RDB-MINER: A SQL-Based Algorithm for Mining True Relational Databases

Abdallah Alashqur
Faculty of Information Technology
Applied Science University
Amman, JORDAN

***Abstract*— Traditionally, research in the area of frequent itemset mining has focused on mining market basket data. Several algorithms and techniques have been introduced in the literature for mining data represented in basket data format. The primary objective of these algorithms has been to improve the performance of the mining process. Unlike basket data representation, no algorithms exist for mining frequent itemsets and association rules in relational databases that are represented using the formal relational data model. Typical relational data can not be easily converted to basket data representation for the purpose of applying frequent itemset mining algorithms. Therefore, a need arises for algorithms that can *directly* be applied to data represented using the formal relational data model and for a conceptual framework for mining such data. This paper solves this problem by introducing an algorithm named *RDB-MINER* for mining frequent itemsets in relational databases.**

***Index Terms*—data mining, association rule, itemset, SQL, relational database.**

I. INTRODUCTION

The first algorithm for mining association rules, Apriori algorithm, was introduced in 1994 [1]. As stated in [1], the motivation behind introducing the Apriori algorithm was the progress that was made at that time in bar-code technology, which enabled retail supermarkets to store large quantities of sales data in their databases. The collected data was referred to as market basket data, or just *basket data*. Since then, numerous algorithms have been introduced [2-11]. These algorithms aimed at improving the performance as compared with the Apriori algorithm. However, these algorithms fall in the class of algorithms that are specialized in mining basket data. Basket data is represented as a set of records where each record consists of a transaction ID and a set of items bought in the transaction. Basket data representation does not conform to the relational data model since the normalization process does not allow muli-valued attributes to exist in a relational database [12]. Table 1 shows an example of market basket data.

In [13,14] we have addressed the problem of mining frequent itemsets (or frequent patterns) in regular relational databases that do not necessarily adhere to the specific format of basket data representation. We show below an example database relation (table) that we extract from [13] in order to demonstrate the feasibility and need for mining relational data that is not represented in basket data format.

TABLE 1
BASKET DATA FORMAT

Transaction_ID	Purchased_items
T200	{TV, Camera, Laptop}
T250	{Camera}
T300	{TV, Monitor, Laptop}

The relation, as shown in Table 2, contains data pertaining to ex-members of a gym club, which represents the data that is kept in the database for members who terminate their membership. This data includes AGE, GENDER, MEMBERSHIP_DURATION (how long a member maintained a valid membership in the club), HOME_DISTANCE (how far a member's residence is from the club location), and HOW_INTRODUCED (how a member was originally introduced to the club such as by referral or by seeing an advertisement in a newspaper).

Table 2 shows this relation as populated with sample data. In real life situations, a large club, with many branches, may have millions of ex-members, thus millions of tuples may exist in such a relation. Below are two patterns that exist in the data of Table 2 and that can be discovered by mining the data.

- Members who were introduced by referral tend to maintain their membership for longer periods, on average, than ones who came to know about the club through newspapers. Business managers may make use of this discovered knowledge by offering a discount to existing members if they persuade a friend or a relative to join the club.
- A pattern involving MEMBERSHIP_DURATION and HOME_DISTANCE can be discovered. According to the given data, most members who live close to the club tend to maintain their membership for a longer period than those who live far from the club location. The discovery of this pattern can be beneficial to business since the club manager can then launch a membership drive by going door-to-door to convince residents in the club neighborhood to join the club.

doi:10.4304/jsw.5.9.998-1005

Table 2
GYM_EX_MEMBERS

ID	AGE	GENDER	MEMBERSHIP DURATION	HOME_ DISTANCE	HOW_ INTRODUCED
1	young	f	Long	close	News_paper
2	middle	m	Short	far	News_paper
3	senior	f	Long	close	referral
4	senior	f	Long	close	referral
5	young	f	Long	far	News_paper
6	middle	m	Short	close	News_paper
7	senior	m	Short	far	News-paper
8	senior	f	Long	close	referral
9	young	f	Long	close	referral
10	middle	f	Long	far	News_paper
11	middle	m	Short	far	News_paper
12	senior	f	Long	close	referral
13	senior	m	Short	far	referral

Each of the above two patterns relates two attributes, therefore they can be referred to as *inter-attribute* patterns. Association patterns relating more than two attributes can also be discovered from the data. It is not straight forward neither practical to convert relational data as that of Table 2 to basket data representation whenever such data needs to be mined by applying one of the Apriori-like algorithms. A need exists for a mining algorithm that can be *directly* applied to relational data represented under the formal relational data model, which is what motivates our work.

