $K$ itemset
4. for 1 to $loopcounter$
5. \hspace{1em} Generate binary of $loopcounter$
6. \hspace{1em} generate subsets of size $k$
7. \hspace{1em} Add itemset to $items$ vector
8. \hspace{1em} end if
9. end for
10. return $items$ vector
11. end for

3.2.1 Binary representation of transaction data
The input to proposed algorithm is a binary vector format of transactional data as shown in Table 1. Each record represents a transaction with transaction identification (TID) and each column represents an item (I) in the table. Each row contains only 1 and 0s. Binary 1 in the transaction represents the presence of an item and 0 represents the absence of an item.

### Table 1. Binary representation of transactional data

<table>
<thead>
<tr>
<th>TID</th>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$I_3$</th>
<th>$I_4$</th>
<th>$I_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

3.2.2 Working of Algorithm
The inputs to the algorithm #1 are incremental database, previous mining results of original database and the value of frequent itemset (K) is to be determined. It is assumed that the previous mining results are available to the proposed algorithm in main memory as frequent itemsets and their count.

The algorithm #1 reads each transaction of incremental database and then finds the number of items present in a transaction called item count ($t_i$). If the length of item count is more than or equal to the itemset (K) to be find then generates all possible sub itemsets otherwise transaction is pruned. Itemset generation is obtained by the use of an algorithm #2. This algorithm uses simple binary principles to generate all possible sub-itemsets directly without generating lower itemsets. Finally, the count of each itemset will be incremented. This process will repeat till the end of incremental database.

3.2.3 Output of Algorithm
The output of algorithm consists of all frequent Kth itemsets and their count of incremental database. It is assumed previous mining results are available in memory and will be updated with incremental mining results inorder to get updated frequent itemsets. The discussion on update mining results and updated database are out of scope of this paper.

4. EXPERIMENTS AND RESULTS
In this section we describe the experimental setup, datasets being used, and performance results of the proposed algorithm.

4.1 Experiments
We conducted a set of experiments to compare the performance of the proposed incremental mining algorithm. The experiments were conducted on Dual Core CPU with 2.10 GHz configured with a 2GB main memory and running Windows 7 Professional. The proposed algorithms are written in Java programming language. The experiments are compared with Apriori algorithm. We tested the algorithms by measuring runtime against varying size of databases of original and incremental databases in fixed propositions (40:60) and keeping the total database size fixed. The size of incremental database is varied in steps of 10% of total database.

4.2 Datasets
We conducted testing of the proposed algorithm on different synthetic datasets. The specifications of these data sets are provided in table 2.

### Table 2. Specifications of datasets

<table>
<thead>
<tr>
<th>Name of dataset</th>
<th>No. of transactions</th>
<th>Size, MB</th>
<th>No. of items</th>
<th>Average items</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS#01</td>
<td>1M</td>
<td>10.773</td>
<td>5</td>
<td>2.562</td>
</tr>
<tr>
<td>DS#02</td>
<td>1M</td>
<td>14.649</td>
<td>7</td>
<td>3.593</td>
</tr>
<tr>
<td>DS#03</td>
<td>1M</td>
<td>20.508</td>
<td>10</td>
<td>5.155</td>
</tr>
</tbody>
</table>
4.3 Results and Discussion

The proposed algorithm is simple in structure that generates $K^{th}$ frequent itemsets directly from the given incremental database for update case. The algorithm uses simple binary concepts for generation of item sub-sets from each transaction. The performance of the proposed algorithm is compared with Apriori [1] and few results of experiments are presented in the form of the graphs in figure 2(a) to 2(d). These figures clearly show that the proposed algorithm takes less runtime as compared with Apriori for different itemset values ranges from 2 to 5.

For higher value of itemsets the amount of runtime required for the proposed algorithm is as low as compared with Apriori. The support count of Apriori was considered as 0.01(1%) for these experiments. The performance of the proposed algorithm is compared with Apriori and found higher performance and scans database only once. From our experimental results it found that the proposed algorithm is efficient as compared with Apriori and is linearly scalable with database size.

