**Mining High Utility Item sets without Candidate Generation**

ABSTRACT

High utility item sets refer to the sets of items with high utility like profit in a database, and efficient mining of high utility item sets plays a crucial role in many real life applications and is an important research issue in data mining area. To identify high utility item sets, most existing algorithms first generate candidate item sets by overestimating their utilities, and subsequently compute the exact utilities of these candidates. These algorithms incur the problem that a very large number of candidates are generated, but most of the candidates are found out to be not high utility after their exact utilities are computed. In this paper, we propose an algorithm, called HUI-Miner (High Utility Item set Miner), for high utility item set mining. HUI-Miner uses a novel structure, called utility-list, to store both the utility information about an item set and the heuristic information for pruning the search space of HUI-Miner. By avoiding the costly generation and utility computation of numerous candidate item sets, HUI-Miner can efficiently mine high utility item sets from the utility lists constructed from a mined database. We compared HUI-Miner with the state-of-the-art algorithms on various databases, and experimental results show that HUI-Miner outperforms these algorithms in terms of both running time and memory consumption.

**INTRODUCTION**

The rapid development of database techniques facilitates the storage and usage of massive data from business corporations, governments, and scientific organizations. How to obtain valuable information from various databases has received considerable attention, which results in the sharp rise of related research topics. Among the topics, the high utility itemset mining problem is one of the most important, and it derives from the famous frequent itemset mining problem [7, 8].

Mining frequent item sets is to identify the sets of items that appear frequently in transactions in a database. The frequency of an item set is measured with the support of the item set, i.e., the number of transactions containing the item set. If the support of an item set exceeds a user-specified minimum support threshold, the item set is considered as frequent. Most frequent itemset mining algorithms employ the downward closure property of item sets [4]. That is, all supersets of an infrequent itemset are infrequent, and all subsets of a frequent itemset are frequent. The property provides the algorithms with a powerful pruning strategy. In the process of mining frequent itemsets, once an infrequent itemset is identified, the algorithms no longer check all supersets of the itemset. For example, for a database with *n* items, after the algorithms identify an infrequent itemset containing *k* items, there is no need to check all of its supersets, i.e., 2(*n−k*) *−* 1 itemsets.

Mining of frequent itemsets only takes the presence and absence of items into account. Other information about items is not considered, such as the independent utility of an item and the context utility of an item in a transaction. Typically, in a supermarket database, each item has a distinct price/profit, and each item in a transaction is associated with a distinct count which means the quantity of the item one bought. Consider the database in Fig. 1. There are seven items in the utility table and seven transactions in the transaction table in the database. To calculate support, an algorithm only makes use of the information of the first two columns in the transaction table, the information of both the utility table and the other columns in the transaction table are discarded. However, an itemset with high support may have low utility, or vice versa. For example, the support and utility of itemset *{*bc*}* appearing in T1, T2, and T6 are 3 and 18 respectively(See Section 2.1 for utility computation), and those of itemset *{*de*}* appearing in T2 and T5 are 2 and 22. In some applications, such as market analysis, one may be more interested in the utility rather than support of itemsets. Traditional frequent itemset mining algorithms cannot evaluate the utility information about itemsets.

**LITERATURE SURVEY**

**IN 2012--IEEE Transactions--** A Data Mining Benchmark Suite.

Data mining constitutes an important class of scientific and commercial applications. Recent advances in data extraction techniques have created vast data sets, which require increasingly complex data mining algorithms to sift through them to generate meaningful information. The disproportionately slower rate of growth of computer systems has led to a sizeable performance gap between data mining systems and algorithms. The first step in closing this gap is to analyze these algorithms and understand their bottlenecks. With this knowledge, current computer architectures can be optimized for data mining applications. In this paper, we present MineBench, a publicly available benchmark suite containing fifteen representative data mining applications belonging to various categories such as clustering, classification, and association rule mining. We believe that MineBench will be of use to those looking to characterize and accelerate data mining workloads.