In this paper, we introduce a new algorithm called *RDB_MINER* for mining relational databases represented using the relational data model as opposed to basket data format. Therefore, this algorithm can be viewed as representing a new class of mining algorithms that is orthogonal to the class of algorithms represented by the Apriori algorithm. Viewed from a different perspective, we can think of *RDB-MINER* as an algorithm that performs *inter-attribute* frequent itemset mining, whereas existing algorithms perform *intra-attribute* mining.

The remainder of this paper is organized as follows. In Section 2, we provide the necessary background by describing certain concepts that constitute a prerequisite to the introduction of the algorithm *RDB-MINER* which is to be described in Section 3. In Section 4, we describe some related issues: basically how the Apriori property [15] can be incorporated in *RDB-MINER* and how confidence of association rules can be computed after *RDB-MINER* is applied to a relation in a relational database. In Section 5, we discuss how *RDB-MINER* can be of wide applicability in several application domains. Conclusions are presented in Section 6.

II. BACKGROUND

In this section, we provide some background necessary for introducing the *RDB-MINER* algorithm. We describe the concepts of itemsets, itemset intension, association rule, and association rule intension, as presented in [13]. These definitions constitute the basis of our approach for mining relational databases.

A. Itemsets

Itemsets have been defined in data mining literature, but in the context of market basket data [1,15] and not based on the formal relational data model [12]. In [13], we recast this definition to the context of the relational model of data. An itemset in our approach is defined as *a set of items such that no two items belong to the same attribute (i.e, no two items are drawn from the same attribute domain, where a domain represents the set of valid values defined for an attribute)*. For example, in Table 2, {m, short, far} is a valid itemset (*IS*) while {m, short, far, close} is not a valid *IS* since 'far' and 'close' are two items that belong to the same attribute, which is HOME_DISTANCE. Stated formally, the following is the definition of a valid itemset.

$\{I_1, I_2,.. I_n\}$ *is valid IS*

$$\textit{iff } (\neg \exists I_j)\ (\neg \exists I_k)\ (j \neq k \wedge (Attr\,(I_j) = Attr\,(I_k))$$

Where *I* is an item from the relation (i.e. an attribute value) and *Attr (I)* is a function that returns the attribute name of item *I*. Logical AND is represented by "$\wedge$".

In Table 2, the domains of attributes are assumed, for simplicity, to be mutually exclusive. If these domains are not mutually exclusive, then one must qualify attribute values by their attribute names. Therefore, in this case, the itemset {short, news_paper} needs to be written as {MEMBERSHIP_DURATION.short, HOW_INTRODUCED.news_paper}. Note that, for clarity, throughout this paper, we use upper case letters for attribute names and lower case letters for attribute values. An itemset that contains k items is referred to as *k-itemset*.

The interestingness of an itemset is measured by the percentage of tuples in the relation that contain the itemset. This measure is referred to, in data mining literature, as *support*. In other words, the support is the probability *P* that the itemset exists in the relation.

$$Support\,(itemset) = P(itemset) = \frac{Num\ of\ tuples\ containing\ itemset}{Total\ Number\ of\ tuples} \times 100$$

The *support count,* on the other hand, is the absolute number of occurrences of an itemset in the relation.

As an example, based on the state shown in Table 2, the *support count* of the 3-itemset {young, f, referral} is 1 since there is only one tuple that contains this itemset. Its support = (1/13) X 100 = 7.7%. The support can be zero in case if the itemset does not exist at all in the relation, such as {m, long}. Normally, the user of a data mining tool supplies the minimum support *minsup* of interest. The data mining tool then finds the itemsets whose support is equal to or greater than *minsup*. Itemsets that satisfy the minimum support are referred to as *frequent* itemsets.

