ISSN:2321-1156

International Journal of Innovative Research in Technology & Science(IJIRTS)

A SIGNATURE – BASED INDEXING METHOD FOR EFFICIENT CONTENT –BASED RETRIEVAL OF RELATIVE TEMPORAL PATTERNS

K.PRIYA III MCA ‘A’; MRS.R.LATHA Assistant Professor; S.PAULRAJ III MCA ‘B’; S.PRABHU III MCA ‘A’;

Abstract

A number of algorithms have been proposed for the discovery of temporal patterns. However, since the number of generated patterns can be large, selecting which patterns to analyze can be nontrivial. There is thus a need for algorithms and tools that can assist in the selection of discovered patterns so that subsequent analysis can be performed in an efficient and, ideally interactive manner. In this paper, we propose a signature-based indexing method to optimize the storage and retrieval of a large collection of relative temporal patterns.

Index Terms—Content-based data mining queries, organizing temporal patterns, signature-based indexing methods

1.Introduction

Many rule discovery algorithms in data mining generate a large number of patterns/rules, sometimes even exceeding the size of the underlying database, with only a small fraction being of interest to the user It is generally understood that interpreting the discovered patterns/rules to gain insight into the domain is an important phase in the knowledge discovery process. However, when there are a large number of generated rules, identifying and analyzing those that are interesting becomes difficult. For example, providing the user with a list of association rules ranked by their confidence and support might not be a good way of organizing the set Of rules as this method would overwhelm the user and not all rules with high confidence and support are necessarily interesting for a variety of reasons.

Therefore, to be useful, a data mining system must manage the generated rules by offering flexible tools for rule selection. In the case of association rule mining, several approaches for the post processing of discovered association rules have been discussed. One approach is to group “similar” rules which work well for a moderate number of rules. However, for a larger number of rules it produces too many clusters. A more flexible approach is to allow the identification of rules that are of special importance to the user through templates or data mining queries. This approach can complement the rule grouping approach and has been used to specify interesting and uninteresting classes of rules (for both association and episodic rules).

The importance of data mining queries has been highlighted by the introduction of the inductive database concept, which allows the user to both queries the data and query patterns, rules, and models extracted from these data.

2. LITERATURE REVIEW

Temporal Database stores data relating to time instances. It offers temporal data types and stores information relating to past, present, and future time, for example, the history of the stock market or movement of employees within an organization. Thus, a temporal database stores a collection of time related data.

I use the type date provided by a non-temporal DBMS to design the temporal database.

2.1 TEMPORAL PATTERN GENERATION

A Temporal pattern of size n is defined by a pair(s,M), where s:{1,...,n}->S maps index i to the corresponding state, and M is an n x n matrix whose elements M[i,j] denote the relationship between intervals [b_i,f_j) and [b_j,f_j). The size of a temporal pattern α is the number of intervals in the pattern, denoted dim (α). If the size of α is n, then α is called an n-pattern.

Example {Key, State-A, start time, end time, State-B}

As an example, consider the patterns in Fig. 1.3; p1 is a sub pattern of p3, but it is not a sub pattern of p4. We can obtain p1 from p3 by removing interval state D; on the other hand, removing interval states C and D from p4 would not result in p1.

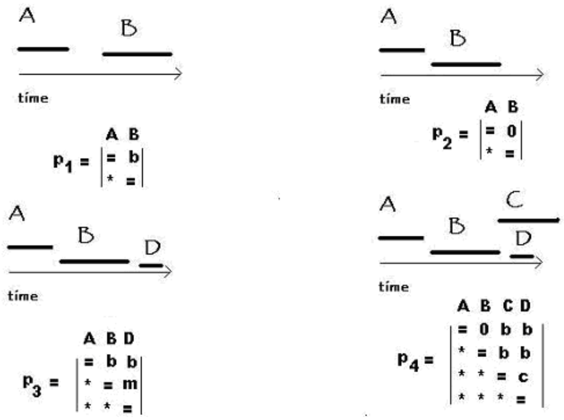


Fig: 2.1.1An Example of Temporal Patterns.