**IN 2009--*****IEEE* Transactions on Knowledge and Data Engineering,**

**IEEE Transactions on Knowledge and Data Engineering (TKDE)** is an archival journal published monthly designed to inform researchers, developers, managers, strategic planners, users, and others interested in state-of-the-art and state-of-the-practice activities in the knowledge and data engineering area

**IN 2003--IEEE Transactions— Data Mining and Knowledge Discovery**

*Data Mining and Knowledge Discovery* is a triannual [peer-reviewed](https://en.wikipedia.org/wiki/Peer_review)[scientific journal](https://en.wikipedia.org/wiki/Scientific_journal) focusing on [data mining](https://en.wikipedia.org/wiki/Data_mining). It is published by [Springer Science+Business Media](https://en.wikipedia.org/wiki/Springer_Science%2BBusiness_Media). As of 2012, the [editor-in-chief](https://en.wikipedia.org/wiki/Editor-in-chief) is Geoffrey I. Webb. It was started in 1996 and launched in 1997 by [Usama Fayyad](https://en.wikipedia.org/wiki/Usama_Fayyad) as founding Editor-in-Chief by Kluwer Academic Publishers (later becoming Springer). The first [Editorial](http://link.springer.com/article/10.1023/A%3A1009715820935) provides a summary of why it was started.[[1]](https://en.wikipedia.org/wiki/Data_Mining_and_Knowledge_Discovery#cite_note-DMKD-journal-1)

**IN 2009--IEEE Transactions—** **Pattern recognition** 

**Pattern recognition** is a branch of [machine learning](https://en.wikipedia.org/wiki/Machine_learning) that focuses on the recognition of patterns and regularities in [data](https://en.wikipedia.org/wiki/Data), although it is in some cases considered to be nearly synonymous with machine learning.[[1]](https://en.wikipedia.org/wiki/Pattern_recognition#cite_note-1) Pattern recognition systems are in many cases trained from labeled "training" data ([supervised learning](https://en.wikipedia.org/wiki/Supervised_learning)), but when no labeled data are available other algorithms can be used to discover previously unknown patterns ([unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning)).

**CHAPTER 3**

**EXISTING SYSTEM**

1. utility mining with the itemset share framework is a hard one as no anti-monotonicity property holds with the interestingness measure.
2. Prior works on this problem all employ a two-phase, candidate generation approach with one exception that is however inefficient and not scalable with large databases

**CHAPTER 4**

**PROPOSED SYSTEM**

1. d2HUP, namely Direct Discovery of High Utility Patterns, which is an integration of the depth-first search of the reverse set enumeration tree, the pruning techniques that drastically reduces the number of patterns to be enumerated, and a novel data structure that enables efficient computation of utilities and upper bounds.
2. This paper proposes a new algorithm, d2HUP, for utility mining with the itemset share framework, which finds high utility patterns without candidate generation.
3. A linear data structure, CAUL, is proposed, which targets the root cause of the two-phase, candidate generation approach adopted by prior algorithms, that is, their data structures cannot keep the original utility information.

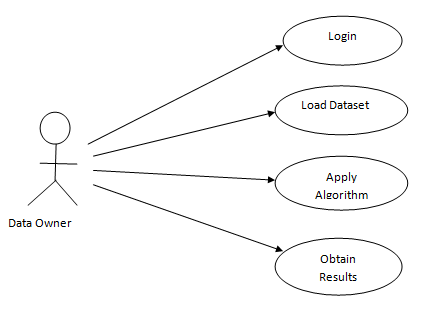
**Advantages of Proposed System**

Our approach is enhanced significantly by the lookahead strategy that identifies high utility patterns without enumeration.

**SYSTEM DESIGN**

System design is the process of defining the architecture, components, modules, interfaces and [data](http://en.wikipedia.org/wiki/Data) for a [system](http://en.wikipedia.org/wiki/System) to satisfy specified [requirements](http://en.wikipedia.org/wiki/Requirement). One could see it as the application of [systems theory](http://en.wikipedia.org/wiki/Systems_theory) to [product development](http://en.wikipedia.org/wiki/Product_development). There is some overlap with the disciplines of [systems analysis](http://en.wikipedia.org/wiki/Systems_analysis), [systems architecture](http://en.wikipedia.org/wiki/Systems_architecture) and [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering). If the broader topic of [product development](http://en.wikipedia.org/wiki/Product_development) "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing [systems](http://en.wikipedia.org/wiki/System) to satisfy specified [requirements](http://en.wikipedia.org/wiki/Requirement) of the user.

**Use Case Diagram**



Data Owner interacts with the modules like login, he has to login after the login load the dataset. To the dataset apply the d2hup algorithm and then get the results.

**6.1.2 DataFlow Diagram**

A data flow diagram is a graphical representation of the "flow" of data through an [information system](http://en.wikipedia.org/wiki/Information_system), modeling its *process* aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the [visualization](http://en.wikipedia.org/wiki/Data_visualization) of [data processing](http://en.wikipedia.org/wiki/Data_processing) (structured design).

