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 cannot 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 multi-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>
<thead>
<tr>
<th>Transaction_ID</th>
<th>Purchased_Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T200</td>
<td>{TV, Camera, Laptop}</td>
</tr>
<tr>
<td>T250</td>
<td>{Camera}</td>
</tr>
<tr>
<td>T300</td>
<td>{TV, Monitor, Laptop}</td>
</tr>
</tbody>
</table>

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.
Each of the above two patterns relates two attributes, therefore they can be referred to as \textit{inter-attribute} patterns. Association patterns relating more than two attributes can also be discovered from the data. It is not straightforward 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 \textit{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 \textit{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 \textit{RDB-MINER} as an algorithm that performs \textit{inter-attribute} frequent itemset mining, whereas existing algorithms perform \textit{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 \textit{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 \textit{RDB-MINER} and how confidence of association rules can be computed after \textit{RDB-MINER} is applied to a relation in a relational database. In Section 5, we discuss how \textit{RDB-MINER} can be of wide applicability in several application domains. Conclusions are presented in Section 6.

\section*{II. BACKGROUND}

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

\subsection*{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 \textit{set of items} such that no two items belong to 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, 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.

\begin{equation}
\{I_1, I_2, \ldots, I_k\} \text{ is valid IS } \iff \left( \neg \exists I_j \left( \neg \exists I_k \left( j \neq k \land \text{Attr}(I_j) = \text{Attr}(I_k) \right) \right) \right)
\end{equation}

Where \( I \) is an item from the relation (i.e. an attribute value) and \text{Attr}(I) is a function that returns the attribute name of item \( I \). Logical AND is represented by \textit{“\&"}.

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 \{\text{MEMBERSHIP}'DURATION, \text{HOME.Distance}, \text{HOW INTRODUCED}, \text{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 \textit{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 \textit{support}. In other words, the support is the probability \( P \) that the itemset exists in the relation.

\begin{equation}
\text{Support(itemset)} = \frac{\text{Num of tuples containing itemset}}{\text{Total Number of tuples}} \times 100
\end{equation}

The \textit{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 \textit{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 \textit{minsup} of interest. The data mining tool then finds the itemsets whose support is equal to or greater than \textit{minsup}. Itemsets that satisfy the minimum support are referred to as \textit{frequent} itemsets.

### B. Itemset Intension

Following the terminology of the relational data model, the definition of \textit{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 \textit{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 \textit{itemset extensions} or simply \textit{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 \textit{IS} is said to be an instantiation of an itemset intension \textit{ISI} if the cardinality of \textit{IS} is the same as the cardinality of \textit{ISI} and each item in \textit{IS} is drawn from a domain of an attribute in \textit{ISI}. Let the symbol “\textit{∈}” denote “instantiation of” and let \textit{CAR (S)} be a function that returns the cardinality of set \textit{S}. We formally define the relationship between an itemset and its itemset intension as follows.

\[ IS \subseteq ISI \text{ iff } CAR(IS) = \]

\[ CAR(ISI) \land (\forall I \in ISI) \text{ Attr (I) } \subseteq IS \]

“If” is an item in the itemset \textit{IS} and \textit{Attr (I)} returns the attribute name of item \textit{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:

\[ lhs \rightarrow rhs, \]

Each of the left hand 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.,

\[ lhs \cap rhs = \emptyset. \]

For instance, \{referral\} \rightarrow \{long\} is an association rule relating the attribute values MEMBERSHIP\_DURATION, long to the attribute value HOW\_INTRODUCED, referral. Each association rule has two metrics to measure its interestingness, \textit{support} and \textit{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 \textit{lhs} and \textit{rhs}. In other words,

\[ \text{Support} (lhs \rightarrow rhs) = \frac{\text{support}(lhs \cup rhs)}{\text{support}(lhs)} \times 100 \]

where \( P \) denotes Probability. As an example, to find the support of the rule \{referral\} \rightarrow \{long\}, we note that 5 out of 13 tuples in the relation of Table 2 contain both referral and long, therefore,

\[ \text{Support} (\text{referral} \rightarrow \text{long}) = \text{Support} \{\text{referral}, \text{long}\} = (5/13) \times 100 = 38.5\% \]