2.2 SIGNATURE-BASED INDEX CONSTRUCTION

In set retrieval with signature files, a target signature is generated for each target set and stored in the signature file. A number of signature generation methods have been proposed by Faloutsos and Chistodoulakis (1987) in the context of text retrieval. These methods are Word Signatures (WS), Superimposed Coding (SC), Bit-Block Compresion (BC), and Run Length Encoding (RL). The description of each method for generating target signatures is given below.

2.2.1Word signature (WS).

In the WS method each element of the target set is hashed into a bit pattern of a certain length. These patterns, called word Signatures are then concatenated to form the target signature.

2.2.2 Superimposed Coding (SC).

In the SC method, each element in a target set is hashed to a binary bit pattern called an element signature. All element signatures have F bit length, and exactly m bits are set to ‘1’, where m < F.

F is called the length of a signature, while m is called the weight of an element signature. Then, a target signature is obtained by bit-wise OR-ing (superimposed coding) element signatures of all the elements in the target set.

2.2.3 Bit-Block Compression (BC).

The signature extraction process for BC is similar to SC. The difference is that the original size (length) of the signature, designated as B, is large, and for each element of a target set only one bit is set to ‘1’ (i.e., m = 1). As a result, the bit vector B of the set signature is sparse. Therefore, before storing the signature, B is divided into groups of consecutive bits of size b and compressed.

2.2.4 Run Length Encoding (RL).

The RL method is similar to both SC and BC. It differs from BC only in the compression method. RL records the distances between the positions of bits with value ‘1’. Of these four methods, the most commonly used method in set retrieval is the superimposed coding (Helmer & Moerkotte 1999, Ishikawa et al. 1993, Tousidou, Bozanis & Manolopoulos 2002, Morzy & Zakrzewicz 1998). Therefore, for the rest of this chapter, unless stated otherwise, it will be assumed that the superimposed coding is used to generate set signatures. The following figure illustrates the generation of set signature using the superimposed coding when the value of
F = 8 and m = 2.

2.2.5 Constructing Signature Files

- Let a set of states S={A,B,C,D}.
- State mapping function f (x) defined as f (A)=1, f (B)=2, f (C)=3 each state with unique value.
- Let pattern p1={1,30}
- Then hash(1)=00000010,
  Hash (30)=01000000.
- Then the signature p1 is computed using bitwise union of the hash(1)
  & hash (30)

2.3 ANSWERING CONTENT-BASED QUERIES USING SIGNATURE FILES

Given a temporal pattern database D and a query pattern q, the algorithm for evaluating sub pattern queries is called evaluate sub pattern (D,q), which finds temporal patterns D, that contains q . If the signatures are stored in SSF, evaluate sub pattern

(D,q) by using the algorithm, The equivalent set E(q) of q is first calculated and then , a query signature s1 is formed . Each target signature s2 in SSF is done examined against the query signature s1.

2.3.1 Query Answering:

```
1: E(q) = Equivalent Set(q)
2: sigq = Signature(E(q))
3: Retrieve the bit slices corresponding to the bit position set to "0" in sigq
4: Perform a bitwise union operation on the retrieved bit slices
5: for each entry where "0" is set in the resulting union bit slice do
6: add the corresponding $pid_p$ into the PID list
7: end for
8: for each $pid_p$ in the PID list do
9: Retrieve p from D
10: if $p \leq q$ then
11: Add p into AnswerSet
12: end if
13: end for
14: return AnswerSet.
```

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Mine-Rule DMQL and OLE DB. These languages are designed to generate the rules from the data rather than allow queries over the discovered rules. Post processing of association rules, an area in which little research to date has been conducted.

In previous number of algorithm proposed for discover the temporal patterns, in that generated patterns are large. Mine-Rule, DMQL to generate the rules from the data rather than allow queries over the discovered rules.

3.2. DISADVANTAGES OF EXISTING SYSTEM

- Set Based Indexing Method.
- Do not consider the order of items within the sets, as is required in the case of indexing and retrieval of sequential patterns.
- Here the generated temporal patterns are large. It will take more time to retrieve the data.

3.3. PROPOSED SYSTEM

Efficiently retrieving subsets of a large collection of previously discovered temporal patterns. Highlights on supporting content-based queries of temporal patterns, as opposed to point- or range-based queries.