The speedup ratio ($s$) was calculated as described [13] and is given below. The value of $s$ increases as increasing itemset ($K$) that ranges between 2.35 and 11.07 during our experimental results.

$$s = \frac{(\text{time for Apriori})_{Wdb} + (\text{time for Apriori})_{Idb}}{(\text{time for proposed algorithm})_{Wdb} + (\text{time for proposed algorithm})_{Idb}}$$

The proposed algorithm supports incremental mining in maintenance of data warehouse and business intelligence projects. This algorithm provides intelligence (or frequent patterns) in both short and as well as long time periods that in turn gives more business value to the users of business organization.

5. CONCLUSIONS

We presented an incremental mining algorithm for $K^{th}$ frequent itemset for update database. The proposed algorithm directly generates the required $K^{th}$ frequent itemsets from the given transactional database without generating any lower values of frequent itemsets. The proposed algorithm uses simple binary concepts for generating $K^{th}$ frequent itemsets for update case. The previous mining results of original database are updated with the extracted knowledge inorder to provide updated knowledge database.

We conclude that the proposed algorithm runs significantly faster than mining the updated database from scratch. The performance of the proposed algorithm is higher as compared with Apriori and found satisfactory results over wide range of transactions that is linearly scalable with database size. The speedup ratio ranges between 2.35 and 11.07 during our experimental results. This algorithm not only provides timely update of mining results to the operational users of an organization for their decision making on continuously growing database but also supports in maintenance of data warehouse and business intelligence projects. We are working currently to extend this work for large databases having large number of items. In the future, the authors will consider the extension of the proposed algorithm for insertion and as well deletion cases of transactions. Further researches could extend this work to problems of various minimum supports.

6. REFERENCES


Major Components of Operational Business Intelligence System

A.D.N.Sarma¹
Research Scholar, Department of CSE

Dr. R.Siva Rama Prasad²
Research Director, Department of CSE, Coordinator - IBS
¹,²Acharya Nagarjuna University
Guntur- 522 510, Andhra Pradesh, India
¹adnsarma@yahoo.com, ²raminenisivaram@yahoo.co.in

Abstract: Business intelligence (BI) becomes more popular nowadays and become a major part of the decision making system for all most all business organizations. Classical business intelligence systems are static, historic in nature and confined to strategic and tactical users only whereas modern business intelligence becomes dynamic, current, event driven and supports not only for strategic users but also tactical and even operational level users for their decision making. The modern business intelligence systems are more popularly known as dynamic business intelligence or Operational business intelligence (Operational BI). In this paper, we present the major components of operational business intelligence system. Further, the proposed components of Operational BI system are divided into various sub-components and explained briefly the functionality of these components. The major components of Operational BI system are business process monitoring and management, event notification and management, operational analytics, operational reporting and portal. Business process monitoring and management contains various sub-components that includes business rules engine, business process monitoring, process measurements and Service Level Agreements (SLAs) management. Event analytics, incremental mining and streaming SQL are few major sub-components of Operational analytics. How the proposed components of operational business intelligence system constitute business value to the organization in terms of reduced action time is envisaged. The very key feature of Operational BI system is not only to provide operational intelligence to the operational users but also tactical and strategic users for their timely decision making.

Keywords: Business intelligence, decision making tool, low level decision making, Operational business intelligence, Operational intelligence, OLAP and OLTP.
1. Introduction

A business intelligence system was first proposed by Hans Peter Luhn (1958). He defined intelligence as “the ability to apprehend the interrelationships of presented facts in such a way as to guide action towards a desired goal.” Business intelligence (BI) becomes more popular nowadays and become a major part of the decision making system for all most all business organizations. Classical business intelligence systems are static, historic in nature and confined to strategic and tactical users only whereas modern business intelligence becomes dynamic, current, event driven and supports not only for strategic users but also tactical and even operational level users for their decision making. The modern business intelligence systems are more popularly known as dynamic business intelligence or Operational business intelligence (Operational BI). The use of Operational BI is to help, drive and optimize business operations on a daily basis, and, in some cases, even for intraday decision making Hans Peter (1958). Operational BI represents a turning point in the evolution of BI Liping et al (2007). In recent years, Operational Business Intelligence has emerged as an important trend in the business intelligence market Aberdeen (2007). Operational BI merges analytical and operational processes into a unified whole Eckerson (2007).