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

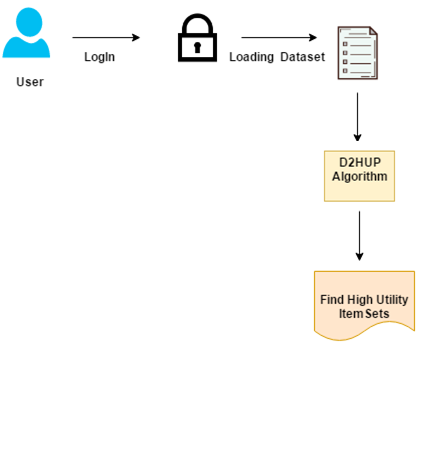
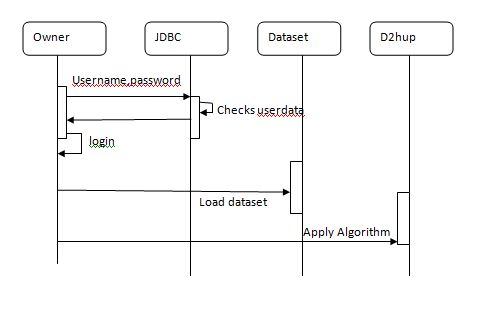


Fig.6.1.2: Dataflow diagram

**Sequence Diagram**

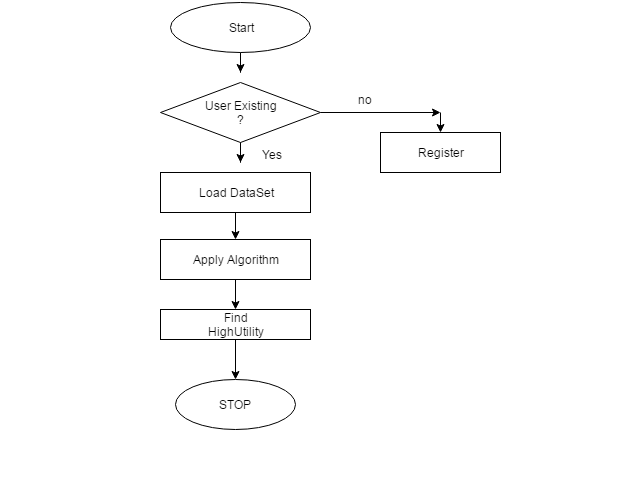
A sequence diagram in a UML is a kind of [interaction diagram](http://en.wikipedia.org/wiki/Interaction_diagram) that shows how processes operate with one another and in what order. It is a construct of a [Message Sequence Chart](http://en.wikipedia.org/wiki/Message_Sequence_Chart). A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically are associated with use case realizations in the Logical View of the system under development.

****

User first enter the username and password, JDBC checks that users data what they sent. After that login and then load the dataset and apply the algorithm to obtain the results.

**6.2 Flowcharts**

A flow chart is a graphical or symbolic representation of a process. Each step in the process is represented by a different symbol and contains a short description of the process step. The flow chart symbols are linked together with arrows showing the process flow direction.



First Start if user present goes to login otherwise they have to register. Load the dataset and apply the d2hup algorithm and find high utility to obtain the results then stop.

**System Modules:**

Administrator:

The administrator maintain database of the transactions made by customers in the daily market basis.

Construction of UP Tree

* Initially Transaction Utiity(TU) of each transaction is computed. Then TWU of each single item is also accumulated.

.• Utilities of unpromising items are eliminated from the TU of the transaction.

* Then remaining promising items in the transaction are sorted according to the descending order of TWU.

D2Hup Algorithm

**d2HUP**is an algorithm for discovering**high-utility itemsets**in a transaction database containing utility information. It is an algorithm that was shown to be more efficient than UP Growth and Two-Phase.

**CHAPTER 7**

**SYSTEM REQUIREMENT SPECIFICATION**

To be used efficiently, all [computer software](http://en.wikipedia.org/wiki/Computer_software) needs certain [hardware](http://en.wikipedia.org/wiki/Computer_hardware) components or other software resources to be present on a [computer](http://en.wikipedia.org/wiki/Computer). These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: [minimum](http://en.wikipedia.org/wiki/System_Requirements#Minimum_System_Requirements) and [recommended](http://en.wikipedia.org/wiki/System_Requirements#Recommended_system_requirements). With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

**7.2 Non functional requirements**

Non functional requirements are the functions offered by the system. It includes time constraints and constraints on the development process and standards. The non functional requirements are as follows:

* **Speed:** The system should process the given input into output within appropriate time.
* **Ease of use:** The software should be user friendly. Then the customers can use easily,

so it doesn’t require much training time.