Iaddress the problem of efficiently retrieving subsets of a large collection of previously discovered temporal patterns. Improves the performance of temporal pattern retrieval

This retrieval system is currently using for monitoring the behavior of a single pattern or a group of patterns over a time.

3.4. ADVANTAGES OF PROPOSED SYSTEM

•Here I address the problem of efficiently retrieving subset of a large collection of previously discovered temporal patterns.
•Generated temporal patterns size is been reduced.
•Here the performance of temporal patterns retrieval is improved.
•The retrieval system is currently using for monitoring the behavior of a single pattern or a group of patterns over time.

4. SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE DIAGRAM

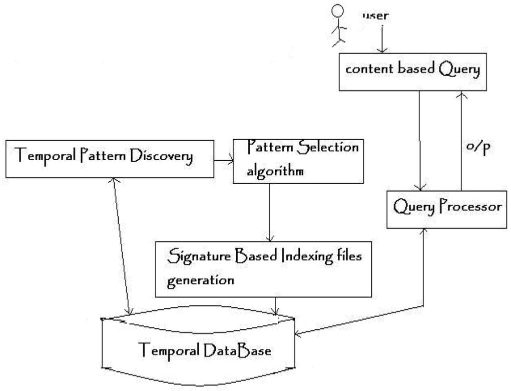

user
content based Query
o/p
Temporal Pattern Discovery
Pattern Selection algorithm
Query Processor
Signature Based Indexing files generation
Temporal DataBase

Fig: System Architecture Diagram

As the above figure shows, my system consists of a number of components such as a GUI to receive the user commands and transaction information, a query processor, generating temporal patterns for the dataset available, generating appropriate signature files for the patterns and so on.

The temporal database is the one that stores the data information along with the necessary time and states. As the temporal data are added into the database, typical patterns are created for the database automatically. After this, signature is created for the temporal pattern appropriately.

When user performs a transaction, the process details are stored in the database and are processed automatically as described above.

When a particular query is submitted, the query processor comes into act processing the query submitted. It compares the processed query with the signature files and gives out the desired result set.

4.2 DATA FLOW DIAGRAM

User Login

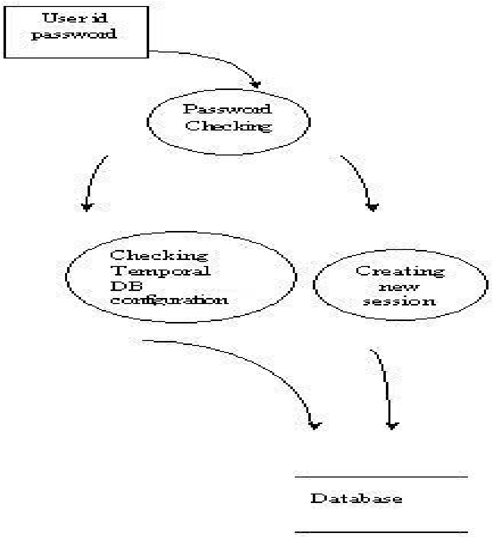
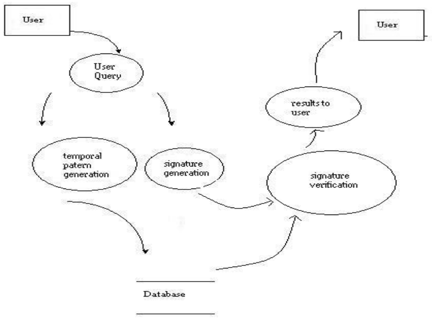


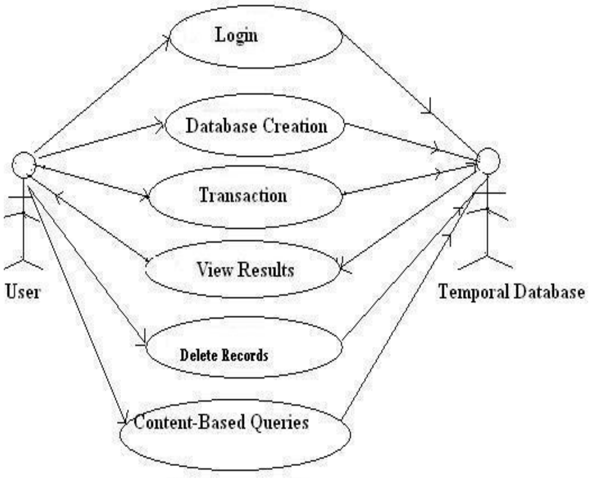
Fig: Data Flow Diagram

Transaction



USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.