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

\[ \text{Confidence} (lhs \rightarrow rhs) = \frac{\text{support}(lhs \cup rhs)}{\text{support}(lhs)} \times 100 \]

In addition to specifying a \textit{minsup}, a \textit{minconf} (minimum confidence) can also be provided to the data mining process, which then discovers association rules that satisfy \textit{minsup} and \textit{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, \textit{AGE} \rightarrow \textit{MEMBERSHIP\_DURATION} is an association rule intension. The following association rules are possible instantiations of the above rule intension.

\[ \text{young} \rightarrow \text{long} \quad \text{young} \rightarrow \text{short} \]

\[ \text{middle} \rightarrow \text{long} \quad \text{middle} \rightarrow \text{short} \]

\[ \text{senior} \rightarrow \text{long} \quad \text{senior} \rightarrow \text{short} \]

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

\[ (lhs \rightarrow rhs) \subseteq (LHS \rightarrow RHS) \quad \text{iff} \quad (lhs \subseteq LHS) \land (rhs \subseteq RHS) \]

### III. MINING ITEMSETS FROM RELATIONAL DATA

In this section, we introduce \textit{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 \( \text{equi-cardinality subsets} \), based on which the algorithm is defined. Let \( R \) be a relation with a set \( A \) of attributes. Let \( \mathcal{P}(A) \) be the powerset of \( A \), whose elements are all possible subsets of \( A \). If \( R \) has three attributes \( R \) (X, Y, Z), then
\[
\mathcal{P}(A) = \{ \{\}, \{X\}, \{Y\}, \{X,Y\}, \{X,Z\}, \{Y, Z\}, \{X,Y,Z\} \}
\]
Note that each of the sets in \( \mathcal{P}(A) \), with the exception of the empty set \( \{\} \), represents an itemset intension (ISI). Members of \( \mathcal{P}(A) \) can be divided into equi-cardinality subsets. An equi-cardinality subset is a subset of \( \mathcal{P}(A) \) 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.

\[
\begin{align*}
E_0 &= \{\} \\
E_1 &= \{\{X\}, \{Y\}, \{Y\}\} \\
E_2 &= \{\{X,Y\}, \{X,Z\}, \{Y, Z\}\} \\
E_3 &= \{\{X,Y,Z\}\}
\end{align*}
\]

In the algorithm that we introduce below, we ignore \( E_0 \) since it is empty. \( E_1 \) contains only one ISI, since the number of attributes of the relation is three, therefore this is the largest ISI.

We can express \( \mathcal{P}(A) \) in terms of its equi-cardinality subsets as follows. Let \( E_c \), where \( 0 \leq c \leq N \), be an equi-cardinality subset of \( \mathcal{P}(A) \), and \( N \) be the cardinality of \( A \), i.e., the number of attributes of the relation \( R \). \( \mathcal{P}(A) \) can be defined as follows.

\[
P(A) = E_0 \cup E_1 \cup \ldots \cup E_N = \bigcup_{c=0}^{c=N} E_c
\]

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

\( \text{exclude set}: \) a subset of the attributes of \( R \)

0 Begin
1 \textbf{VarChar} SQL_str (512)
2 Compute \( N \) (\( N, R, \text{exclude set} \))
3 Compute_Powerset (\( \mathcal{P}(A), R, \text{exclude set} \))
4 \textbf{For} \( c = 1 \) to \( N \) \textbf{do}
5 \hspace{1em} Extract \( E_c \) \( E_c = \mathcal{P}(A) \) \text{ and each ISI } \in E_c \text{ has a cardinality of } c. \)
6 \hspace{1em} For each itemset intension ISI \( \in E_c \) \textbf{do}
7 \hspace{2em} Generate SQL (SQL_Str, ISI, Relation_Name);
8 \hspace{2em} Execute SQL Str;
9 \hspace{2em} 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 \( N \), which 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 powerset, \( \mathcal{P}(A) \), of relation \( R \) after excluding the attributes of exclude_set. Line 5 extracts the equi-cardinality subset \( E_c \) from \( \mathcal{P}(A) \). For example, in the 2\textsuperscript{nd} 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 ISIs 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 ISIs: \{\{X,Y\}, \{X,Z\}\} and \{\{X,Y,\}, \{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 \( N, R2(\text{RID, A,B,C}) \), and \{\{\text{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 \( \mathcal{P}(A) \), where \( A \) is the set of attributes after excluding the members of exclude_set. The value of \( \mathcal{P}(A) \) that is returned is:

