BAB II
TINJAUAN PUSTAKA

2.1 E-Commerce

Electronic Commerce (e-commerce) adalah proses pembelian, penjualan atau pertukaran produk, jasa dan informasi melalui jaringan komputer. *E-commerce* merupakan bagian dari *e-business*, dimana cakupan *e-business* lebih luas, tidak hanya sekedar perniagaan tetapi mencakup juga pengkolaborasian mitra bisnis, pelayanan nasabah, lowongan pekerjaan dll. Selain teknologi jaringan www, *e-commerce* juga memerlukan teknologi basis data atau pangkalan data (database), e-surat atau surat elektronik (e-mail), dan bentuk teknologi non komputer yang lain seperti halnya sistem pengiriman barang, dan alat pembayaran untuk *e-commerce* ini (Siregar, 2010).

E-commerce secara strategis dapat berperan sebagai diferensiator yang dapat membentuk daya saing perusahaan melalui sejumlah keunikan baik produk maupun sistem pelayanannya (Turban dkk.,2010). Usaha bisnis perniagaan tidak lagi bergantung kepada ruang dan waktu dengan segmen pasar yang terbatas. *E-commerce* merupakan satu set dinamis teknologi, aplikasi, dan proses bisnis yang menghubungkan perusahaan konsumen, dan komunitas tertentu melalui transaksi elektronik dan perdagangan barang, pelayanan, dan informasi yang terjadi secara elektronik (Fingar dkk., 2000). Sistem *e-commerce* sebagai suatu bentuk kemajuan teknologi informasi telah membawa sejumlah perubahan, diantaranya menurunkan biaya transaksi antara pembeli dan penjual, interaksi menjadi lebih mudah tanpa batasan waktu dan tempat, lebih banyak alternative dan transparansi bisnis dan kemudahan memberikan pelayanan kepada konsumen atau pelanggan (Bernadi, 2013).

Dari sekian banyak kelebihan, *e-commerce* juga memiliki kekurangan. Salah satu kesalahan yang sering terjadi adalah kurangnya informasi produk dan pelayanan mengenai produk tersebut (Kienan, B., 2000). Kurangnya informasi produk, akan mempengaruhi keputusan pengguna saat akan membeli. Cara yang

dapat dilakukan pelaku bisnis *e-commerce* adalah dengan mengelola data transaksi penjualan dan respon pengguna untuk memberikan pelayanan yang lebih baik kedepannya.

2.2 Pengguna Relationship Management

Customer Relationship Management (CRM) adalah strategi bisnis yang terdiri dari *software* dan layanan yang didesain untuk meningkatkan keuntungan (*profit*), pendapatan (*revenue*) dan kepuasan pelanggan (*customer satisfaction*). Caranya adalah dengan membantu berbagai bentuk perusahaan untuk mengidentifikasi pelanggannya dengan tepat memperoleh lebih banyak pelanggan dengan lebih cepat, dan mempertahankan kesetiaan pelanggannya (Andreani, 2007).

Customer Relationship Management (CRM), dalam Bahasa Indonesia menjadi menajemen relasi pelanggan. Konsep CRM merupakan spesifikasi dari konsep *Relationship Marketing* (RM). Konsep *relationship marketing* (pemasaran hubungan) menerangkan bahwa perusahaan harus berinteraksi dan berhubungan dengan berbagai pihak yang berkepentingan terhadap perusahaan (*stakeholders*), karena hubungan yang baik merupakan *asset* yang paling mendasar bagi suatu perusahaan. Adapun konsep CRM lebih menekankan pada menjalin hubungan baik dengan pelanggan sebagai salah satu *stakeholders*, karena pelanggan dianggap sebagai ujung tombak suatu bisnis (Imbar & Gunawan, 2013).

Kerangka komponen CRM diklasifikasikan menjadi 3 yaitu (Kurniawan, 2009) :

1. Operasional CRM: Operasional CRM dikenal sebagai *front office* perusahaan. Komponen CRM ini berperan dalam interaksi dengan pelanggan. Operasional CRM mencakup proses otomatisasi yang terintegrasi dari keseluruhan proses bisnis, seperti otomatisasi pemasaran, dan pelayanan. Salah satu penerapan CRM yang termasuk dalam kategori operasional CRM adalah dalam bentuk aplikasi web. Melalui web, suatu perusahaan dapat memberikan pelayanan kepada pelanggan.