Answering Content Based Queries

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. It differs from the flowchart as it shows the data flow instead of the control flow of the program. A data-flow diagram can also be used for the visualization of data processing (structured design).Developing a data-flow diagram helps in identifying the transaction data in the data model.

There are different notations to draw data-flow diagrams, defining different visual representations for processes, data stores, data flow, and external entities. In the above DFD the flow of data that is provided in our system has been illustrated.

4.3 ACTIVITY DIAGRAM

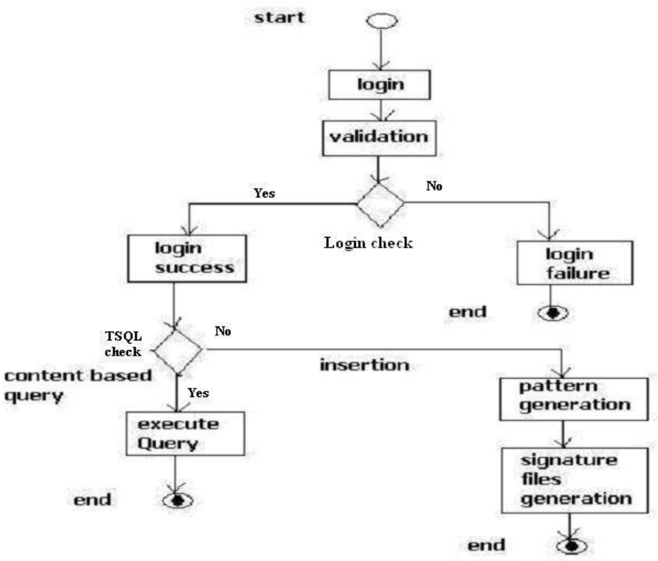


Fig: Activity Diagram

Activity diagram is a loosely defined diagram technique for showing workflows of stepwise activities and actions, with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

Sequence Diagram

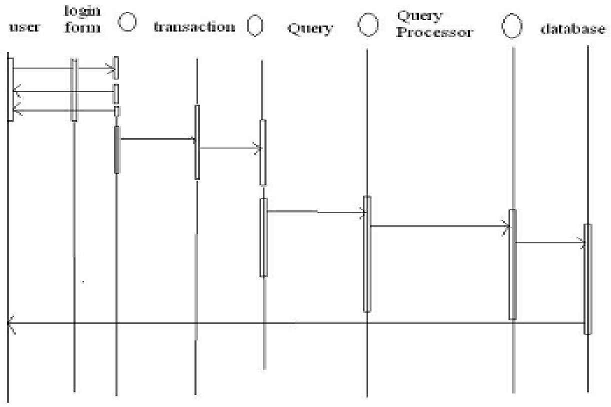


Fig: Sequence Diagram

Sequence Diagrams are easy and inductive way of describing the behavior of a system by viewing the interaction between the system and its environment. A sequence diagram shows the interaction arranged in a time sequence. It shows the object participating by their life lines and the messages they exchange, arranged in a time sequence.

CONCLUSION

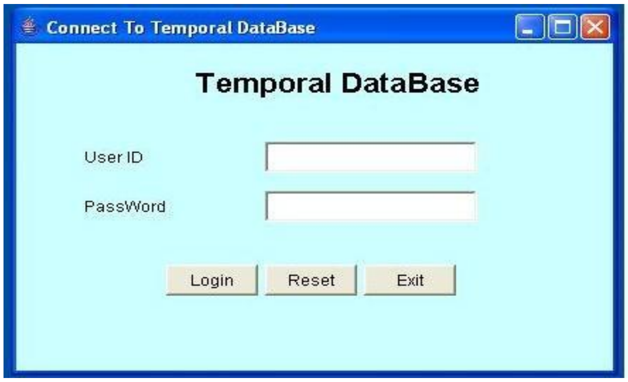
The use of a signature-based index for content-based retrieval of temporal patterns has been completed. The signatures of temporal patterns are created by first converting temporal patterns into equivalent sets and then generating the signatures from the equivalent sets. The study focused on the sequential and BSSF organizations, and a series of experiments compared the performance of both signature files in processing sub pattern and super pattern queries.