\[
\mathcal{P}(A) = \{\{\}, \{A\}, \{B\}, \{C\}, \{A,B\}, \{A,C\}, \{B,C\}, \{A,B,C\}\}
\]
Each iteration of the outer for loop that starts at line 4 extracts an equi-cardinality subset $E_i$ from $\mathcal{P}$ (A). Each iteration of the inner for loop that starts at line 6 extracts an itemset intension ISI from $E_i$ 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 3-itemsets along with their support count. The result of this query is shown in Table 4.

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

\[
\text{SELECT} \ <\text{list of attributes in ISI}>\text{,}
\text{count (*) as sup_count}
\text{FROM} \ <\text{relation_name}>
\text{GROUP BY} \ <\text{list of attributes in ISI}>
\]

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

\[
\text{SELECT} \ A, \ \text{count (*) as sup_count}
\text{FROM} \ R2
\text{GROUP BY} \ A
\]

The result of this query is shown in Table 4.

<table>
<thead>
<tr>
<th>A</th>
<th>sup_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>2</td>
</tr>
<tr>
<td>a2</td>
<td>2</td>
</tr>
<tr>
<td>a3</td>
<td>2</td>
</tr>
<tr>
<td>a4</td>
<td>2</td>
</tr>
<tr>
<td>a5</td>
<td>3</td>
</tr>
<tr>
<td>a6</td>
<td>1</td>
</tr>
</tbody>
</table>

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

\[
\text{SELECT} \ B, \ \text{count (*) as sup_count}
\text{FROM} \ R2
\text{GROUP BY} \ B
\]

Table 4

<table>
<thead>
<tr>
<th>A</th>
<th>sup_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>b1</td>
<td>5</td>
</tr>
<tr>
<td>b2</td>
<td>4</td>
</tr>
<tr>
<td>b3</td>
<td>1</td>
</tr>
<tr>
<td>b4</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5

<table>
<thead>
<tr>
<th>B</th>
<th>sup_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>5</td>
</tr>
<tr>
<td>c2</td>
<td>4</td>
</tr>
<tr>
<td>c3</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 6

<table>
<thead>
<tr>
<th>C</th>
<th>sup_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1 c1</td>
</tr>
<tr>
<td>a2</td>
<td>b2 c1</td>
</tr>
<tr>
<td>a3</td>
<td>b1 c2</td>
</tr>
<tr>
<td>a4</td>
<td>b2 c2</td>
</tr>
<tr>
<td>a5</td>
<td>b3 c3</td>
</tr>
<tr>
<td>a6</td>
<td>b4 c3</td>
</tr>
</tbody>
</table>

Query Q3

\[
\text{SELECT} \ C, \ \text{count (*) as sup_count}
\text{FROM} \ R2
\text{GROUP BY} \ C
\]

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

\[
\text{SELECT} \ B, \ C, \ \text{Count (*) as sup_count}
\text{FROM} \ R2
\text{GROUP BY} \ B, \ C
\]

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

\[
\text{SELECT} \ A, \ B, \ C, \ \text{Count (*) as sup_count}
\text{FROM} \ R2
\text{GROUP BY} \ A, \ B, \ C
\]