Operational BI system speed is critical and focuses Claudia Imhoff (2006) to provide real-time monitoring of business processes. Figure 1 shows action time versus business value. The traditional BI system can be transformed into an Operational BI system by reducing action time.

![Figure 1: Action time versus business value](image-url)
The action time is defined as the time interval between occurrence of business event and action that is shown in figure 1. There are four different factors involved in action distance. The action time is the algebraic sum of data latency, analysis latency, decision latency and response latency.

1. **Data latency** is the time from the occurrence of the business event until the data is stored and ready for analysis.

2. **Analysis latency** is the time from the point when data is available for analysis to the time when information is generated out. It includes the time to determine root causes of business situations.

3. **Decision latency** is the time it takes from the delivery of the information to selecting a strategy in order to change the business environment. This type of latency mostly depends on the time the decision makers need to decide on the most appropriate actions for a response to the business environment.

4. **Response latency** is the time needed to take an action based on the decision made and to monitor its outcome. This includes communicating the decision made as a command or suggestion, or executing a business action in a target system.

Reduction of action time in the system can be achieved by introducing real time integration between operational data sources and Operational BI engines and by introducing new components in the system.

### 1.1 Aim of the Paper

The aim of this paper is to identify the major components of operational business intelligence system based on improved business value and reduced action time. These identified components are further decomposed into various sub-components that constitute the full system. The functionality of the proposed components of Operational BI system is explained. Envisaged how the proposed components of Operational BI system improves business value to the organization internos of reduced action time. The major components of Operational BI system are business process monitoring and management, event notification and management, operational analytics, operational reporting and portal. In this paper the components of classical BI system are out of the scope of discussion.
The rest of the paper is organized as follows. Section 2 provides related work. In section 3 the major components of Operational BI system are presented and explain their functionally. Section 4 covers discussion of the proposed components, will improve business value to the organization. In Section 5 conclusion of our work.

2. Related Work

Operational BI is an emerging concept in BI domain. There is very limited work has been reported in this area. Operational BI has gaining popularity in the recent past in all most all business domains because of increased business value in reduced action time. A bidirectional communication between the operational systems and the analytical applications was described Daniela Ioana (2008) for an Operational BI system. The history of the informatics system for managed is marked by 3 key moments described Daniela Ioana (2008). The first key moment is an operational transaction system (OLTP). The second key moment is analytical applications that include data warehouse (DW), business intelligence (BI) and Business process management (BPM). The third key moment as Operational BI systems that is a combination of first and second.

Identified Aberdeen (2007) six "flavors" of operational business intelligence: 1) transactional BI with analysis and reporting, 2) real-time analytics with business rules applied to data as it is captured, 3) near real-time analytics business rules, 4) operational reporting, 5) business activity monitoring or business process monitoring, and 6) decision management based on business rules with integrated reporting and analytic applications.

The purpose of Business Process Monitoring (BPM) in BI architecture described Matteo Golfarelli et al (2004) for monitoring time critical operational process to quantify the enterprise strategy. An architecture for real time analysis proposed Seufert et al (2005) with the aim of reducing the action time and thereby increasing the value of business intelligence. This architecture uses information infrastructure and business integration infrastructure. The information infrastructure is responsible for managing data for business intelligence purpose and offers data analysis to decision makers. The business integration infrastructure is a sense and response system that communicates events via hugs with the internal and external business environment.
3. Typical Operational BI System and Major Components

In this section we identify the major components of Operational BI system based on improved business value with reduced action time. The proposed components are further sub-divided into sub-components that constitute the full Operational BI system. The functionality of the proposed components is also explained briefly.