In conclusion, the use of signature files improves the performance of temporal pattern retrieval; the bit-slice signature file performs better than the SSF and is a good choice for content-based retrieval of temporal patterns.

FUCTURE ENHANCEMENTS

At Present this retrieval system is currently being combined with visualization techniques for monitoring the behavior of a single pattern or a group of patterns over time. In future we may extend the signature based on time with visual Query Operator.

USER LOGIN

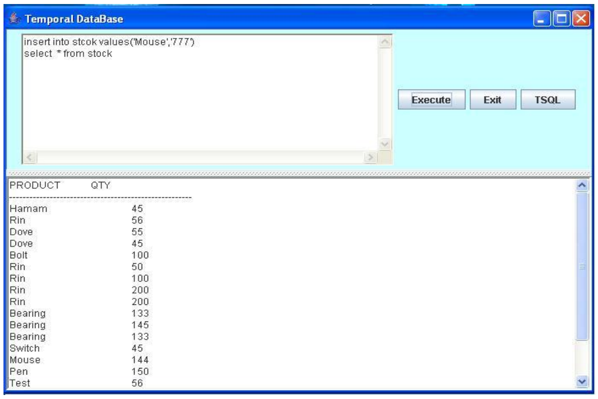

Connect To Temporal DataBase

Temporal DataBase

User ID

PassWord

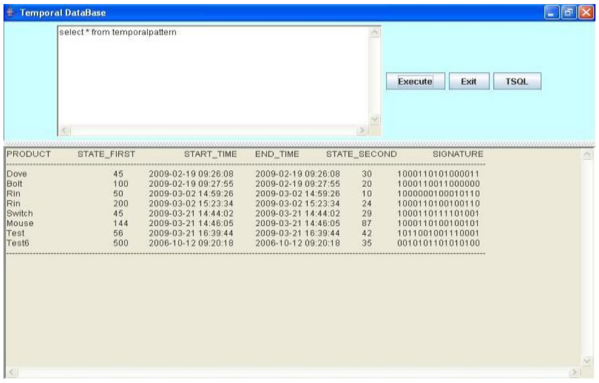
QUERY TOOL



```
insert into stcok values('Mouse','777')
select  * from stock
```

```
PRODUCT         QTY
---------------------------------------------------
Hamam                    45
Rin                      56
Dove                     55
Dove                     45
Bolt                     100
Rin                      50
Rin                      100
Rin                      200
Rin                      200
Bearing                  133
Bearing                  145
Bearing                  133
Switch                   45
Mouse                    144
Pen                      150
Test                     56
```

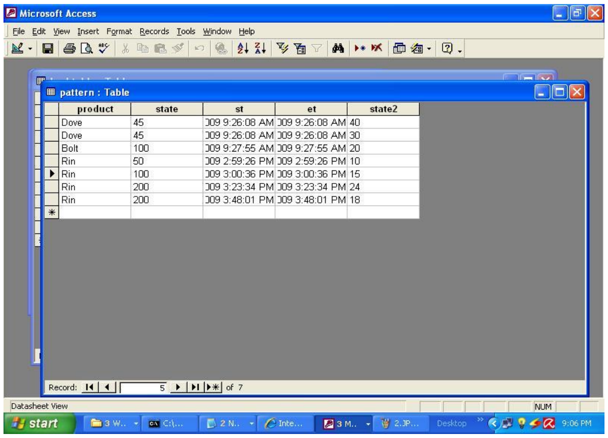
PATTERNS



```
select * from temporalpattern
```

