CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter discusses the challenges and issues in RFID (Radio-Frequency Identification) based data acquisition systems. Prior work relevant to the research in elimination of RFID data redundancy and retrieving relevant information about customer movement path sequences in data mining is reviewed. The related work proposed by authors on frequent pattern mining is also discussed in this chapter.

2.2 CHALLENGES IN RFID DATA

RFID technology is usually used in large scale data acquisition applications. The huge amount of captured RFID based data poses several challenges in mining and analysis.

The challenges are:

- The volume of data associated with RFID will be extremely large, because of the large number of tags which may be tracked by a single reader.

- The data can be noisy and redundant, with many tags being completely dropped, and others being read by multiple readers at multiple instants, resulting in redundancy in the observations.
In addition to this the large volume of the data makes the process of cleaning much more challenging. Therefore, effective methods need to be designed to compress and clean such data. The cleaning process is expensive and challenging.

2.3 CLASSIFICATION OF RFID READINGS

Reliability of RFID readings is of concern in many circumstances (Brusey et al 2003, Vogt et al 2002). There are three types of readings

1. False Negative Readings
2. False Positive Readings
3. Duplicate Readings

2.3.1 False Negative Readings

The RFID tags might not be read by the reader at all. This can be caused

- When multiple tags are to be simultaneously detected, RF collisions occur and signals interfere with each other, preventing the reader from identifying any tags
- Due to water or metal shielding or RF interference.

2.3.2 False Positive Readings (or noise)

In addition to the Tags to be read, additional and unexpected readings are generated and these readings are called False Positive Readings. This can be caused due to following reasons.
- RFID tags outside the normal reading scope of a reader are captured by the reader. For example, while reading items from one area, a reader may read items from an adjacent area.

- Unknown reasons from the reader or environment.

### 2.3.3 Duplicate Readings

A reading is defined as duplicate when it is repeated and does not deliver new information to the system. Duplicate Readings can be caused by the following reasons:

- Tags in the scope of a reader for a long time (in multiple reading frames) are read by the reader multiple times;

- Multiple readers are installed to cover larger area or distance, and tags in the overlapped areas are read by multiple readers.

- To enhance reading accuracy, multiple tags with same EPCs are attached to the same object, thus generating duplicate readings.

### 2.4 RFID DATA FILTERING

The problem of duplicate readings in RFID is a serious issue that needs an efficient approach to solve it Derakhshan et al (2007). Some of the factors that contribute to the duplicate data generation include unreliability of the readers and duplicate readings generated by adjacent readers.

Duplicate readings unnecessarily consume system resources and impose traffic burdens on the system. Because RFID-enabled applications primarily use RFID data to automate business processes, inaccurate and duplicate readings could misguide application users Jeffery et al (2008).
Therefore, RFID data must be processed to filter out duplicates before an application can use it. Although, there are many approaches in the literature to filter duplicate readings (Wang et al 2008, Shen et al 2008) most of the existing approaches focus on data level filtering. They also tend to have high computation costs and they do not reduce much of the transmission overhead. Moreover, they tend to focus on a single RFID reader system whereas the research focuses on a system with multiple readers. Many applications used multiple readers for different purposes including increasing the reading ability, reading objects passing by different doors at the warehouse (Leong et al 2006) and supply chain management (Martinez-Sala et al 2009).

Rao et al (2006) introduces a deferred approach for detecting and correcting RFID anomalies. Each application uses declarative sequence-based rules in order to specify, detect, and correct relevant anomalies. It is noted that this approach is generally different from the methods proposed in (Floerkemeier et al 2005, Jeffrey et al 2006), which make the filtering process application independent. Clearly, both approaches have their own advantages in different scenarios. The generally accepted principle (Floerkemeier et al 2005) is that the separation of the middleware from the applications is a desirable goal because of the diversity of the applications.

Oleksandr Mylyy et al (2005) introduced a sliding filtering technique. A sliding window is a window with certain size that moves with time. Suppose the window size has time coordinate of \([t_1, t_1+\text{window\_size}]\), after some time, the coordinate will become \([t_1+s, t_1+\text{window\_size}+s]\). RFID reading tuples will enter the window and get expired as time moves. Therefore, the noise readings are read with count of distinct tag EPC values below a noise threshold. Denoising essentially performs the following operations: within any time window with size of window\_size surrounding an RFID reading, if
the count of the readings with same tag EPC values appears equal to or above threshold, then the observed EPC is not noise and needs to be forwarded for further processing: otherwise the reading is discarded. Two parameters used here are window size of a sliding time window, and a threshold for noise detection.