B. Itemset Intension

Following the terminology of the relational data model, the definition of *itemset intension* (*ISI*) was introduced in [13], and is summarized here. Such definition does not exist in the context of *market basket* data representation. An *itemset intension* (*ISI*) is a subset of the attributes of a relation. For example, in Table 2, {HOME_DISTANCE, MEMBERSHIP_DURATION} is an *ISI*. The itemsets that consist of actual attribute values belonging to these two attributes are instantiations of this itemset intension and are referred to as *itemset extensions* or simply *itemsets* (itemsets are described in Section 2.A). In Table 2, the itemsets that are instantiations of the *ISI*
{HOME_DISTANCE, MEMBERSHIP_DURATION}
are as follows:
{close, long}, {far, long}, {close, short}, {far, short}.

An itemset *IS* is said to be an instantiation of an itemset intension *ISI* if the cardinality of *IS* is the same as the cardinality of *ISI* and each item in *IS* is drawn from a domain of an attribute in *ISI*. Let the symbol "$\sqsubset$" denote "instantiation of" and let *CAR (S)* be a function that returns the cardinality of set S. We formally define the relationship between an itemset and its itemset intension as follows.

$$IS \sqsubset ISI \text{ iff } CAR(IS) = CAR(ISI) \text{ AND } (\forall I_j \in IS)(Attr(I_j) \in ISI)$$

"I" is an item in the itemset *IS* and *Attr (I)* returns the attribute name of item I. Note that the formal definition of *itemset*, as described in Section 2.A, prevents any two values in an itemset from belonging to the same attribute.

C. Association Rules

The association patterns among attribute values can be represented as association rules, where an association rule is an implication of the form:
lhs → *rhs,*
Each of the left had side (lhs) and right hand side (rhs) is a set of attribute values, provided that no attribute value exists in both lhs and rhs, i.e.,
$lhs \cap rhs = \Phi$.

For instance, {referral} → {long} is an association rule relating the attribute value
EMBERSHIP_DURATION.long to the attribute value HOW_INTRODUCED.referral. Each association rule has two metrics to measure its interestingness, *support* and *confidence*. The support of an association rule is the support of the itemset that contains all items in the rule, that is, the itemset containing the union of the items of the *lhs* and *rhs*. In other words,

$$Support\ (lhs \rightarrow rhs) = support\ (lhs \cup rhs) = P\ (lhs \cup rhs)$$

where P denotes Probability. As an example, to find the support of the rule {referral} → {long}, we note that 5 out of 13 tuples in the relation of Table 2 contain both referral and long, therefore,
Support (referral → long) = *Support* {referral, long} = (5/13) X 100 = 38.5%

Similar to basket data representation, we define the *confidence* of the rule (*lhs* → *rhs*) as the percentage of tuples that contain *rhs* from those that contain *lhs*. In other words, confidence is the conditional probability *P(rhs | lhs)*. *Confidence* can be expressed in terms of support as follows:

$$Confidence\ (lhs \rightarrow rhs) = \frac{support(lhs \cup rhs)}{support(lhs)} \times 100$$

In addition to specifying a *minsup*, a *minconf* (minimum confidence) can also provided to the data mining process, which then discovers association rules that satisfy *minsup* and *minconf.*

D. Association Rule Intension

In addition to introducing the concept of Itemset Intension in [13], we also introduce the concept of Association Rule Intension. Association Rule intension is a rule template that is shared by multiple association rules. Similar to an itemset intension, an association rule intension is expressed in terms of attribute names instead of actual data values. For example,
AGE → *MEMBERSHIP_DURATION*
is an association rule intension. The following association rules are possible instantiations of the above rule intension.

young → *long*	*young* → *short*
middle → *long*	*middle* → *short*
senior → *long*	*senior* → *short*

Generally, an association rule intension can be written as *LHS* → RHS where each of *LHS* and RHS represents a set of attribute names (hence, they are written in upper-case letters), provided that $LHS \cap RHS = \Phi$. An association rule of the form *lhs* → *rhs* (written in lower case letters) is said to be an *instantiation of* ($\sqsubset$) an association rule intension of the form *LHS* → *RHS if* $lhs \sqsubset LHS$ AND $rhs \sqsubset RHS$ (the symbol "$\sqsubset$" which stands for "instantiation of" is described in Section 2.B) . In this case we say that $(lhs \rightarrow rhs) \sqsubset (LHS \rightarrow RHS)$. In other words,

$$(lhs \rightarrow rhs) \sqsubset (LHS \rightarrow RHS) \text{ iff } (lhs \sqsubset LHS) \wedge (rhs \sqsubset RHS)$$

III. MINING ITEMSETS FROM RELATIONAL DATA

In this section, we introduce *RDB-MINER*, an algorithm for mining frequent itemsets using standard SQL and we demonstrate how the algorithm works on a detailed example.