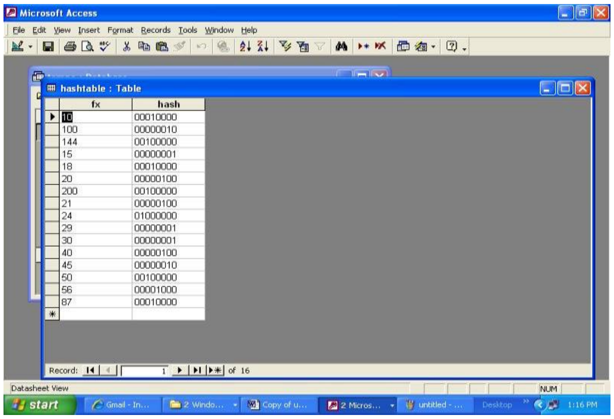
```
PRODUCT       STATE_FIRST          START_TIME       END_TIME        STATE_SECOND        SIGNATURE
-------------------------------------------------------------------------------------------------
Dove                 45      2009-02-19 09:26:08   2009-02-19 09:26:08    30      1000110101000011
Bolt                 100     2009-02-19 09:27:55   2009-02-19 09:27:55    20      1000110011000000
Rin                  50      2009-03-02 14:59:26   2009-03-02 14:59:26    10      1000000100010110
Rin                  200     2009-03-02 15:23:34   2009-03-02 15:23:34    24      1000110100100110
Switch               45      2009-03-21 14:44:02   2009-03-21 14:44:02    29      1000110111101001
Mouse                144     2009-03-21 14:46:05   2009-03-21 14:46:05    87      1000110100100101
Test                 56      2009-03-21 16:39:44   2009-03-21 16:39:44    42      1011001001110001
Test6                500     2006-10-12 09:20:18   2006-10-12 09:20:18    35      0010101101010100
-------------------------------------------------------------------------------------------------
```

PATTERN STORED IN DATABASE



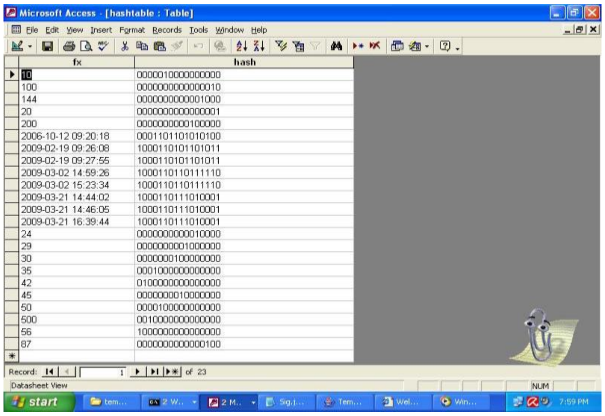
product	state	st	et	state2
Dove	45	009 9:26:08 AM	009 9:26:08 AM	40
Dove	45	009 9:26:08 AM	009 9:26:08 AM	30
Bolt	100	009 9:27:55 AM	009 9:27:55 AM	20
Rin	50	009 2:59:26 PM	009 2:59:26 PM	10
Rin	100	009 3:00:36 PM	009 3:00:36 PM	15
Rin	200	009 3:23:34 PM	009 3:23:34 PM	24
Rin	200	009 3:48:01 PM	009 3:48:01 PM	18

HASH TABLE: TABLE



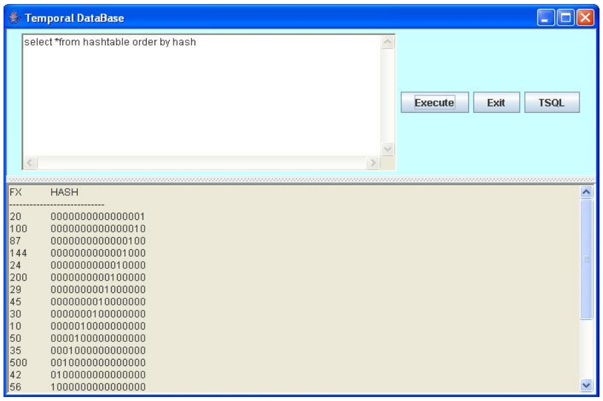
fx	hash
10	00010000
100	00000010
144	00100000
15	00000001
18	00010000
20	00000100
200	00100000
21	00000100
24	01000000
29	00000001
30	00000001
40	00000100
45	00000010
50	00100000
56	00001000
87	00010000