A typical Operational BI system is shown in figure 1. The bottom layer consists of data sources that may be CRM, ERP, text files, emails and other legacy systems of the organization. This will act as the source to the Operational BI system.

![Typical Operational business intelligence system](image)

The data can be extracted on real time basis from these data sources and middle layer consists of various components of BI system. In addition to the basic components of classical BI system there are some additional components that enhance the business value to the organization and reduce the latency at data, analytics and reporting levels of the system. The major components
of Operational BI system are envisaged as business process monitoring and management, event notification and management, operational analytics, operational reporting and portal.

3.1 Business Process Monitoring (BPM)

Every organization is associated in extending product or services to the external world that associate process. In the competitive business environment the business processes of an organization to be measured from time to time. In order to measure the business process there is a great need of tools to help us understand, manage, and improve what organizations do from time to time. Performance measures quantitatively tell us something important about products, services, and the processes of an organization. This functionality can be achieved from the help of business process monitoring (BPM). Operational BI applications are closely integrate with operational data and operational processes Prasanna Keny et al (2006). BPM release key operational metrics to BI and BI captures these metrics on real time basis as events unfold in the process domain. Further, if the organization is able to establish a relationship between strategic levels to the operational level metric behind the trends. BPM provides to configure and monitor various key performance indicators (KPIs) and service level agreements (SLAs) of business operational measurements.

Business rules engine facilitates configuration of business rule definitions and change of these configurable values from time to time. More commonly, Operational BI applications generally cause changes to operational procedures or processes. The changes are usually needed to ensure that the operational BI information is optimally used. Since these are new applications, the traditional processes may have to be rethought or rewritten to ensure proper execution as described Sarma et al (2012). Hence, there is a great need of business rule engine for Operational BI system to make the changes of business rules dynamically without affecting system.

3.2 Event Notification and Management

Event notification is one of the most important functional components of Operational BI system. It identifies events that are deviated with respect to the business rules configured in the system and then generates alerts on real time basis. Event notification will receive input messages from various other components of the system that business process and monitoring include KPIs/SLAs of the system under measurement. In addition to this, if any abnormal event happening in the system in terms of process behavior or system functionality then monitoring system will
notify and communicate to an event engine. In turn event engine will generate alerts and send to the user. Event engine parses the incoming messages, fetches data from the configuration files and validates the configured parameters limits with computed aggregate values from incoming data. Event engine selects the message and dynamically composes the contents from predefined templates and deliver the message to the user.

### 3.3 Operational Analytics

Operational Analytics engine is an important functional component of Operational BI system. This contains several analytical engines that suits to both operational and as well as traditional BI system. An Operational BI system can be considered as hybrid BI system because it covers operational and analytics functionalities together. This provides 360° view business intelligence to the organization to cover decision making information to all the levels of users that include operational, tactical and strategic. The major functionality of analytics layer is to extract knowledge from the operational data and operational business process. This engine also works as per the predefined schedule to extract knowledge from historical data that is from data warehouse. This includes various mining algorithms such as association, classification, clustering, classification, and prediction.

In a typical data mining process, full data is rarely collected in one attempt. The collection of data is a continuous and ongoing process. Typically in the context of business operations the data collection time is as small as weeks, days, hours, minutes and even seconds. In many cases, data collection is carried out in phases. Consequently, the content of the underlying database changes over time. To keep track changes in the mining pattern to be studied in accordance with changes in operational data the specific mining algorithms have to be executed whenever the underlying database changes. We refer to this problem as incremental mining which includes update, insertion and deletion operations in not only database but also the corresponding operations on extracted knowledge to get timely update of discovering hidden relations.

Incremental data mining system will help to analyze operational data and the corresponding changes in the historical data (in terms of insertion, deletion and update operations) and support ongoing maintenance of data warehouse or business intelligence applications. The objective of incremental mining is defined Nittaya kerdprasaop et al (2003) to avoid re-learning of rules from
the old data and utilize knowledge that has already been discovered. This greatly reduces analysis latency of action time which in turn brings value to the business.