Before introducing the algorithm, we first define *equi-cardinality subsets*, based on which the algorithm is defined. Let R be a relation with a set A of attributes. Let $\wp$ (A) be the powerset of A, whose elements are all possible subsets of A. If R has three attributes R (X, Y, Z), then

$\wp$ (A) = { {}, {X}, {Y}, {Y}, {X,Y}, {X,Z}, {Y, Z}, {X,Y,Z} }

Note that each of the sets in $\wp$ (A), with the exception of the empty set {}, represents an *itemset intension (ISI)*. Members of $\wp$ (A) can be divided into *equi-cardinality* subsets. An equi-cardinality subset is a subset of $\wp$ (A) in which every *ISI* has the same number of elements, i.e., has the same cardinality.

In the above example, we have four equi-cardinality subsets E_0, E_1, E_2 and E_3, where the subscript denotes the cardinality. These four equi-cardinality subsets are as shown below.

E_0 = {}
E_1 = { {X}, {Y}, {Y} }
E_2 = { {X,Y}, {X,Z}, {Y, Z} }
E_3 = { {X,Y,Z} }

In the algorithm that we introduce below, we ignore E_0 since it is empty. E_3 contains only one *ISI*, since the number of attributes of the relation is three, therefore this is the largest *ISI*.

We can express $\wp$ (A) in terms of its equi-cardinality subsets as follows. let E_c *, where* $0 \le c \le N$, be an equi-cardinality subset of $\wp$ (A), and N be the cardinality of A, i.e., the number of attributes of the relation R. $\wp$ (A) can be defined as follows.

$$P(A) = E_0 \cup E_1 .... \cup E_N = \bigcup_{c=0}^{c=N} E_c$$

In the remainder of this section, we first introduce algorithm *RDB-MINER* for computing the support count of itemsets. Next we give a general explanation of the algorithm, then demonstrate how it works by applying it to an abstract example

A. Algorithm RDB-MINER

Algorithm *RDB-MINER*
Input
R: a database relation
exclude_set: a subset of the attributes of R

0 *Begin*
1 ***Varchar*** *SQL_str (512)*
2 *Compute_N (N, R, exclude_set)*
3 *Compute_PowerSet (* $\wp$ (A) , R, exclude_set) *;*
4 ***For*** *c = 1 to N* ***do***
5 *Extract_*E_c *(*E_c *,* $\wp(A)$ *);*
/* $E_c \subset \wp(A)$ *and each ISI* $\in E_c$ *has a cardinality of c.* */
6 ***For*** *each itemset intension ISI* $\in E_c$ ***do***
7 *Generate_SQL (SQL_Str, ISI, Relation_Name);*
8 *Execute SQL_Str;*
9 *SQL_str = "";*
10 *End*
11 *End*
12 *End*

B. General Description of the Algorithm

Line 1 of the algorithm declares a variable called *SQL_str*, which is used to hold the SQL statement to be generated by the algorithm. Line 2 calls the procedure *compute_N,* that returns N, the number of attributes of relation R after excluding the attributes in *exclude_set*. The input arguments to *Compute_N* are relation R and *exclude_set*. *Exclude_set* is the set of attributes (normally primary key attributes) to be excluded from the computation of N. *Exclude_set* is left empty if all the attributes of R are to be included in the computation. *Compute_powerset* procedure in line 3 returns the power set, $\wp$ (A), of relation R after excluding the attributes of *exclude_set*. Line 5 extracts the equi-cardinality subset E_c from $\wp$ (A). For example, in the 2nd iteration of the `for` loop, where c=2, this step returns the subset E_2 = { {X,Y}, {X,Z}, {Y, Z} }, assuming we have the three attributes X, Y, and Z. The `for` loop between lines 6 and 10 extracts the *ISI*s from the equi-cardinality subset and generates and executes a SQL statement that computes the *support* of each such itemset. For instance, for E_2, a SQL query is generated and executed for each of the *ISI*s: {X,Y}, {X,Z} and {Y, Z}.