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>sup_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>c1</td>
<td>2</td>
</tr>
<tr>
<td>a2</td>
<td>b2</td>
<td>c1</td>
<td>2</td>
</tr>
<tr>
<td>a3</td>
<td>b1</td>
<td>c2</td>
<td>2</td>
</tr>
<tr>
<td>a4</td>
<td>b2</td>
<td>c2</td>
<td>2</td>
</tr>
<tr>
<td>a5</td>
<td>b3</td>
<td>c3</td>
<td>1</td>
</tr>
<tr>
<td>a6</td>
<td>b4</td>
<td>c3</td>
<td>1</td>
</tr>
</tbody>
</table>

C. Finding Frequent Itemsets

To find frequent itemsets (i.e., itemsets whose support count is equal or above a minimum threshold), the procedure Generate_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.

Query Q6

\[
\text{SELECT} \ B, \ \text{count (*) as sup_count}
\text{FROM} \ R2
\text{GROUP BY} \ B
\text{HAVING} \ \text{count (*)} \geq 2
\]
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.

### 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**

```sql
SELECT B, C, COUNT(*) as sup_count
FROM R2
WHERE B in (SELECT B FROM Q2_Result WHERE sup_count \geq 2)
AND C in (SELECT C FROM Q3_Result WHERE sup_count \geq 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 relations so that generated SQL queries can easily refer to them in the WHERE clause.

#### B. Computing Confidence

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

$$
Confidence (lhs \rightarrow rhs) = \left( \frac{support \{lhs U rhs\}}{support \{lhs\}} \right) \times 100
$$

To find the confidence of an association rule such as $Confidence (b1 \rightarrow c1)$, 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.

$$
Confidence (b1 \rightarrow c1) = \left( \frac{support \{b1, c1\}}{support \{b1\}} \right) \times 100
= \left( \frac{3}{5} \right) \times 100 = 60\%
$$

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**

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

**Query Q9**

```sql
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.
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.

**Query Q10**
SELECT HOW_INTRODUCED,
MEMBERSHIP_DURATION,
COUNT (*) as supp_count
FROM GYM_EX_MEMBERS
GROUP BY HOW_INTRODUCED,
MEMBERSHIP_DURATION

**Query Q11**
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>
<thead>
<tr>
<th>HOW_INTRODUCED</th>
<th>MEMBERSHIP_DURATION</th>
<th>Supp_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>news_paper</td>
<td>long</td>
<td>7</td>
</tr>
<tr>
<td>referral</td>
<td>short</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HOME_DISTANCE</th>
<th>MEMBERSHIP_DURATION</th>
<th>Supp_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>close</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>far</td>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>

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

- \(\text{news_paper} \rightarrow \text{long}\) \((s=3, c=3/7)\)
- \(\text{news_paper} \rightarrow \text{short}\) \((s=4, c=4/7)\)
- \(\text{referral} \rightarrow \text{long}\) \((s=5, c=5/6)\)
- \(\text{referral} \rightarrow \text{short}\) \((s=1, c=1/6)\)

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:
\(\text{HOME_Distance} \rightarrow \text{MEMBERSHIP_Duration}\)
These four association rules are

- \(\text{close} \rightarrow \text{long}\) \((s=6, c=6/7)\)
- \(\text{far} \rightarrow \text{short}\) \((s=4, c=4/6)\)
- \(\text{far} \rightarrow \text{long}\) \((s=2, c=2/6)\)
- \(\text{close} \rightarrow \text{short}\) \((s=1, c=1/7)\)

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>
<thead>
<tr>
<th>ID</th>
<th>HIGH SCHOOL AVERAGE</th>
<th>FAMILY INCOME</th>
<th>COLLEGE AVERAGE</th>
<th>SAT SCORE RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>C</td>
<td>20K – 30K</td>
<td>B</td>
<td>500-600</td>
</tr>
<tr>
<td>101</td>
<td>B</td>
<td>20K – 30K</td>
<td>B</td>
<td>600-700</td>
</tr>
<tr>
<td>102</td>
<td>C</td>
<td>30K – 40K</td>
<td>A</td>
<td>700-800</td>
</tr>
</tbody>
</table>

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 whether 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 buried 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
[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.

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.