The Energy-Efficient In-Network RFID Data Filtering Scheme (EIFS) is proposed to filter duplicate readings in wireless sensor network (Bashir et al 2011). The objective of EIFS is to reduce the burden at the central processing and distribute the filtering task to the cluster head. In contrast, the aim of the study is to preserve the reading.

Leong et al (2006) proposed to arrange radio frequency absorbing material to stop the RFID propagation from dispersing to neighbouring reader’s area is discussed by Leong et al (2006). The absorbing material is placed between readers to prevent a reader from reading neighbouring tags. However, this solution is not feasible in most applications because of the design constraints and its high cost.

Carbunar et al (2005) proposed for the redundant reader elimination problem a randomized, decentralized, and localized approximation algorithm, called RRE. Although RRE assumes the presence of passive tags only, it is applicable to any number of RFID readers and tags and makes no assumptions on the underlying reader or tag topology. One significant assumption for this approach is that, RFID tags have limited memory, part of which is writable. RRE consists of two different steps. In the first step, each reader attempts to write the number of covered tags (its tag count) to all its covered tags. Therefore, first it must identify its tag count. We may notice that discussed above smoothing algorithm can also be used for this purpose. After each reader identified its tag count, it sends a write command containing its reader
identifier and tag count to all its covered tags. An RFID tag may store only
the highest value seen for tag count, along with the identity of the
corresponding reader.

In the second step, an RFID reader queries each of its covered tags.
Each tag replies with stored information, and, if the reader receives from a tag
its own ID that means it locked this tag and is responsible for the monitoring
of this tag. A reader that has locked no tag can be considered as redundant and
safely turned off. The problem with this approach is reader’s synchronization
problem. To solve this problem all active readers can maintain a list of locked
tags and listen passively for RFID tag replies to queries. And if an RFID
reader receives a message that its locked tag has another holder with a higher
tag count than its own, it removes this tag from its list of locked tags. Second,
the algorithm assumes that the position of readers does not change for a long
period of time. This assumption may not be held in applications such as
supply chain where the reader position may change in order to optimize the
business process (Chawathe et al 2004). A solution for this problem can be a
periodical reactivation of inactive readers and execution RRE on all readers.
Finally to this approach, even with centralized knowledge of the RFID system
topology, an optimal solution for the redundant reader elimination problem is
NP-hard.

Shin et al (2009) proposed to use the anti-collision algorithm such
as MRFID. MRFID is the enhancement of Time Division Multiple Access
(TDMA) anti-collision algorithm that takes into account the mobility of RFID
reader. However, this approach can be applied to applications that require all
the readers to be operated at the same time.

Approaches that explore the Bloom filter for filtering duplicate
readings in RFID has recently emerged in the literature (Wang et al 2008,
Shen et al 2008). The main idea of the standard Bloom Filter (BF) (Bloom et al 1970) is to represent an element in a form of positive counter in a bit array of size m using k number of hash functions. All bits in BF are initially set to 0 and will be replaced by 1 when it is hashed by the element. To test whether an element is a member of a set, the element will be run through the same hash functions used to insert the elements into the array. The element is said to be the member of the set if all the bits in which the element was mapped to is positive. For each new element, the corresponding k bits in the array are set to 1. An approach that used the original Bloom filter to remove the duplicate is discussed in (Wang et al 2008). Two approaches were proposed by Wang et al (2008) an eager and lazy approach that uses a Bloom filter to filter duplicate data. Generally, when a new reading comes at local reader, it will be inserted in the Bloom filter. The filter then will be sent to the central filter for update. The central filter coordinates readings from all the readers under its network. In the eager approach, the copy of the Bloom filter will be sent to every other reader to avoid the same reading from entering through them again. However it is too costly to update all the readers every time a new reading arrives. In the lazy approach, only a reader that sends a new reading will have new copy of the Bloom filter from the central filter. The research focused only on filtering at the central server and to preserve the reading only to the authorized reader. In contrast, the work presented in this research takes into account multiple RFID readers.

Methodologies for improving reliability of RFID data proposed in the literature can be divided into three main categories: physical solutions, middleware solutions and deferred solutions (Darcy et al 2009). Physical solutions include improvement of hardware performance to improve the reliability of the data such as those described in (Trotter et al 2010). Redundant techniques include using multiple tags and readers to identify the
same object (Rahmati et al 2007, Chen et al 2010), and additional techniques to remove duplicate readings generated from such deployments (Mahdin et al 2011). Middleware solutions include algorithms to correct the incoming sensor data streams before the data is passed into the database (Jefery et al 2006). The deferred solutions incorporate intelligent techniques which correct the data in the later stages within the data storage (Rao et al 2006).