* **Reliability:** The rate of failures should be less then only the system is more reliable
* **Portability**: It should be easy to implement in any system.

**7.2.1 Specific Requirements**

The specific requirements are:

* **User Interfaces:** The external users are the clients. All the clients can use this software for indexing and searching.
* **Hardware Interfaces:** The external hardware interface used for indexing and searching is personal computers of the clients. The PC’s may be laptops with wireless LAN as the internet connections provided will be wireless.
* **Software Interfaces:** The Operating Systems can be any version of Windows.
* **Performance Requirements:** The PC’s used must be atleast Pentium 4 machines so that they can give optimum performance of the product.

**7.3 Software requirements**

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application.

These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

* Java1.4 or higher
  + Java Swing – front end
  + JDBC –Database connectivity
  + UDP-User Datagram Protocol
  + TCP-Transmission Control Protocol
  + Networking-Socket programming
* ORACLE –Back end
* Windows 98 or higher-Operating System

**7.4 Hardware requirements**

The most common set of requirements defined by any [operating system](http://en.wikipedia.org/wiki/Operating_system) or [software application](http://en.wikipedia.org/wiki/Software_application) is the physical computer resources, also known as [hardware](http://en.wikipedia.org/wiki/Computer_hardware), A hardware requirements list is often accompanied by a [hardware compatibility list](http://en.wikipedia.org/wiki/Hardware_compatibility_list), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

All computer [operating systems](http://en.wikipedia.org/wiki/Operating_system) are designed for a particular [computer architecture](http://en.wikipedia.org/wiki/Computer_architecture). Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture.

The power of the [central processing unit](http://en.wikipedia.org/wiki/Central_processing_unit) (CPU) is a fundamental system requirement for any software. Most software running on [x86 architecture](http://en.wikipedia.org/wiki/X86_architecture) define processing power as the [model](http://en.wikipedia.org/wiki/List_of_microprocessors) and the [clock speed](http://en.wikipedia.org/wiki/Clock_rate) of the CPU. Many other features of a CPU that influence its speed and power, like [bus speed](http://en.wikipedia.org/wiki/Front_side_bus), [cache](http://en.wikipedia.org/wiki/CPU_cache), and [MIPS](http://en.wikipedia.org/wiki/Instructions_per_second) are often ignored. This definition of power is often erroneous, as [AMD](http://en.wikipedia.org/wiki/Advanced_Micro_Devices) [Athlon](http://en.wikipedia.org/wiki/Athlon) and [Intel](http://en.wikipedia.org/wiki/Intel) [Pentium](http://en.wikipedia.org/wiki/Pentium_%28brand%29) CPUs at similar clock speed often have different throughput speeds.

* + - * 10GB HDD(min)
      * 128 MB RAM(min)
      * Pentium P4 Processor 2.8Ghz(min)

**7.5 Overview of technologies**

The technologies used in TARF is described as below:

**7.5.1 History of Java**

Java language was developed by James Gosling and his team at sun Microsystems and released formally in 1995. Its former name is oak. Java Development Kit 1.0 was released in 1996 to popularize java and is freely available on Internet.

**7.5.2 Overview of Java**

Java is loosely based on c++ syntax, and is meant to be Object-Oriented Structure of java is midway between an interpreted and a compiled language. The java compiler into ByteCodes, which are secure and portable across different platforms, compiles Java programs. These byte codes are essentially instructions encapsulated in single type, to what is known as java virtual machine (JVM), which resides in standard browser.

JVM is available for almost all OS. JVM converts these byte codes into machine specific instructions at runtime. Java is actually a platform consisting of three components:

* Java programming language.
* Java library of classes and interfaces.
* Java Virtual Machine

**7.5.3 Features of Java**

* Java is a simple language. It does not make use of pointers, function overloading etc,.
* Java is object-oriented language and supports encapsulation, inheritance, Polymorphism and dynamic binding, but does not support multiple inheritance.
* Everything in java is an object except some primitive data types.
* Java is portable.
* It is an architecture neutral that is java programs once compiled can be executed on any machine that is enabled.
* Java is distributed in its approach and used for Internet programming.
* Java is robust, secured, high performing and dynamic in nature.
* Java supports multithreading. Therefore different parts of the program can be executed at the same time.