2. Analitikal CRM: Analitikal CRM dikenal sebagai *back office* perusahaan. Komponen CRM ini berperan dalam memahami kebutuhan pelanggan. Analitikal CRM berperan dalam melaksanakan analisis pelanggan dan pasar, seperti analisis trend pasar dan analisis kebutuhan dan perilaku pelanggan. Data yang digunakan pada CRM analitik adalah data yang berasal dari CRM operasional.
3. *Collaborative* CRM: Komponen kolaborasi CRM meliputi e-mail *personalized publishing*, *ecommunities*, dan sejenisnya yang dirancang untuk interaksi antara pelanggan dengan perusahaan. Tujuan utamanya adalah memberikan nilai tambah dan memperluas loyalitas pelanggan ke pelanggan lain yang masih belum berada di level kesetiaan pelanggan. *Collaborative* CRM juga mencakup pemahaman atau kesadaran bahwa pelanggan yang setia dapat menjadi magnet bagi pelanggan lain.

2.2.1 Metode *Up-Selling*

Up-Selling adalah teknik penjualan yang mendorong konsumen untuk membeli barang-barang yang lebih mahal, *upgrade*, atau *add-ons* dalam upaya membuat penjualan lebih menguntungkan (Diki, et al., 2015).

Stephan Schiffman mendefinisikan *Up-Selling* adalah "apa yang terjadi ketika anda mengambil inisiatif untuk meminta seseorang yang sudah membeli sesuatu yang anda tawarkan untuk membeli lebih dari itu atau lebih dari sesuatu yang lain". *Up-Selling* berarti bergerak"*up"* ke versi yang lebih mahal dari apa yang konsumen sudah pertimbangkan untuk dibeli. Metode ini digunakan setelah memilih *item* yang dipilih oleh konsumen tetapi sebelum pembelian actual. *Customer relation* perlu memahami beberapa faktor dalam proses *Cross-Selling* dan *Up-Selling* ini, antara lain (Afina & Ainur, 2015):

1. Kepuasan pelangggan terhadap pelayanan sebelumnya
2. Sejauh mana produk dan pelayanan tambahan ini memenuhi kebutuhan pelanggan
3. Harga yang ditawarkan.

Cara Kerja metode *Up-Selling* yang akan diaplikasi pada *website* yaitu: (1) Mengambil data barang yang harga jualnya lebih tinggi dari barang yang sedang dilihat, (2) Urutkan dari harga termahal, (3) Tampilkan barang dari harga termahal sampai termurah (Imbar & Gunawan, 2013).

2.2.2 Metode *Cross-Selling*

Cross-Selling biasanya digunakan oleh sebagian besar *online store* untuk menentukan rekomendasi produk apa yang seharusnya dijual juga. Sebagai contoh dapat dilihat salah satu *online bookstore* terkemuka seperti *Amazon.com* dapat dengan jelas terlihat bahwa jika seorang konsumen membeli buku secara *online* maka *website* akan memberikan pula rekomendasi mengenai *related books* yang direkomendasikan untuk dibeli. Hal ini dapat dilakukan melalui analisis *Cross-Selling* berdasarkan pola pembelian konsumen yang bertransaksi secara *online* melalui *website* (Tang & MacLennan, 2005).

Penerapan *Cross-Selling* harus didahului oleh analisis yang mendalam mengenai data transaksi pelanggan dengan menggunakan konsep *data mining* yang melibatkan proses pengambilan sumber informasi dari sebuah transaksi pelanggan, yang mencakup produk apa yang mereka beli, perilaku pembelian pelanggan, dan lain-lain (Tama, 2012).

Ada tiga manfaat utama dari teknik *Cross-Selling* yaitu: (1) meningkatnya *revenue* perusahaan, (2) meningkatkan loyalitas konsumen (*customer loyalty*), dan (3) meningkatkan *customer awareness* ke satu perusahaan (Chasin, 2003).