The typical process of incremental mining system is shown in figure 3 that maintains the both incremental mining results and as well as previous mining results. Previous mining results will be updated with recently extracted mining results and this forms an updated mining results.

![Figure 3: Typical Incremental mining system](image)

**3.4 Operational Reporting**

The report engine includes customizable reports on self-service basis that can present high-level findings as well as enable a user to drill down to find specific details. This module includes standard report templates that provide the user to create customizable reports. The reports module consist infrastructure for strategic reporting to serve the strategic and tactical management as well as operational reporting for low level decisions of business operations on day to day basis. Most commonly the following two popular approaches are employed for operational reports by querying transaction system directly, or they can off-load the transaction data and query the data separately as described Eckerson, (2007).

**3.5 Portal**

Portal will act a collaborative tool between an operational and as well as analytic applications and is shown in figure 4.

![Figure 4: Integration between OLTP and OLAP system with Portal](image)
This provides not only a common interface to various users of the system that acts as a single entry point for incoming requests but also integrates between Online analytical processing (OLTA) and Online transaction processing systems (OLTP). It acts as information dissemination tool. The users can login and access resources such as reports, dashboards and other business applications and services as per their access privileges.

4. Improved Business Value

The identified components of Operational BI system are based on business value and reduced action time. According to figure 1 it is observed that the business value increases when action time reduces. In order to reduce action time the components in the system could be decomposed in such a way that they work efficiently and integrates with other components in the system to reduce the overall execution time. BPM provides an efficient monitoring of the operational business process and also managing the same. The monitored process results can be stored in the database and as well as it helps to provide the managing the key performances of the organization. Business rule engine provides not only configuration of rules and is flexible to change from time to time as business is agile in nature. With the help of BPM the current system behavior can analyze against the business processes interested in the organization that helps to the users of the organization in terms of improved business value as opposed to classical BI system.

Event notification and management component can send events to the required users who are interested in knowing the deviations of various threshold parameters that are configured in the system. Event notification is one of the important component in Operational BI system that detection if any deviation in the system parameters that are interested to measure but also communicates timely information to the required user for their immediate decision. Event management manages all the events that are occurred and stored in the database. An analytics engine will run on this database and provides extracted knowledge from the events to the user timely through which it improves the business value.

Operational analytics is one of the most important components in the system which has various sub-components such as incremental mining and event analytics. Through incremental mining system the pattern behavior of the incremental data can be analyzed from time to time. This will also provide the incremental mining results but also updated mining results from the previous
mining results. Incremental mining system uses previous mining results through which the computation time of mining greatly reduces but also timely information to the user of the organization.

Operational reporting component dynamic reporting is possible which can be linked to historical reports to forecast the trend analysis. This module not only provides the timey update of information but also trend analysis to the users through which the decision making become much faster and improves business value in the system. The portal integrates both OLAP and as well as OLTP system and also acts as single source of information dissemination. From the above identified major components of Operational BI system not only reduces action time but also increases business value greatly. The effective use of Operational BI system improves the efficiency of the system and also returns higher business value to the organizations.

5. Conclusion

In this paper, we identified and presented the major components of Operational BI system. The identification of these major components of the system is based on improving the business value with reduced action time. The major components of Operational BI system are business process monitoring and management, event notification and management, operational analytics, operational reporting and portal. The components are further divided into various sub-components that constitute the full system. The functionality of the proposed components explained briefly. The very key feature of Operational BI system is not only to provide operational intelligence to the operational users but also tactical and strategic users for their timely decision making. Finally, we conclude that the proposed major components of Operational BI system provides improved business value with reduced action time as compared with classical BI system and also brings business value in current time as opposed to past. Further this work can be extended by developing a prototype of a typical Operational BI system and extending the functionality to one or more business verticals.

References