The proposed work falls in the category of middleware-based solutions. It decided to use the window based method because of their simplicity and the proposed work extends the work proposed by Yijian Bai et al (2006). Many commercial RFID-middleware solutions Bornhoevd et al (2004) contain a fixed temporal-based sliding window data-smoothing filter as a solution to RFID unreliability, and applications are required to set the window size.

The goal is to reduce or eliminate dropped readings by giving each tag more opportunity to be read within the smoothing window. The study by Yijian Bai et al (2006) shows that setting the appropriate smoothing-window size is a non-trivial task, especially in the mobile environment.

Jeffery et al (2006) proposed an adaptive sliding-window cleaning method called Statistical sMoothing for Unreliable RFID data (SMURF). SMURF models the unreliability of RFID readings by viewing RFID streams as a statistical sample of tags in the physical world, and exploits techniques grounded in sampling theory to drive its cleaning processes. SMURF does not expose the smoothing-window parameter to the application; instead it automatically determines the most appropriate window size and continuously adapts it over the lifetime of the system based on observed readings.
Yijian Bai et al (2006) proposed base-line merge technique to eliminate data duplication. This algorithm focuses on time stamp of the tag-id. It produces duplicate RFID reading because it is not focusing on locations.

In order to reduce data duplication the research work proposed an efficient algorithm called EEDR (Efficient Elimination of Duplicate Readings).

The experimental analysis of the proposed approach and comparison with sliding windows-based approach Y Bai et al (2006) and the approach proposed in Jeffery et al (2006), The results show that our proposed approach demonstrates superior performance as compared to the other baseline approaches.

2.5 MODELING IN-STORE MOVEMENTS

The movements in the supermarket from different locations are considered as frequent walks. To analyze the customer behaviour in the supermarket frequent walks are extracted from the RFID path data.

Data mining techniques are applied to extract the frequent walks from the cleaned RFID data. Association rule mining is used to extract frequent walks. In this report the frequent item sets are considered as frequent walks among the different locations of a supermarket.

In the literature there are various techniques which are proposed for generating frequent item sets so that association rules are mined efficiently.

2.6 FREQUENT PATTERN MINING

Frequent pattern mining was first proposed by Agrawal et al (1993) for market basket analysis in the form of association rule mining. It
analyses customer buying habits by finding associations between different items that customers place in their “shopping baskets”. For instance, if customers purchase milk, how likely are they going to also buy cereal (and what kind of cereal) on the same trip to the supermarket? Such information can lead to increased sales by helping retailers do selective marketing and arrange their shelf space. Since the first proposal of this new data mining task and its associated efficient mining algorithms on various kinds of extensions and applications, ranging from scalable data mining methodologies, to handling a wide diversity of data types, various extended mining tasks, and a variety of new applications.

The concept of frequent itemset was first introduced for mining transaction databases (Agrawal et al. 1993). Let \( I = \{i_1, i_2, \ldots, i_n\} \) be a set of all items. A \( k \)-itemset \( a \), which consists of \( k \) items from \( I \), is frequent if \( a \) occurs in a transaction database \( D \) no lower than \( \theta |D| \) times, where \( \theta \) is a user-specified minimum support threshold (called min_sup), and \( |D| \) is the total number of transactions in \( D \).

### 2.7 DATA MINING TECHNIQUES

The approach of generating frequent itemsets in the data mining is divided into 3 basic categories.

**Horizontal layout based data mining techniques**

- Apriori algorithm
- DHP Algorithm
- Partition
- Sample
- Improved Apriori
Vertical layout based data mining techniques

- Eclat algorithm

Projected database mining techniques

- FP-tree algorithm
- H-Mine algorithm

There are a good number of algorithms used to mine frequent item sets. Some of them are very well known and have started a new era in the data mining. They made the concept of mining frequent item sets and association rules possible. A few algorithms are briefly discussed in this report. These algorithms vary mainly in how the candidate item sets are counted.

The steps involved in extracting frequent item sets are:

1. In the first pass, the support of each individual items is counted, and the large ones are determined.

2. In each subsequent pass, the large item sets determined in the previous pass is used to generate new item sets called candidate item sets.

3. The support of each candidate item set is counted, and the large ones are determined.

4. This process continues until no new large item sets are found.

2.7.1 Mining from Horizontal Layout Database

In this each row of database represents a transaction which has a transaction (TID), followed by a set of movement locations of customers also
called as items. The following Table 2.1 illustrates the horizontal layout dataset.