Cara kerja Metode *Cross-Selling* yang akan di aplikasikan pada *Website* yaitu : (1)Mengambil data barang apa saja yang telah terjual bersama barang yang sedang dilihat, (2)Akumulasikan banyak barang yang paling banyak dibeli bersama barang yang sedang dilihat, (3)Urutkan Berdasarkan Barang yang paling banyak dibeli, (4)Ambil 3 barang teratas untuk di rekomendasikan (Imbar & Gunawan, 2013).

2.3 Sistem Rekomendasi

Sistem Rekomendasi adalah perangkat lunak dan teknik memberikan saran untuk barang yang berguna bagi pengguna. Saran yang diberikan bertujuan untuk mendukung pengguna dalam berbagai proses pengambilan keputusan, seperti barang apa yang akan dibeli, musik apa yang diingin didengarkan atau berita apa yang ingin dibaca (Ricci, et al., 2011). Saat ini, rekomendasi *online* menjangkau layanan seperti rekomendasi buku, musik, film, halaman web dan rekomendasi restoran, menunjukkan berbagai domain penerapan sistem *recommender* yang ada (Weiyang, 2000).

Beberapa manfaat dari sistem rekomendasi antara lain (Ricci, et al., 2011) :

1. Meningkatkan penjualan barang.

   Penjual dapat menambahkan beberapa barang dengan barang yang sering terjual meski tanpa direkomendasikan. *Item* yang ditambahkan disesuaikan dengan yang dibutuhkan atau diinginkan oleh pembeli.

2. Menjual lebih beragam item.

   Tujuan lain dari sistem rekomendasi adalah membuat pengguna memilih item yang sulit ditemukan jika tidak direkomenFdasikan. Singkatnya, sistem akan merekomendasikan juga *items* yang jarang dibeli.

3. Meningkatkan Kepuasan pengguna.

   Sebuah sistem rekomendasi yang baik juga dapat meningkatkan pengalaman pengguna melalui situs atau aplikasi. Pengguna akan menemukan rekomendasi yang menarik, relewan, dan dengan desain yang baik pengguna akan menikmati pemakaian sistem.

4. Meningkatkan loyalitas pengguna.

   Pengguna akan semakin setia pada sebuah situs *web* yang ketika dikunjungi, akan mengenali pelanggan lama dan diperlakukan sebagai pengunjung yang berharga. Sistem akan merepresentasikan informasi *item* yang sesuai dengan pengguna berdasarkan informasi dari interaksi-interaksi sebelumnya.

5. Memahami keinginan pengguna.

   Berdasarkan informasi dari aktivitas pengguna dapat dijadikan sebagai pengetahuan dalam mengatur persediaan barang ataupun memproduksinya.

Pengguna menginginkan sebuah sistem rekomendasi apabila sistem tersebut secara efektif mendukung kegiatan dan memenuhi tujuan pengguna. Konsekuensinya, sistem rekomendasi haru memberikan pelayanan yang sesuai dengan pengguna.

2.3.1 Association Rules

Aturan *asosiasi* adalah teknik *data mining* untuk menemukan aturan asosiatif antara kombinasi *item*. Penting atau tidaknya suatu aturan asosiasi dapat diketahui dengan dua parameter, *support* yaitu persentase kombinasi item tersebut dalam *database* dan *confidence* yaitu kuatnya hubungan antara *item* dalam *association rule* (Azhari & Anshori, 2009).

Association Rule Mining berfokus menemukan aturan yang akan memprediksi terjadinya sebuah item yang muncul berdasarkan kejadian item lainnya dalam suatu transaksi (Ricci, et al., 2011).

Tujuan dari *Association rule mining* adalah menemukan semua *rule* yang memiliki *support* ≥ *minsup* (minimal *support*) dan *confidence* ≥ *mincof* (minimal *confidence*). Penggalian aturan asosiasi tersebut dilakukan dengan cara (Ricci, et al., 2011):

1. *Frequent itemset generation*

   Menemukan semua *itemset* dengan syarat *support* ≥ *minsup*. *Itemset* ini dinamakan sebagai *itemset* yang frekuen atau sering muncul.

2. *Rule Generation*

   Menemukan *confidence* paling tinggi dari *frequent itemset* sehingga ditemukanlah aturan yang kuat (*strong rule*).