HASH VALUES STORED IN DATABASE


Microsoft Access - [hashtable : Table]
File Edit View Insert Format Records Tools Window Help
Record: 1 of 23
Datasheet View
NUM
start
tem...
2 W..
2 M..
Sig.j...
Tem...
Wel...
Win...
7:59 PM

fx	hash
10	0000010000000000
100	0000000000000010
144	0000000000001000
20	0000000000000001
200	0000000000100000
2006-10-12 09:20:18	0001101101010100
2009-02-19 09:26:08	1000110101101011
2009-02-19 09:27:55	1000110101101011
2009-03-02 14:59:26	1000110110111110
2009-03-02 15:23:34	1000110110111110
2009-03-21 14:44:02	1000110111010001
2009-03-21 14:46:05	1000110111010001
2009-03-21 16:39:44	1000110111010001
24	0000000000010000
29	0000000001000000
30	0000000100000000
35	0001000000000000
42	0100000000000000
45	0000000010000000
50	0000100000000000
500	0010000000000000
56	1000000000000000
87	0000000000000100

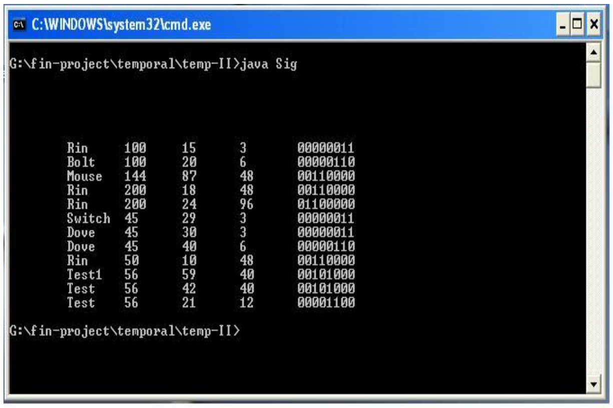
HASH VALUE SELECTION



```
select *from hashtable order by hash
```

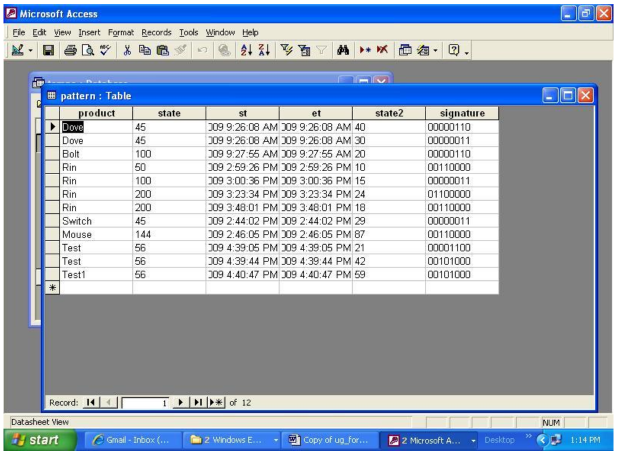
```
FX        HASH
---------------------------
20        0000000000000001
100       0000000000000010
87        0000000000000100
144       0000000000001000
24        0000000000010000
200       0000000000100000
29        0000000001000000
45        0000000010000000
30        0000000100000000
10        0000010000000000
50        0000100000000000
35        0001000000000000
500       0010000000000000
42        0100000000000000
56        1000000000000000
```

SIGNATURE GENERATION



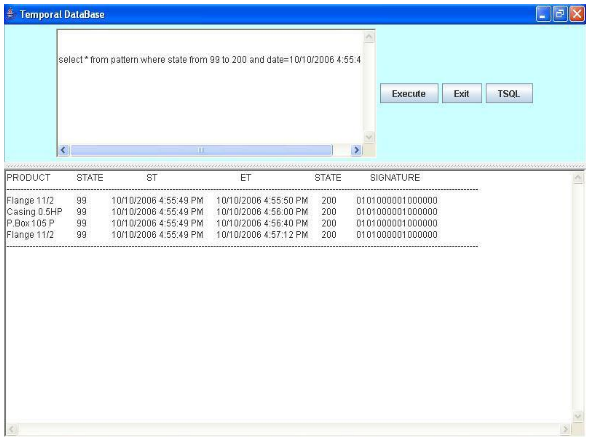
```
G:\fin-project\temporal\temp-II>java Sig



        Rin     100     15      3       00000011
        Bolt    100     20      6       00000110
        Mouse   144     87      48      00110000
        Rin     200     18      48      00110000
        Rin     200     24      96      01100000
        Switch  45      29      3       00000011
        Dove    45      30      3       00000011
        Dove    45      40      6       00000110
        Rin     50      10      48      00110000
        Test1   56      59      40      00101000
        Test    56      42      40      00101000
        Test    56      21      12      00001100

G:\fin-project\temporal\temp-II>
```