Table 2.1 Sample Horizontal Layout Database

<table>
<thead>
<tr>
<th>TID</th>
<th>Customer movement Locations to Purchase(Items)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>I1, I2, I3, I4, I5, I6</td>
</tr>
<tr>
<td>T2</td>
<td>I1, I2, I4, I7</td>
</tr>
<tr>
<td>T3</td>
<td>I1, I2, I4, I5, I6</td>
</tr>
<tr>
<td>T4</td>
<td>I1, I2, I3</td>
</tr>
<tr>
<td>T5</td>
<td>I3, I5</td>
</tr>
</tbody>
</table>

2.7.2 Apriori Algorithm

Apriori algorithm is proposed by Agrawal et al (1994) for mining frequent itemsets. The apriori property is “A nonempty subsets of a frequent item set must also be frequent”. This algorithm is the most classical and important for mining frequent itemsets. Apriori is used to find all frequent itemsets in a given database DB. The key idea of Apriori algorithm is to make multiple passes over the database. It employs an iterative approach known as a breadth-first search through the search space, where k - itemsets are used to explore (K+1) - itemsets.

The working of Apriori algorithm fairly depends upon the Apriori property. It also describes the anti monotonic property which says if the system cannot pass the minimum support test, all its supersets will fail to pass the test (Agrawal et al 1994). Therefore if the one set is infrequent then all its supersets are also frequent and vice versa. This property is used to prune the infrequent candidate elements. In the beginning, the set of frequent 1-
itemsets (1-length path) is found. The set of that contains one item i.e. 1-length path of the customer movement in the supermarket, which satisfy the support threshold, is denoted by L₁. In each subsequent pass, begin with a set of itemsets found to be large in the previous pass. This seed set is used for generating new potentially large itemsets, called candidate itemsets and count the actual support for these candidate itemsets during the pass over of the data. At the end of the pass, it determines which of the candidate itemsets are actually large (frequent), and they become the seed for the next pass. Therefore, L₁ is used to find L₂ the set of frequent 2-itemsets, which is used to find L₁, and so on, until no more frequent k-itemsets can be found. The feature first invented by R Agrawal et al (1994) in Apriori algorithm is used by the many algorithms for frequent pattern generation.

The basic steps to mine the frequent elements are as follows:

Generate the test: In this, first find the 1-itemset frequent elements L₁ by scanning the database and removing all those elements from C1, which cannot satisfy the minimum support criteria.

Join step: To attain the next level elements Ck join the previous frequent elements by self join i.e. Lₖ₋₁ * Lₖ₋₁ known as Cartesian product of Lₖ₋₁. I.e. the step generates new candidate k-itemsets based on joining Lₖ₋₁ with itself which is found in the previous iteration. Let Cₖ denote candidate k-itemset and Lₖ be the frequent i-itemset

Prune step: Cₖ is the superset of Lₖ so members of Cₖ may or may not be frequent but all K-1 frequent itemsets are included in Cₖ thus prunes the Cₖ to find K frequent itemsets with the help of Apriori property. I.e. this step eliminates some of the candidate k-itemsets using the Apriori property. A scan of the database to determine the count of each candidate in Cₖ will result in the determination of Lₖ (i.e., all candidates having a count no less
than the minimum support count is frequent by definition, and therefore belong to $L_k$)

However $C_k$ can be huge and so this will have complex computation. To shrink the size of $C_k$, the Apriori property is used as follows. Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any (k-1)-subset of candidate k-itemset is not in $L_{k-1}$ then the candidate cannot be frequent so it can be removed from $C_k$. Step 2 and 3 is repeated until no new candidate set is generated. To illustrate this, suppose n frequent 1-itemsets and minimum support is 1 then according to Apriori will generate $n^2 + (n^2)$ candidate 2-itemset (n 3) candidate 3-itemset and so on. For example, if there are 1000 frequent 1-sequences, apriori-based algorithms will generate $1000 \times 1000 + (1000 \times 999)/2 = 1,499,500$ candidate 2-sequences and $(1000 \times 999 \times 998)/6 = 166,167,000$ candidate 3-sequences are produced (Nizar et al 2010).

It is very clear that Apriori algorithm successfully finds the frequent elements from the database. But the significant drawbacks of Apriori algorithm are

1. More search space is required, this in turn will increase I/O cost.
2. Number of database scans are increased thus candidate generation will increase resulting in increase of computational cost.