Ada 2 tahap metodologi dasar analisis asosiasi :

a. Analisis Pola frekuensi tinggi.

   $$\text{Support (A)} = \frac{\text{Jumlah Transaksi mengandung A}}{\text{Total Transaksi}}$$

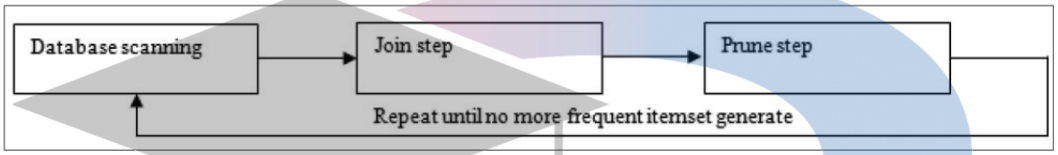
b. Pembentukan Asosiasi

   Dimana Nilai *Confidence* suatu aturan A→ B diperoleh dari:

   $$\text{Confidence } P(B|A) = \frac{\text{Jumlah Transaksi mengandung A dan B}}{\text{Jumlah Transaksi Mengandung A}}$$

2.3.2 Algoritma *Apriori*

Algoritma *Apriori* adalah suatu algoritma yang sudah sangat dikenal dalam melakukan pencarian *frequent itemset* dengan menggunakan teknik aturan asosiasi. Algoritma ini pertama kali diajukan oleh R. Agrawal dan R. Srikant pada tahun 1994. Algoritma *apriori* menggunakan *knowledge* untuk menentukan *frequent itemset*, setelah itu memproses hasil tersebut menjadi informasi yang digunakan untuk menghasilkan kandidat-kandidat yang mungkin berhubungan.



Gambar 2. 1 Blok diagram algoritma Apriori
(sumber : Tandel & Soni, 2017)

Pada proses mendapatkan *item sets* seperti pada blok diagram diatas, Algorima *Apriori* selalu mengulangi proses *scan database* kemudian melanjutkan ke proses pengabungan dan proses penghapusan *candidate*. Hal ini akan terus berulang sampai tidak ada lagi *frequent item set* yang dibangkitkan (Tandel & Soni, 2017).

Notasi-notasi yang digunakan dalam algoritma apriori (Agrawal, R., dan Ramakrishnan, S., 1994) antara lain :

1. Minsup (minimum support) : ambang batas support
2. Mincof (minimum confidence) : ambang batas confidence
3. Itemset : kelompok produk
4. Support count (s.count) : frekuensi kejadian itemset dari seluruh transaksi
5. Kandidat itemset (Ck) : kandidat k-itemset, dimana k menunjukkan jumlah pasangan item
6. Large itemset (Lk): itemset yang sering terjadi
7. C : confidence
8. T : transaksi
9. S : support
10. Ct : Kandidat yang ada pada t

Ada dua proses dalam membangun kandidat *itemset* yang dilakukan dalam algoritma *Apriori* (Agrawal, R., dan Ramakrishnan, S., 1994), yaitu:

1. *Join* (Penggabungan)

Untuk menemukan *large itemset* (L_k), kandidat *itemset* (C_k) dibangkitkan dengan melakukan proses *join* L_{k-1} dengan dirinya sendiri, $C_k = L_{k-1} * L_{k-1}$, lalu anggota C_k diambil hanya yang terdapat dalam L_{k-1}. Pseudocode dapat dilihat sebagai berikut *(*Sumber : *Agrawal, R., dan Ramakrishnan, S., 1994)*

```
Insert into Ck
Select p.item1, p.item2,…, p.itemk-1, q.itemk-1
from Lk-1 p,Lk-1q
where p.item1=q.item1,…, p.itemk-2=q.itemk-2, p.itemk-1<1.itemk-1;
```

2. Prune (Pemangkasan)

Menghapus semua *itemset* c dalam C_k dimana (k-1) *subset* dari c tidak ada pada Lk-1. Pseudocode dapat dilihat sebagai berikut *(*Sumber : *Agrawal, R., dan Ramakrishnan, S., 1994)*

```
Forall itemsets c ∈ Ck do
    forall (k-1)subsets s of c do
      if (s ∈ Lk-1) then
        delete c from Ck
```

Berikut merupakan algoritma dari *Apriori (Agrawal, R., dan Ramakrishnan, S., 1994):*