SIGNATRURE GENERATION: OUPUT



	product	state	st	et	state2	signature
▶	Dove	45	009 9:26:08 AM	009 9:26:08 AM	40	00000110
	Dove	45	009 9:26:08 AM	009 9:26:08 AM	30	00000011
	Bolt	100	009 9:27:55 AM	009 9:27:55 AM	20	00000110
	Rin	50	009 2:59:26 PM	009 2:59:26 PM	10	00110000
	Rin	100	009 3:00:36 PM	009 3:00:36 PM	15	00000011
	Rin	200	009 3:23:34 PM	009 3:23:34 PM	24	01100000
	Rin	200	009 3:48:01 PM	009 3:48:01 PM	18	00110000
	Switch	45	009 2:44:02 PM	009 2:44:02 PM	29	00000011
	Mouse	144	009 2:46:05 PM	009 2:46:05 PM	87	00110000
	Test	56	009 4:39:05 PM	009 4:39:05 PM	21	00001100
	Test	56	009 4:39:44 PM	009 4:39:44 PM	42	00101000
	Test1	56	009 4:40:47 PM	009 4:40:47 PM	59	00101000
*						

CONTENT BASED QUERY PROCESS: OUTPUT



```
select * from pattern where state from 99 to 200 and date=10/10/2006 4:55:4
```

```
PRODUCT      STATE       ST                      ET                      STATE     SIGNATURE
--------------------------------------------------------------------------------------------------
Flange 11/2     99     10/10/2006 4:55:49 PM   10/10/2006 4:55:50 PM   200    0101000001000000
Casing 0.5HP    99     10/10/2006 4:55:49 PM   10/10/2006 4:56:00 PM   200    0101000001000000
P.Box 105 P     99     10/10/2006 4:55:49 PM   10/10/2006 4:56:40 PM   200    0101000001000000
Flange 11/2     99     10/10/2006 4:55:49 PM   10/10/2006 4:57:12 PM   200    0101000001000000
--------------------------------------------------------------------------------------------------
```

REFERENCES

1. J. Santos, D. Gomes, S. Sargento, R. L. Aguiar, N. Baker, M. Zafar, and
   A. Ikram, “Multicast/broadcast network convergence in next generation
   mobile networks,” Comput. Netw., vol. 52, pp. 228–247, January 2008.
2. Kathy Sierra & Bert Bates, “Head First Java”, O’REILLY May-2003.
3. What is 100% Pure Java, http://www.javacoffeebreak.com/faq/faq0006.html.
4. Mike McGrath, “Java Server Pages”, Dreamtech Press, 2005.
5. [1] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The case for
   vm-based cloudlets in mobile computing,” IEEE Pervasive Computing, vol. 8, pp. 14–23, 2009.
6. J.F.Roddick and M.Spiliopoulou, “A Survey of Temporal knowledge Discovery Paradigms and Methods,” IEEE Trans. Knowledge and Data Eng., vol. 14, no. 4, pp. 750-767, Mar./ Apr.2002.
7. [2] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, “Thinkair:
   Dynamic resource allocation and parallel execution in the cloud for
   mobile code offloading,” in Proc. of IEEE INFOCOM, 2012.
8. D.L. Lee and C.-W. Leng, “A Partitioned Signature File Structure for Multi attribute and Text Retrieval,” Proc. Sixth Int’l Conf. Data Eng. (ICDE ’90), pp. 389-397, 1990.
9. [4] T. Coppens, L. Trappeniners, and M. Godon, “AmigoTV: towards a
   social TV experience,” in Proc. of EuroITV, 2004.
10. N. Mamoulis, D.W. Cheung, and W. Lian, “Similarity Search in Sets and Categorical Data Using the Signature Tree,” Proc. 19th Int’l Conf. Data Eng.(ICDE’03),U.Dayal,K.Ramamritham, and T.Vijayaraman, eds., pp. 75-86, 2003.