Therefore many variations have been taken place in the Apriori algorithm to improve the efficiency and to minimize the above limitations arises due to increase in the size of database.
The algorithms proposed to improve the Apriori by:

- Reducing the passes of transaction database scans
- Shrinking number of candidates
- Facilitating support counting of candidates

2.7.3 Direct Hashing and Pruning (DHP)

It is absorbed that reducing candidate items from the database is one of the important tasks for increasing the efficiency. Thus a DHP technique was proposed by Park et al (1995). This technique is used to reduce the size of the candidate k-itemsets, \( C_k \) for \( k > 1 \). In this method, for example, when scanning each transaction in the database to generate the frequent 1-itemsets, \( L_1 \), from the candidate 1-itemsets in \( C_1 \), it generates all of the 2-itemsets for each transaction. The support count is mapped to different buckets of a hash table structure and increase the bucket count. If the bucket count in the hash table is below the support threshold it cannot be frequent and thus it is removed from the candidate set. This technique substantially reduces the number of candidate k-itemsets. However, this technique reduces the generation of candidate sets in the early stages but as the level increases the size of the bucket also increase thus it is difficult to manage hash table.

2.7.4 Partitioning Algorithm

Partitioning algorithm is proposed by Sevasere et al (1995) which is based to find the frequent elements on the basis portioning of database in “n” number of parts. It overcomes the memory problem for large database which does not fit into the main memory because small parts of database can easily fit into main memory. This algorithm is divided into two passes.
1. In the first pass whole database is divided into n number of parts.

2. Each partitioned database is loaded into main memory one by one and local frequent elements are found.

3. Combine the all locally frequent elements and make it globally candidate set

4. Find the globally frequent elements from this candidate set

It should be noted that if the minimum support for transactions in whole database is min_sup, then the minimum support count for a partition is min_sup X the number of transactions in that partition. For each partition, all frequent itemsets within the partition are found. These are referred to as local frequent itemsets.

A local frequent itemset may or may not be frequent with respect to the entire database thus any itemset which is potentially frequent must include in any one of the frequent partition.

As this algorithm is able to reduce the database scan for generating frequent itemsets but in some cases, the time needed to compute the frequency of candidate set generated in each partition is greater than the database scan thus resulting in increased computational cost.

### 2.7.5 Sampling algorithm

Toivonen et al (1996) is used to overcome the limitation of I/O overhead by not considering the whole database for checking the frequent itemsets. It is just based on the idea to pick a random sample of itemset R instead of the whole database D. The sample is picked in such a way that it is
accommodated in the main memory. Because the searching for frequent itemsets in the sample rather than in whole database D, it is possible that there is a chance to miss the global frequent elements in that sample. Therefore lower support threshold is used instead of actual minimum support to find the frequent itemsets local to sample. In the best case only one pass is needed to find all frequent elements if all the elements are included in the sample and if elements are missed in the sample then second pass are needed to find the itemsets missed in the first pass or in the sample.

Thus this approach is beneficial if efficiency is more important than the accuracy because this approach gives the result in less time and overcome the limitation of memory consumption.

2.7.6 Dynamic Itemset Counting (DIC)

Brins et al (1997) is used to reduce the number of database scans. It is based upon the downward disclosure property in which adds the candidate itemsets at different point of time during the scan. In this dynamic blocks are formed from the database marked by start points and unlike the previous techniques of Apriori it dynamically changes the sets of candidates during the database scan. Unlike the Apriori it cannot start the next level scan at the end of first level; it starts the scan by starting label attached to each dynamic partition of candidate sets. It reduces the database scan for finding the frequent itemsets by just adding the new candidate at any point of time during the run time. But it generates the large number of candidates and computing their frequencies are the bottleneck of performance while it requires fewer database scans.

The performance of all the above algorithms relies on an implicit assumption that the database is homogeneous and thus they will not generate too many extra candidates than Apriori algorithm does. For example, if all
the partitions in partition algorithm are not homogeneous and completely different sets of local frequent items-sets are generated from them.

2.7.7 Improved Apriori Algorithm

Dongme Sun et al (2007) proposed that the improved algorithm is based on the combination of forward scan and backward scan of given database. If certain conditions are satisfied, the improved algorithm can greatly reduce the iteration and scan time of the database required for finding of candidate itemsets. Suppose the itemset is frequent, all of its nonempty subsets are frequent, contradictory to the given condition that one nonempty subset is not frequent, the itemset is not frequent.

Based on this idea, first find frequent itemsets from the maximum itemset, then get all the nonempty subsets of the frequent itemset. Secondly, scan the database once again from the lowest level and count the frequent itemsets. The key advantage of improved Apriori algorithm is to extract frequent itemsets fast.