Masukan : D, *database* transaksi; *minsup,* nilai batas dari minimum *support.*

Keluaran : L, *frequent itemset* di dalam D.

```
1)    L1 = find_frquent_1-itmsets(D);
2)    For (k=2; Lk-1 ≠ Φ ;k++) {
3)        Ck = apriori _gen (Lk-1);
4)        Foreach transaction t ∈ D {
5)            Ct = subset (Ck,t);
6)            Foreach candidate c ∈ Ct
7)              c.count ++;
8)        }
9)        Lk = {c ∈ Ck | c.count ≥ min_sup}
10)   }
11)   Return L = ∪k Lk.
```

Langkah pada algoritma *Apriori* adalah (Agrawal, R., dan Ramakrishnan, S., 1994):

1. Mencari frequent itemset dari database transaksi.
2. Menghapus *itemset* dengan frekuensi rendah berdasarkan *minsup* yang telah ditentukan sebelumnya
3. Menentukan aturan asosiasi (*association rule*) dari *itemset* yang memenuhi ketentuan nilai *minconf* dalam *database.*

Algoritma *apriori* bekerja dengan menghasilkan kandidat baru dari k-itemset untuk menemukan *frequent itemset* dan menghitung nilai *support k-itemset* tersebut. *Itemset* yang memiliki nilai *support* dibawah *minsup* akan dihapus. Langkah selanjutnya adalah menghitung rule berdasarkan *minconf*. Bila *rule* yang didapatkan memenuhi ataupun melebihi batasan *minconf* yang sudah ditentukan, maka rule tersebut merupakan *strong rule*. Proses perhitungan dalam algoritma berhenti ketika tidak lagi *frequent itemset* baru yang dihasilkan (Agrawal, R., dan Ramakrishnan, S., 1994).

2.3.3 Algoritma *DC_Apriori*

DC_Apriori merupakan algoritma peningkatan dari *Apriori* yang menutupi masalah pada algoritma *Apriori seperti:* (1). Pengulangan scanning database, (2). Menghasilkan *candidate sets* yang besar, (3). Menghabiskan banyak waktu dalam perhitungan (Du, et al., 2016).

Inferences terkait Algoritma *DC_Apriori.*

- Inference 1: Jika *frequent k -item sets* L_k tetap dapat menghasilkan *frequent* (k+1)-*item sets* $L_{k+1,}$ Penjumlahan dari *items* di dalam L_k harus lebih besar dari pada k.
- Inference 2: Ketika $k \geq 2$, dengan memperhitungkan $\forall_{Ck} \in c_k$, dimana $C_k \in (L_{k-1} \infty L_{k-1),}$ maka $(C_{k-1} \infty L_1)$.

Cara kerja dari Algoritma *DC_Apriori* antara lain (Du, et al., 2016) :

1. Langkah pertama, *scan* database transaksi dan atur kembali struktur penyimpanan pada *database*, kemudian bangun kembali struktur dari jumlah transaksi Tid dan dari *item* ke *Item-Tid*, kemudian disimpan. Setelah dibangun kembali, database baru lebih mudah untuk menghitung nilai *support* dari *candidate item sets*, jumlah transaksi dari *list Tid* berdasarkan kemunculan

item. Contoh tabel transaksi dengan *minimum support* 2/10.

Tabel 2. 1 Transaction Table (Du, et al., 2016)

Tid	***Item***
1	11,12,15
2	12,14
3	11,12,13
4	11,13
5	12.13
6	13,15
7	11,15
8	12,13,15
9	14,15
20	12

Scan table transaksi pada table 2.1, Simpan data untuk penggabungan ulang *item -Tid* pada *database*, seperti yang terlihat pada tabel 2.2, Dalam proses *Scan,* jumlah transaksi per item dicatat, kemudian akan didapatkan *Candidates* C_1, bandingkan seberapa panjang list per transaksi *items* dan hilangkan semua *item* yang lebih kecil dari *Minimum Support.* Semua *canditates* C_1 harus lebih besar dari pada *minimum support.*

Tabel 2. 2 Recombined Item Sets Table (Du, et al., 2016)

Item	**Tid**					
11	1	3	4	7		
12	1	2	3	5	8	10
13	3	4	5	6	8	
14	2	9				
15	1	6	7	8	9	

2. Langkah kedua, Mengunakan Inference 1, untuk menentukan apakah saat ini *frequent Item set* terpenuhi dengan kondisi untuk menghasilkan *candidate* berikutnya. Jika benar lanjut ke langkah berikutnya, atau jika tidak *frequent*

item set adalah jawaban akhir.