**7.6 Java Database Connectivity (JDBC)**

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

**7.6.1 Result set enhancements**

The JDBC 1.0 API provided result sets that had the ability to scroll in a forward directionally. Scrollable result sets allow for more flexibility in the processing of results by providing both forward and backward movement through their contents. In addition, scrollable result sets allow for relative and absolute positioning. For example, it's possible to move to the fourth row in a scrollable result set directly, or to move directly to the third row following the current row, provided the row exists. The JDBC API allows result sets to be directly updateable, as well.

**7.6.2 Batch updates**

The batch update feature allows an application to submit multiple update statements (insert/update/delete) in a single request to the database which can provide a dramatic increase in performance when a large number of update statements need to be executed.

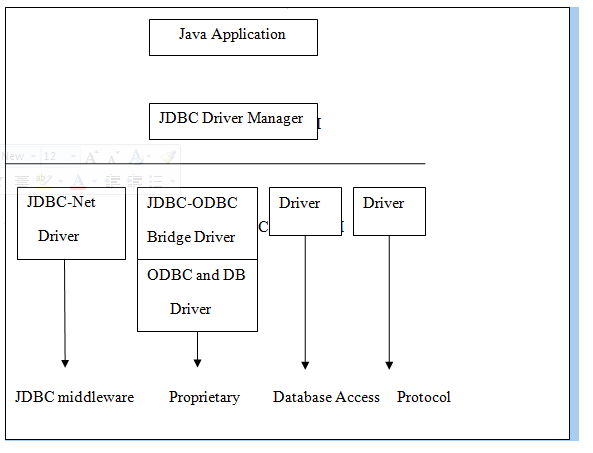
**7.6.3 Prepared Statements**

An element in a batch consists of a parameterized command and an associated set of parameters when a Prepared Statement is used. The batch update facility is used with a Prepared Statement to associate multiple sets of input parameter values with a single Prepared Statement object. The sets of parameter values together can then be sent to the underlying DBMS engine for execution as a single unit.

* 1. **JDBC drivers**

There are four types of JDBC drivers. They are:

* JDBC-ODBC bridge plus ODBC driver
  + JDBC-Net all-Java driver
  + Native-API partly-Java driver
  + Native-protocol all-Java driver

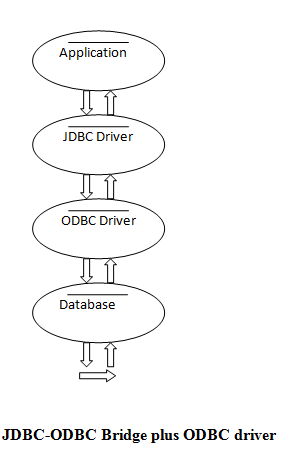
Figure 7.7: JDBC driver types.

Each of the JDBC driver is explained in detail below.

**7.7.1 JDBC-ODBC bridge plus ODBC driver**

The Java Soft bridge product provides JDBC access via ODBC drives. The ODBC binary code and in many cases database client code must be loaded on each client machine that uses this driver.

As a result, this kind of driver is most appropriate on a corporate network where client installations are not a major problem, or for application server code written in Java in three-tier architecture.



**Fig. 7.7.1: JDBC-ODBC Bridge plus ODBC driver**

**7.7.2 JDBC-Net all-Java driver**

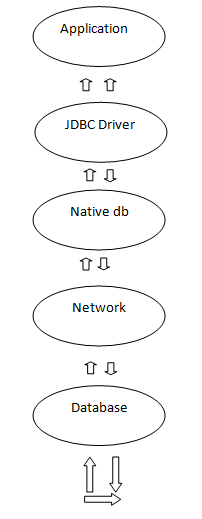
This driver translates JDBC calls into a DB MS-independent net protocol, which is then translated, to a DBMS protocol by a Server. This net Server middle ware is able to connect its all-Java clients to many different databases. The specific protocol used depends on the vendor. In general this is most flexible JDBC alternative.

It is likely that all vendors of this solution will provide products to also support Internet access through firewalls, etc, that the web imposes. Several vendors are adding JDBC drivers to their existing database middleware products.

**7.7.3 Native-API partly-Java Driver:**

This kind of driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS. Note that, like the Bridge driver, this style of driver requires that some binary code be loaded on each client machine.