C. Applying the Algorithm to an Abstract Example

To demonstrate with some detail how algorithm *RDB-MINER* works, we explain the steps it goes through when applied to the relation R2 (RID, A,B,C) shown in Table 3. The primary idea is to use the 'GROUP BY' clause of SQL along with the COUNT aggregate function in the generated SQL statement to compute the support count of itemsets.

The first input to the algorithm is relation schema R2, the relation shown in Table 3. The second input is the *exclude_set*, which in this case contains only one attribute (the primary key attribute RID) but it can contain multiple attributes depending on what the user wants to exclude. Before the first `for` loop starts, *Compute_N* is executed with the arguments N, R2(RID,A,B,C) , and {RID}, where {RID} is the *exclude_set*. The procedure returns N = 3, the number of attributes in the relation after excluding the *exclude_set*. *Compute_Powerset* computes $\wp$ (A), where A is the set of attributes after excluding the members of *exclude_set.* The value of $\wp$ (A) that is returned is:

$\wp$ (A) = { {}, {,A}, {B}, {C}, {A,B}, {A,C}, {B,C}, {A,B,C} }

Each iteration of the outer `for` loop that starts at line 4 extracts an equi-cardinality subset E_c from $\wp(\mathcal{A})$. Each iteration of the inner `for` loop that starts at line 6 extracts an itemset intension *ISI* from E_c and generates and executes a SQL statement for that *ISI*. The generated SQL query computes the support count for a set of itemsets at once. Therefore, the net effect of the two nested `for` loops is to first compute 1-itemsets along with their *support count*. Next, the support count values of 2-itemsets are computed, followed by computing the support count of the 3-itemset {A,B,C}.

Table 3
Relation R2

RID	A	B	C
1	a1	b1	c1
2	a2	b2	c1
3	a3	b1	c2
4	a4	b2	c2
5	a1	b1	c1
6	a2	b2	c1
7	a3	b1	c2
8	a4	b2	c2
9	a5	b1	c1
10	a5	b3	c3
11	a5	b4	c3
12	a6	b4	c3

D. Generation of SQL Statements

The general format of the SQL statements that is generated by the procedure Generate_SQL is:

SELECT <list of attributes in ISI>,
count (*) as sup_count
FROM <relation_name>
GROUP BY <list of attributes in ISI>

Where *sup_count* represents the support count.

In the first iteration of the outer `for` loop, E_1 = { {A}, {B}, {C} } is computed. The inner loop then generates and executes a SQL query for each of the members of E_1. In the first iteration of the inner loop, the following query is generated and executed for the first *ISI*, which is {A}.

Query Q1
SELECT A, count (*) as sup_count
FROM R2
GROUP BY A

The result of this query is shown in Table 4.

Table 4
Q1_Result

A	sup_count
a1	2
a2	2
a3	2
a4	2
a5	3
a6	1

Table 5
Q2_Result

B	sup_count
b1	5
b2	4
b3	1
b4	2

Table 6
Q3_Result

C	sup_count
c1	5
c2	4
c3	3

Similarly, the second and third iterations of the inner loop generate the following two queries that, when executed, return 1-itemsets containing B values and C values, respectively, along with their support counts.

Query Q2
SELECT B, count (*) as sup_count
FROM R2
GROUP BY B

Query Q3
SELECT C, count (*) as sup_count
FROM R2
GROUP BY C

The results of the above two queries are shown in Table 5 and Table 6, respectively.

In the second iteration of the outer `for` loop, the equi-cardinality subset E_2 = { {A,B}, {A,C}, {B,C} } is computed. The inner `for` loop generates and executes a SQL query for each member of E_2, in other words, for each *ISI* in E_2. As an example, below is the query generated for {B,C}.

Query Q4
SELECT B,C, Count (*) as sup_count
FROM R2
GROUP BY B, C

Table 7
Q4_Result

B	C	sup_count
b1	c1	3
b2	c1	2
b1	c2	3
b2	c2	2
b3	c3	1
b4	c3	2

The resulting itemsets, whose intension is *ISI* = {B,C}, along with their support counts are shown in Table 7. Similar queries are generated for the two itemset intensions: {A,B} and {A,C}.

The third and final iteration of the outer `for` loop, computes the equi-cardinality subset E_3 = { {A,B,C} }, and the inner `for` loop generates and executes the following query for the ISI {A,B,C}.

Query Q5
SELECT A,B,C, Count (*) as sup_count
FROM R2
GROUP BY A,B,C

The resulting 3-itemsets and their support count values are shown in Table 8.