Improved Apriori algorithm extracts the maximum frequent itemsets directly and then extracts its subsets and compare them with the items in the database. Thus, it saves much time and the memory usage. The main drawback of this algorithm is it will lose mean because of extracting frequent itemsets fast.

2.7.8 Mining from Vertical Layout Database

In vertical layout format each column corresponds to an item, followed by a TID list, which is the list of rows that the item appears. An example of vertical layout database set is shown in the Table 2.2.
Table 2.2 Sample Vertical layout database

<table>
<thead>
<tr>
<th>Item</th>
<th>TID list</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>T1,T2,T3,T4</td>
</tr>
<tr>
<td>I2</td>
<td>T1,T2,T3,T4</td>
</tr>
<tr>
<td>I3</td>
<td>T1,T4,T5</td>
</tr>
<tr>
<td>I4</td>
<td>T1,T2,T3</td>
</tr>
<tr>
<td>I5</td>
<td>T1,T3,T5</td>
</tr>
<tr>
<td>I6</td>
<td>T1,T3</td>
</tr>
<tr>
<td>I7</td>
<td>T2</td>
</tr>
</tbody>
</table>

2.7.9 ECLAT Algorithm

Zaki (2000) proposed Equivalence CLASS Transformation (ECLAT) algorithm by exploring the vertical data format. The first scan of the database builds the TID_set of each single item. Starting with a single item (k = 1), the frequent (k+1) - itemsets grown from a previous k - itemset can be generated according to the Apriori property, with a depth-first computation order similar to FP-growth (Han et al 2000). The computation is done by intersection of the TID_sets of the frequent k- itemsets to compute the TID_sets of the corresponding (k+1) - itemsets. This process repeats, until no frequent itemsets or no candidate itemsets can be found. Besides taking advantage of the Apriori property in the generation of candidate (k + 1) - itemset from frequent k- itemsets, another merit of this method is that there is no need to scan the database to find the support of (k + 1) - itemsets (for k ≥1). This is because the TID_set of each k- itemset carries the complete information required for counting such support. Another related work which mines the frequent itemsets with the vertical data format is (Holsheimer et al
1995). This work demonstrated that, impressive results have been achieved for some data mining problems using highly specialized data structures.

2.7.10 Mining from Projected Database

The concept of projected database was proposed by few authors and applied to extract the frequent itemsets efficiently. The early approaches are able to mine the frequent itemsets but uses huge amount of memory. The concept of projected database uses divide and conquer principle to mine itemsets. Therefore support is computed efficiently than the Apriori based algorithms. The projected layout based approaches use tree structure to store and to mine the itemsets. This layout contains the record-id separated by column then record.

Tree projection algorithms based upon two types of ordering breadth-first and depth-first. In breadth-first order, nodes are constructed level by level in the lexicographic tree for frequent itemsets. In order to compute frequencies of nodes at k level, tree projection algorithm maintained matrices at nodes of the k-2 level and one database scan was required for counting support.

2.8 FP-GROWTH ALGORITHM

The FP-growth method (Han et al 2000) is a depth-first and divide-and-conquer algorithm. In this method, a structure called FP-tree is used to obtain a compact representation of the original transactions. Every branch of the FP-tree represents a transaction composed of the subset of frequent items. The nodes along the branches are stored in decreasing order of the frequency of all frequent items. Compression is achieved by overlapping itemsets which share prefixes of the corresponding branches to build the FP-tree. The FP-tree
has sufficient information to mine complete frequent patterns. Each node in the FP-tree has three fields: item-name, count, and node-link. The frequent itemsets can be found from the FP-tree quickly without having to scan the database on the disk frequently.

A frequent-item header table is built to make traverse the tree more easily. All frequent items are stored in the header table in decreasing order of their frequency. Each item points to its occurrence in the tree via a head of node-link. Nodes with the same item are linked via node-links. Each entry in the header table has two fields: item-name and head of node-link.

The FP-growth method scans the database only twice. In the first scan it finds all frequent items and inserts them into the header table in decreasing order of their counts. In the second scan, the root of FP-tree is created with “null.” The set of frequent items in each transaction is inserted into the FP-tree as a branch. If an itemset has the same prefix with another itemset already in the tree, this part of branch will be shared. A count in a node stores the number of the item which appears in this path. When a transaction is inserted into a new branch, the count is updated. After all nodes are linked from the header table, the FP-tree is completely constructed. The next step is to find all frequent itemsets. It collects all the patterns which a node participates by starting from its head in the header table and following its node-link. The mining process starts from the bottom of the header table. Paths with the same prefix item in the FP-tree construct the conditional pattern base of the prefix item with its support. Frequent items in the conditional pattern base construct the conditional FP-tree of the prefix item. It keeps constructing the conditional FP-tree until a single path is found. Frequent itemsets with the same prefix are generated by the single path.
The main work of FP-growth method is traversing FP-trees and constructing new conditional FP-trees from the global FP-tree. It needs to traverse the original FP-tree twice to construct a new conditional FP-tree. The first traversal finds all frequent items in the conditional pattern base and constructs a new header table for new conditional FP-tree. The second traversal constructs the new tree. FPclose (Grahne G et al 2005) will omit the first traversal by adopting an FP-array technique.