Fig 7.7.3 shows Native-API partly JAVA Driver, where the application program requires a driver to connect to the database. Usually we use sun.jdbc.odbc.jdbcodbc driver this driver should request driver manager using driver manager.getconnection.



**Fig.7.7.3: Native API partly Java driver**

**7.7.4 Native-protocol all-Java driver:**

This kind of driver converts JDBC calls into the network protocol used by DBMS's directly. This allows a direct call from the client machine to the DBMS server and is practical solution for Internet access. Since many of these protocols are proprietary, database vendors themselves will be the primary source. Several database vendors have these in progress.

**7.8 Java RMI**

Java Remote Method Invocation (Java RMI) enables the programmer to create distributed Java technology-based to Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. RMI uses object serialization to marshal and unmarshal parameters and does not truncate types, supporting true object-oriented polymorphism

**7.9 Java Socket Programming**

URLs and URL Connections provide a relatively high-level mechanism for accessing resources on the Internet. Sometimes your programs require lower-level network communication, for example, when you want to write a client-server application.

In client-server applications, the server provides some service, such as processing database queries or sending out current stock prices. The client uses the service provided by the server, either displaying database query results to the user or making stock purchase recommendations to an investor. The communication that occurs between the client and the server must be reliable. That is, no data can be dropped and it must arrive on the client side in the same order in which the server sent it.

TCP provides a reliable, point-to-point communication channel that client-server application on the Internet use to communicate with each other. To communicate over TCP, a client program and a server program establish a connection to one another. Each program binds a socket to its end of the connection. To communicate, the client and the server each reads from and writes to the socket bound to the connection.

**7.9.1 What Is a Socket?**

Normally, a server runs on a specific computer and has a socket that is bound to a specific portnumber. The server just waits, listening to the socket for a client to make a connection request.

On the client-side: The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.



Fig.7.9.1: Socket connection request

If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.

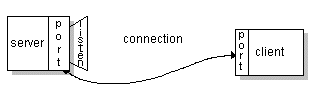


Fig. 7.9.2: Socket connection

On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server. The client and server can now communicate by writing to or reading from their sockets.

A *socket* is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent.

An endpoint is a combination of an IP address and a port number. Every TCP connection can be uniquely identified by its two endpoints. That way you can have multiple connections between your host and the server. The java.net package in the Java platform provides a class, Socket, that implements one side of a two-way connection between your Java program and another program on the network.

The Socket class sits on top of a platform-dependent implementation, hiding the details of any particular system from your Java program. By using the java.net.Socket class instead of relying on native code, your Java programs can communicate over the network in a platform-independent fashion. Additionally, java.net includes the Server Socket class, which implements a socket that servers can use to listen for and accept connections to clients.

This shows how to use the Socket and Server Socket classes. If we are trying to connect to the Web, the URL class and related classes (URL Connection, URL Encoder) are probably more appropriate than the socket classes. In fact, URLs are a relatively high-level connection to the Web and use sockets as part of the underlying implementation. See Working with URLs for information about connecting to the Web via URLs.

* 1. **Packages**

One of the most innovative features of java is packages. The packages both a naming and a visibility control mechanism we can define classes inside a package that are not accessible by code outside the package.It can define the class members that are only exposed to the other members of the same package. Java uses file system directories to store packages. For example the .class files for any classes you declare to be part of My Package must be stored in the directory called MyPackage remember that cases significant and directory name must match the package name exactly.

A package hierarchy must be reflected in the file system of your java development system. For example the package declared as -package java.awt.image; needs to be stored in java\awt\image in a windows environment.

**7.10.1 Java.lang package**

The java package, java.lang contains fundamental classes and interfaces closely tied to the language and run time system which includes the root classes that form the class hierarchy, types tied to the language definition, basic exceptions, math functions, threading, security functions as well as some information on the underlying native system.

**7.10.2 Java.util**

Data structures that aggregate objects are the focus of the Java.util package included in the packet is the collections API and organized data structure hierarchy influenced heavily by design pattern consideration.

**7.10.3 Java .security**

It provides the classes and interfaces for security framework. It includes classes that implement an easily configurable, fine grained access control security architecture. The packages also supports a generation and storage of cryptographic public key pairs. Finally this package provides classes that support signed/guarded objects and secure random number generation.