C. Finding Frequent Itemsets

To find *frequent* itemsets (i.e., itemsets whose support count is equal or above a minimum threshold), the procedure Genreate_SQL can be implemented to add a `Having` clause to the generated query in order to filter out infrequent itemsets. Assuming a minimum support count of 2, the generated query that computes frequent itemsets whose intension is {B} would be as follows.

Table 8
Q5_Result

A	B	C	sup_count
a1	b1	c1	2
a2	b2	c1	2
a3	b1	c2	2
a4	b2	c2	2
a5	b1	c1	1
a5	b3	c3	1
a5	b4	c3	1
a6	b4	c3	1

Query Q6
SELECT B, count (*) as sup_count
FROM R2
GROUP BY B
HAVING count (*) ≥ 2

Table 9 shows the result of this query. The difference between Table 9 and Table 5 is that all rows below the minimum support count are filtered out from the result.

Table 9
Q6_Result

B	sup_count
b1	5
b2	4
b4	2

IV. RELATED ISSUES

A. Considering the Apriori Property

In our approach of using SQL to mine for frequent itemsets, the Apriori property [15] can be incorporated in the computation process.

The Apriori property, in effect, states that any superset of an infrequent itemset must be infrequent and any subset of a frequent itemset must be frequent. This means that if a k-itemset s_1 is found to be infrequent, then there is no need to compute the support count of any (k+1)-itemset that is a superset of s_1 since it is guaranteed to be infrequent.

This property has been incorporated in many existing data mining algorithms to eliminate unneeded computations and therefore improve the performance. We can easily incorporate the Apriori property in the computation process of *RDB-MINER* by including a Where clause in the generated SQL query. The purpose of the Where statement is to filter out any itemsets that have infrequent subsets.

Assume that the minimum support count threshold is 2. Itemset {b3} in Table 5 is infrequent since its support count is 1. According to the Apriori property, any itemset that is a superset of {b3} is infrequent, and can be excluded from the result. Therefore, Query Q4 that computes the support count of itemsets whose intension is {B, C} can be modified to filter out any itemsets of the form {B, C} if the support count of either the {B} component or the {C} component is below 2. Query Q7 below is a modified version of Query Q4 that takes the Apriori property into consideration.

Query Q7

```
SELECT B,C, Count (*) as sup_count
FROM R2
WHERE B in (SELECT B FROM Q2_Result
                  WHERE sup_count ≥  2)
      AND C in (SELECT C FROM Q3_Result
                  WHERE sup_count ≥ 2)
GROUP BY B, C
```

In Query Q7, Q2_Result and Q3_Result are the two relations shown in Table 5 and Table 6, respectively. Of course, for Query Q7 to be possible, Q2_Result and Q3_Result should persist in the database. This can be performed by altering the Generate_SQL function in *RDB-MINER* algorithm to make it add a "CREATE TABLE … AS …" clause to the generated query. This requires a systematic naming convention to name the persisting relatios so that generated SQL queries can easily refer to them in the Where clause.

B. Computing Confidence

Confidence of a rule of the format *lhs* → *rhs* can be computed as described in Section 2 using the following formula.

Confidence (*lhs* → *rhs*) =
(*support* {*lhs* U *rhs*} / *support* {*lhs*}) X 100

To find the confidence of an association rule such as *Confidence (b1 → c1),* we need to get the support of {*lhs* U *rhs*}, which is {b1, c1} and the support of {*lhs*}, which is {b1} from the relevant result tables and substitute in the formula above, as follows.

Confidence (b1 → c1) =
(support {b1, c1} / support {b1}) X 100
= (3/5) X 100 = 60%

The support count of {b1,c1} shown above is taken from Table 7, whereas the support count of {b1} is taken from Table 5.

V. PRACTICAL APPLICATIONS

In this section, we drive the point home by showing some practical applications in which our approach of using *RDB-MINER* to perform inter-attribute frequent itemset mining can be very useful. The first application is the one shown in Table 2 for a gym club. Below we show how *RDB-MINER* can arrive to the two patterns described in section 1.

In the first iteration of *RDB-MINER*, it computes the support of 1-itemsets. The two 1-itemset intentions that are needed in the computation of the two patterns described in Section 1 are {HOW-INTRODUCED} and {HOME-DISTANCE}. Below are the two SQL queries that are generated by the algorithm to compute the support of itemsets that belong to these two itemset intentions.