<table>
<thead>
<tr>
<th>a</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>2</td>
</tr>
</tbody>
</table>

**Figure 2.1 FP-array example**

<table>
<thead>
<tr>
<th>a</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>a</td>
<td>f</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>d</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td>g</td>
</tr>
<tr>
<td>b</td>
<td>d</td>
<td>g</td>
</tr>
<tr>
<td>b</td>
<td>d</td>
<td>g</td>
</tr>
</tbody>
</table>

**Figure 2.2 Database after the second scan**
Using the example of Figure 2.1 from (Grahne et al 2005), an array structure called FP-array is used in this method. During the second scan of the database of Figure 2.2, it constructs the FP-tree and the FP-array as shown in Figure 2.3. The mining method of FPclose (Grahne et al 2005) is different from CLOSET+ (Wang et al 2003). It will not do subset checking to avoid generating redundancies of a prefix, but generates a lot of candidates to check their closure. A structure called CFI-tree (Grahne et al 2005) as shown in Figure 2.4 is used to store frequent itemsets efficiently and to do closure checking.

The CLOSET+ algorithm (Wang et al 2003) is based on the FP-tree and uses the two-level hash indexed result tree to store frequent closed itemsets for closure checking. When it finds a single path in the conditional FP-tree, it must perform subset checking to check if it should generate the closed itemset candidate with a prefix itemset from all known frequent closed itemsets. If the prefix itemset is a subset of a known frequent itemset and has the same support, the itemsets of this prefix itemset will not be generated.

On the other hand, the closed itemset candidates are generated and then it must check the closure in the result tree to determine if they are really frequent closed itemsets. CLOSET+ needs to do many times of subset checking and closure checking. Since the result tree is searched many times, it takes a lot of time. The proposed algorithm ECIM is based on CLOSET+ algorithm but it will not check the closure check. This reduces the execution time and memory usage.
Figure 2.3 An FP-tree example
Figure 2.4 Construction of CFI-tree
The CLOSET+ algorithm (Wang et al 2003) is based on the FP-tree and uses the two-level hash indexed result tree to store frequent closed itemsets for closure checking. When it finds a single path in the conditional FP-tree, it must perform subset checking to check if it should generate the closed itemset candidate with a prefix itemset from all known frequent closed itemsets. If the prefix itemset is a subset of a known frequent itemset and has the same support, the itemsets of this prefix itemset will not be generated. On the other hand, the closed itemset candidates are generated and then it must check the closure in the result tree to determine if they are really frequent closed itemsets. CLOSET+ needs to do many times of subset checking and closure checking. Since the result tree is searched many times, it takes a lot of time. The proposed algorithm ECIM is based on CLOSET+ algorithm but it will not check the closure check. This reduces the execution time and memory usage.

2.9 EXTRACTING CLOSED FREQUENT ITEMSETS

A major challenge in mining frequent patterns from a large data set is the fact that it generates a huge number of patterns satisfying the min_sup threshold, especially when min_sup is set low. This is because if a pattern is frequent, each of its sub-patterns is frequent as well. A large pattern will contain an exponential number of smaller, frequent sub-patterns. To overcome this problem, closed frequent pattern mining and maximal frequent pattern mining were proposed. The scope of this thesis is limited to ‘closed frequent pattern’ mining.