**7.11 Swings**

Swing is a widget toolkit for Java. It’s a part of sun Microsystems Java foundation classes-API for providing graphical user interface for Java programs. Swing was developed to provide a more sophisticated set of GUI components than the earlier abstract window toolkit. Swings provide a native look and feel that emulates look and feel of several look and feel unrelated to the underlying platform. Swings introduced a mechanism that allows the look and feel of every component in an application to be altered without making substantial changes to the application code. The introduction of support for a plugable look and feel allows swing components to emulate for the appearance of native components while still retaining the benefits of platform independence. The above feature also makes it easy to make an application written in swing look very different from native programs if desired.

**Look and feel**

In software design look and feel is used in respect of GUI and comprises of its design, including elements such as colors, shapes, layout and typefaces(the “LOOK”) as well as the behavior of dynamic elements such as button, boxes and menus(the “FEEL”). The term look and feel is used in reference to both software and websites.

**CHAPTER 9**

**TESTING**

Testing is a critical element which assures quality and effectiveness of the proposed system in (satisfying) meeting its objectives. Testing is done at various stages in the System designing and implementation process with an objective of developing an transparent, flexible and secured system. Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, complies with the standards, that it was designed to. Testing process involves building of test cases, against which, the product has to be tested.

* 1. **Test objectives**
* Testing is a process of executing a program with the intent of finding an error.
* A good case is one that has a high probability of finding an undiscovered error.
* A successful test is one that uncovers a yet undiscovered error. If testing is conducted successfully (according to the objectives) it will uncover errors in the software. Testing can't show the absences of defects are present. It can only show that software defects are present.

**9.2 Testing principles**

Before applying methods to design effective test cases, a software engineer must understand the basic principle that guides software testing. All the tests should be traceable to customer requirements.

**9.3 Testing design**

Any engineering product can be tested in one of two ways:

**9.3.1 White box Testing**

This testing is also called as glass box testing. Inthis testing, by knowing the specified function that a product has been designed to perform test can be conducted that demonstrates each function is fully operation at the same time searching for errors in each function.

it is a test case design method that uses the control structure of the procedural design to derive test cases.

**9.3.2 Black box Testing**

Inthis testing by knowing the internal operation of a product, tests can be conducted to ensure that "all gears mesh", that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

* Graph based testing methods
* Equivalence partitioning
* Boundary value analysis
* Comparison testing

**9.4 Testing strategies**

A software testing strategy provides a road map for the software developer. Testing is a set of activities that can be planned in advanced and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be defined for software engineering process.

**Any software testing strategy should have the following characteristics:**

* 1. Testing begins at the module level and works outward toward the integration of the entire computer based system.
  2. Different testing techniques are appropriate at different points in time.
  3. The developer of the software and an independent test group conducts testing.
  4. Testing and debugging are different activities but debugging must be accommodated in any testing strategy.

**9.5 Levels of Testing**

Testing can be done in different levels of SDLC. They are:

**9.5.1 Unit Testing**

The first level of testing is called unit testing. Unit testing verifies on the smallest unit of software designs-the module. The unit test is always white box oriented. In this, different modules are tested against the specifications produced during design for the modules. Unit testing is essentially for verification of the code produced during the coding phase, and hence the goal is to test the internal logic of the modules. It is typically done by the programmer of the module. Due to its close association with coding, the coding phase is frequently called “coding and unit testing.” The unit test can be conducted in parallel for multiple modules.

The Test cases in unit testing are as follows:

Table I: Unit Test Case 1

|  |  |
| --- | --- |
| Test Case ID | Unit Test Case 1 |
| Description | Login with username and Password |
| Input | UserName and Password |
| Expected output | Login Succesful. |
| Actual Result/Remarks | Got the expected output |
| Passed(?) | Yes |

Table II: Unit Test Case 2

|  |  |
| --- | --- |
| Test Case ID | Unit Test Case 2 |
| Description | Load the Dataset |
| Input | Dataset path |
| Expected output | Dataset Loaded |
| Actual Result/Remarks | Got the expected output |
| Passed (?) | Yes |

Table III: Unit Test Case 3

|  |  |
| --- | --- |
| Test Case ID | Unit Test Case 3 |
| Description | Read the Dataset |
| Input | Dataset.txt file |
| Expected output | Dataset read completely |
| Actual Result/Remarks | Got the expected output |
| Passed (?) | Yes |

Table IV: Unit Test Case 4

|  |  |
| --- | --- |
| Test Case ID | Unit Test Case 4 |
| Description | Apply Algorithm |
| Input | Dataset content and Minimum Utility |
| Expected output | Found itemsets with given minimum utility and higher. |
| Actual Result/Remarks | Working as required |
| Passed(?) | Yes |

**9.5.2 Integration Testing**

The second level of testing is called integration testing. Integration testing is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing. In this, many tested modules are combined into subsystems, which are then tested. The goal here is to see if all the modules can be integrated properly.