Query Q8

```
SELECT HOW_INTRODUCED,
        COUNT (*) as supp_count
FROM GYM_EX_MEMBERS
GROUP BY HOW_INTRODUCED
```

Query Q9

```
SELECT HOME_DISTANCE,
        COUNT (*) as supp_count
FROM GYM_EX_MEMBERS
GROUP BY HOME_DISTANCE
```

The results of these two queries are shown in Table 10 and Table 11, respectively.

Table 10:
Q8_Result

HOW_ INTRODUCED	Supp_count
News_paper	7
referral	6

Table 11:
Q9_Result

HOME_DISTANCE	supp_count
close	7
far	6

In the next iteration, the algorithm computes the support of all 2-itemsets. It generates several SQL queries to perform this task. The two 2-itemset intensions that we need to show are {HOW-INTRODUCED, MEMBERSHIP_DURATION} and {HOME-DISTANCE, EMBERSHIP_DURATION}. Below are the two SQL queries that the algorithm generates and that compute the support of all 2-itemsets that are instantiations of the above two itemset intentions.

<u>Query Q10</u>

```
SELECT HOW_INTRODUCED,
        MEMBERSHIP_DURATION,
        COUNT (*) as supp_count
FROM GYM_EX_MEMBERS
GROUP BY HOW_INTRODUCED,
MEMBERSHIP_DURATION
```

<u>Query Q11</u>

```
SELECT HOME_DISTANCE,
        MEMBERSHIP_DURATION,
        COUNT (*) as supp_count
FROM GYM_EX_MEMBERS
GROUP BY HOME_DISTANCE,
MEMBERSHIP_DURATION
```

The results generated by these two queries are shown in Table 12 and Table 13, respectively.

Table 12:
Q10_Result

HOW_ INTRODUCED	**MEMBERSHIP DURATION**	**Supp_count**
news_paper	long	3
news_paper	short	4
referral	long	5
referral	short	1

Table 13:
Q8_Result

HOME_ DISTANCE	**MEMBERSHIP DURATION**	**Supp_count**
Close	long	6
Far	short	4
Far	long	2
Close	short	1

There are four association rules of the form:
HOW_INTRODUCED → MEMBERSHIP_DURATION

These four association rules along with their support (s) and confidence (c) values are:

news_paper → long (s= 3 , c= 3/7)
news_paper → short (s= 4 , c= 4/7)
referral → long (s= 5 , c= 5/6)
referral → short (s= 1 , c= 1/6)

The confidence values are computed from the support values as described in Section 4. The third of these association rules has the highest confidence and is well-supported. This rule means that members who were introduced by referral tend to maintain their membership for longer periods, on average, than ones who came to know about the club through newspapers. This discovered piece of knowledge can be of benefit to the business.

Similarly, we have four association rules of the form:
HOME_DISTANCE → MEMBERSHIP_DURATION
These four association rules are

close → long (s= 6, c= 6/7)
far → short (s= 4, c= 4/6)
far → long (s= 2, c= 2/6)
close → short (s= 1, c= 1/7)

The first of these rules has the highest confidence and is well-supported, therefore it represents a discovered knowledge that is valuable to the business: that members who live close to the club tend to maintain their membership for longer periods.

Our approach of using *RDB-MINER* as a SQL-based algorithm for mining relational data that are not necessarily represented in basket data format can be very useful in many other application domains such as banking, education, and health care, to name a few.

For example, in an education database (university database) there can be a STUDENT relation as shown in Table 14.

Table 14:
STUDENT

ID	**HIGH SCHOOL AVERAGE**	**FAMILY INCOME**	**COLLEGE AVERAGE**	**SAT SCORE RANGE**
100	C	20K – 30K	B	500-600
101	B	20K – 30K	B	600-700
102	C	30K – 40K	A	700-800
…	…	…	…	…

Mining such data using *RDB-MINER*, may discover interesting inter-attribute association rules between HIGH_SCHOOL_AVERAGE grade and COLLEGE_AVERAGE grade and also between SAT_SCORE_RANGE and COLLEGE_AVERAGE. The discovered knowledge can guide a college in recruiting future students, by deciding weather to give more weight to high school grade or to the SAT score in evaluating applicants. Other association rules between FAMILY_INCOME and COLLEGE_AVERAGE may also be discovered. This could be of benefit to social studies that assess the impact of social and financial status of students on their college performance.