A pattern \( \alpha \) is a closed frequent pattern in a data set \( D \) if \( \alpha \) is frequent in \( D \) and there exists no proper super-pattern \( \beta \) such that \( \beta \) has the same support as \( \alpha \) in \( D \). A pattern \( \alpha \) is a maximal frequent pattern (or max-pattern) in set \( D \) if \( \alpha \) is frequent, and there exists no super-pattern \( \beta \) such that \( \alpha \subset \beta \) and \( \beta \) is frequent in \( D \). For the same min_sup threshold, the set of
closed frequent patterns contains the complete information regarding to its corresponding frequent patterns; whereas the set of max-patterns, though more compact, usually does not contain the complete support information regarding to its corresponding frequent patterns. The mining of frequent closed itemsets was proposed by Pasquier et al (1999), where an Apriori-based algorithm called A-Close for such mining was presented. Other closed pattern mining algorithms include CLOSESET (Pei et al 2000), CHARM (Zaki & Hsiao 2002), CLOSESET+ (Wang et al 2003), FPClose (Grahne & Zhu 2003) and AFOPT (Liu et al 2003). The main challenge in closed (maximal) frequent pattern mining is to check whether a pattern is closed (maximal). There are two strategies to approach this issue: (1) to keep track of the TID list of a pattern and index the pattern by hashing its TID values. This method is used by CHARM which maintains a compact TID list called a diffset; and (2) to maintain the discovered patterns in a pattern-tree similar to FP-tree.

This method is exploited by CLOSESET+, AFOPT and FPClose. A Frequent itemset Mining Implementation (FIMI) workshop dedicated to the implementation methods of frequent itemset mining was reported by Goethals & Zaki (2003). Mining closed itemsets provides an interesting and important alternative to mining frequent itemsets since it inherits the same analytical power but generates a much smaller set of results. Better scalability and interpretability is achieved with closed itemset mining.

2.10 TRACKING MOVEMENT PATH OF CUSTOMERS

In RFID enabled warehousing, tagged products can be traced with the help of installed RFID readers. An active tag fixed to the shopping cart or attached to the products enables tracing the movement of the customers. Based on this, Berenyi & Charaf (2008) had assumed such an environment, and analyzed how appropriate information can be recovered utilizing data mining techniques from the accumulated RFID-related data.
Taknobu Nakahara et al (2010) used RFID tags equipped shopping carts to track customers’ in-store movements to collect data on their paths. These Path data obtained from customers movements recorded in a spatial configuration contain valuable information for marketing.

The potential sequential patterns of customers and their in-store movements are extracted by employing LCMseq algorithm to shopping path data. The frequent sequence patterns have been efficiently enumerated by the LCMseq algorithm. Finally, in-store movements of most important customers have been determined by constructing a decision tree model utilizing the extracted patterns.

The behaviour of the sequences may vary with time as the data is often generated from a dynamic environment. Chen & Hu (2006) had integrated time (recent sequences) and compactness into conventional sequential mining for adjusting the discovered patterns to these changes.

The existing studies on finding sequential patterns can be fundamentally classified into two major categories. In the first category, the continuous patterns are discovered in which all the elements in the pattern appear in the next following spots in transactions. In the second category, the discontinuous patterns are mined in which the nearby elements in the pattern need not be present successively in transactions. According to this, Berenyi et al (2008) have proposed an algorithm for finding the walking path sequence of the customer behaviour from RFID equipped data. They observed how significant information was retrieved with the help of data mining methods from the gathered RFID related tracking data. The frequent sequences have been taken out, by exposing the most visited spots and the walks across the RFID warehouse and the typical products selected along the way.
The existing literature supports the view of considering a shopping trip as a process of goal setting and task accomplishment. One theory suggests that people like to be in situations in which they are constantly making progress towards their goals (Deci et al 1985). Takahashi (1988) points out that there is clear evidence that the choice of a destination depends substantially on the utility of possible later destinations on the same trip chain. Although empirically such behaviour was only studied in out-of store multi-destination shopping trip scenarios (Brooks et al 2008), within a store, people are expected to exhibit similar trip chaining behaviour. Shopping trips should be carefully examined from a spatial standpoint to shed lights on customer search and purchase behaviour.

Limited work has been dedicated to studying in-store shopping behaviour utilizing the traversal paths each shopper makes. This is mainly due to the difficulty of accessing consumer in-store shopping trip data. Recently, researchers start to leverage the datasets made available with RFID devices for shopping behaviour study (Hui et al 2008). The proposed research is generating frequent walking patterns among different locations. These frequent patterns are used to generate association rules. Generating association rules is out of the scope of the present research.

Association rule discovery is well-suited for applications of market basket analysis to reveal regularities in the purchase behaviour of customers. Association rules in a retailing context are rules showing what items are more frequently bought together than others. These rules can be obtained by analyzing sales transaction data (Agrawal et al 1993). The authors have recently proposed to utilize association rule mining approaches to resolve product assortment and allocation problems in retailing (Brijs et al 1999) (Chen et al 2007).
2.11 CONCLUSION

This chapter gave a short walkthrough of the research works carried out in the elimination of data redundancy in RFID applications. The works related to frequent pattern mining are discussed.