There are three types of integration testing:

* + - *Top-Down Integration*: Top down integration is an incremental approach to construction of program structures. Modules are integrated by moving downwards throw the control hierarchy beginning with the main control module.
    - *Bottom-Up Integration*: Bottom up integration as its name implies, begins Construction and testing with automatic modules.
    - *Regression Testing*: In this contest of an integration test strategy, regression testing is the re execution of some subset of test that have already been conducted to ensure that changes have not propagated unintended side effects.

Table IX: Integration Test Case

|  |  |
| --- | --- |
| Test Case ID | Integration Test Case 1 |
| Description | All servers are running properly |
| Input | packets is passed from one to another |
| Expected output | packets is received at server |
| Actual Result/Remarks | Working as required |
| Passed(?) | Yes |

**9.5.3 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Table X: Functional Testing items

|  |  |
| --- | --- |
| Valid Input | Identified classes of valid input must be accepted. |
| Invalid Input | Identified classes of invalid input must be rejected. |
| Functions | Identified functions must be exercised. |
| Output | Identified classes of application outputs must be exercised. |

***Systems/Procedures:*** Interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**9.6 Validation testing**

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been covered and corrected, and final series of software tests-validating testing may begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by customers. Reasonable expectation is defined in the software requirement specification- a document that describes all user visible attributes of the software. The specification contains a section title “validation criteria”. Information contained in that section forms the basis for validation testing approach

**9.7 Alpha testing**

It is virtually impossible for a software developer to forsee how the customer will really use a program. Instructions for use may be misinterpreted; strange combination of data may be regularly used and output that seemed clear to the tester may be unintelligible to a user in field.

When custom software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements by the end user rather than system developer and acceptable test can range from an informal “test drive” to a planned and systematically executed series of tests. In fact, acceptance testing can be conducted over a period of weeks or months, thereby uncovering cumulative errors that might degrade the system over time. If software is developed as a product to be used by many customers, it is impractical to perform formal acceptance test with each one. Most software product builders use a process called alpha and beta testing to uncover errors that only the end user seems able to find.

A customer conducts the alpha test at the developer’s site. The software is used in a natural setting with the developer “Looking over the shoulder” of the user and recording errors and usage problems. Alpha tests are conducted in controlled environment.

**9.8 Beta testing**

The beta test is conducted at one or more customer sites by the end user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a “live” application of the software in an environment that cannot be controlled by the developer. The customer records all problems that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta test, the software developer makes modification and then prepares for release of the software product to the entire customer base.

**9.9 System Testing and Acceptance Testing**

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Include recovery testing during crashes, security testing for unauthorized user, etc.

Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactorily. This testing in FDAC focuses on the external behavior of the system.

**CONCLUSIONS**

In this paper, we have proposed a novel data structure, utility-list, and developed an efficient algorithm, HUIMiner, for high utility itemset mining. Utility-lists provide not only utility information about itemsets but also important pruning information for HUI-Miner. Previous

algorithms have to process a very large number of candidate itemsets during their mining processes. However, most candidate itemsets are not high utility and are discarded finally. HUI-Miner can mine high utility itemsets without candidate generation, which avoids the costly generation and utility computation of candidates. We have studied the performance of HUI-Miner in comparison with the stateof- the-art algorithms on various databases. Experimental

results show that HUI-Miner gains significant performance improvement over these algorithms in terms of both running time and memory consumption.

**REFERENCES**

[1] Frequent Itemset Mining Dataset Repository. http:// fimi.ua.ac.be/, 2012.

[2] NU-MineBench: A Data Mining Benchmark Suite. <http://cucis.ece.northwestern.edu/projects/> DMS/MineBench.html, 2012.

[3] Valgrind: A GPL’d System for Debugging and Profiling Linux Programs. http://valgrind.org/, 2012.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. Int'l Conf. Very Large Data Bases*, pages 487–499, 1994.

[5] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee. Efficient tree structures for high utility pattern mining in incremental databases. *IEEE Transactions on Knowledge and Data Engineering*, 21(12):1708–1721, 2009.