VI. CONCLUSION

Mining frequent itemsets from data represented using basket data format has received a lot of attention from researchers in the past. However, a need exists for mining relational databases that are not represented in basket data format in order to discover knowledge that could be berried in these databases. One way is to go through complex and time-consuming conversions to convert relational data to basket data representation before applying a mining algorithm. In many cases, typical relational data can not be easily mapped to basket data representation. In this paper, we introduced an algorithm called *RDB-MINER* that can be used to directly mine relational databases without having to resort to any conversions prior to starting the mining process. A second advantage of *RDB-MINER* is that it is portable (i.e., it can be applied to any relational database) because it is based on SQL, which is the *standard* interface to relational databases.

ACKNOWLEDGMENT

The author would like to thank the Applied Science University in Amman, Jordan, for supporting this publication.

REFERENCES

[1] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules,” Proceedings of the International Conf. on Very Large Databases (VLDB’94), 1994, pages 487-499, Santiago, Chile.

[2] J. Wang, J. Han, Y. Lu, and P. Tzvetkov. “TFP: An effcient algorithm for mining top-k frequent closed itemsets.” *IEEE Trans. Knowledge and Data Engineering*, 2005, 17:652-664.

[3] D. Xin, J. Han, X. Yan, and H. Cheng. “Mining compressed frequent-pattern sets.” In *Proc. 2005 Int.Conf. Very Large Data Bases (VLDB'05)*, pages 709-720, Trondheim, Norway, Aug. 2005.

[4] B. Rozenberg, E. Gudes, “Association rules mining in vertically partitioned databases,” Data and Knowledge Engineering 59(2) 2006. Pages: 378–396.

[5] Mohammad-Hossein Nadimi-Shahraki, Norwati Mustapha, Md Nasir Sulaiman, Ali B. Mamat, “A New Algorithm for Discovery Maximal Frequent Itemsets,” International Conference on Data Mining (DMIN) 2008:309-312

[6] B. Lucchese, S. Orlando, R. Perego, “Fast and memory efficient mining of frequent closed itemsets,” IEEE Transactions on Knowledge and Data Engineering 18 (1). 2006, 21–36.

[7] H. Moonestinghe, S. Fodeh, P.N. Tan, “Frequent closed itemset mining using prefix graphs with an efficient flow-based pruning strategy,” Proceedings of the 6th International Conference on Data Mining, Hong Kong, 2006, pp. 426–435.

[8] Lisheng Ma, Yi Qi, “An Efficient Algorithm for Frequent Closed Itemsets Mining,” International Conference on Computer Science and Software Engineering (CSSE) 2008:259-262

[9] G. Liu, H. Lu, Y. Xu, and J. X. Yu, “Ascending Frequency Ordered Prefix-tree: Efficient Mining of Frequent Patterns,” Proc. 2003 Int. Conf. on Database Systems for Advanced Applications (DASFAA03), Kyoto, Japan, March, 2003.

[10] J.Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation” In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, pages 1-12, Dallas, TX, May 2000.

[11] K. Gouda, and M. Zaki, “GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets,” Data Mining and Knowledge Discovery: An International Journal, 2005, 11(3): 223-242.

[12] R. Elmasri, and S. Navathe. *Fundamentals of Database Systems*, *Fifth Edition,* Addison-Wesley, 2007.

[13] A. Alashqur, “Expressing Database Functional Dependencies in Terms of Association Rules,” European Journal of Scientific Research (EJSR). Vol. 32, Issue 2, 2009.

[14] A. Alashqur, “Mining Association Rules: A Database Perspective” International Journal of Computer Science and Network Security IJCSNS, VOL.8 No.12, December 2008, ISSN 1738-7906.

[15] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. 2006. Addison Wesley Publisher

Abdallah Alashqur obtained his Masters and Ph.D. degrees from the University of Florida in 1985 and 1989, respectively. Between 1989 and 2006 he worked in the IT field for several corporations in the USA. Since 2006 he has been a faculty member in the Faculty of Information Technology at Applied Science University in Amman, Jordan, where he is currently serving as the dean of the faculty. His research interests include data mining, database systems, and artificial intelligence.