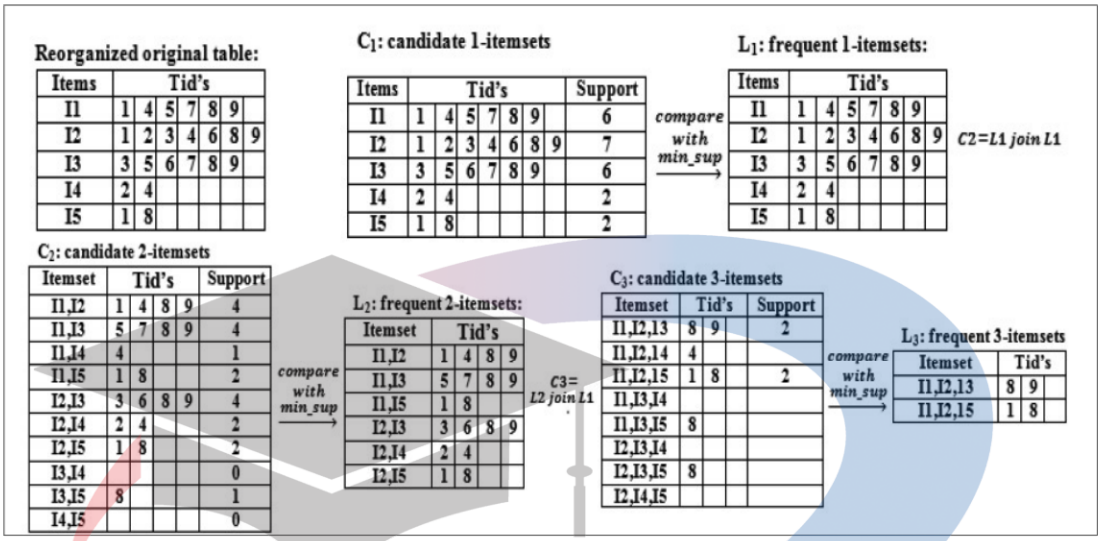
3. Langkah Ketiga, Sesuai dengan Inference 2, ketika $k \geq 2$, k-*canditates* C_k dapat di hasilkan dengan penggabungan *frequent* (k-1) *item sets* L_{k-1} dan frequent *item sets* L_1. Sehingga selama proses $L_{k-1} \infty L_{k-1}$, hanya perlu membandingkan ukuran element terakhir dari t ($t \in l_{k-1}$) dalam l_{k-1}($l_{k-1} \in L_{K-1}$) dan l_1 ($l_1 \in L_1$) dalam index L_1. Jika $t<l_1$, Ambil *intersection* dari keduanya antara l_{k-1} dan l_1 transaksi list, c_k ($c_k \in C_k$) berasal dari list transaksi. Kemudian memutuskan apakah dukungan dari C_k lebih kecil dari *minimal support,* dan menghapus c_k jika tidak memenuhi persyaratan dari C_k, kemudian mendapatkan *Frequent k-item sets* L_k, seperti pada table 2.3.

Tabel 2. 3 Frequent 2-Item Sets Table (Du, et al., 2016)

Item	Tid		
11,12	1	3	
11,13	3	4	
11,15	1	7	
12,13	3	5	8
12,15	1	8	
13,15	6	8	

4. Langkah ke 4, Mengulangi langkah kedua dan ketiga, sampai tidak menghasilkan *Frequent item set* lagi.

Sebagai contoh dari proses penggunaan algoritma *DC_Apriori* dalam mendapatkan *itemsets* seperti pada ilustrasi dibawah ini.



Gambar 2.2 Ilustrasi Prosess Algoritma DC_Apriori dalam mendapatkan itemsets (Sumber : Tandel & Soni, 2017)

Seperti yang terlihat pada ilustrasi proses algoritma *DC_Apriori* dalam mendapatkan *itemset* diatas, Algoritma *DC_Apriori* selalu membandingkan setiap nilai *support* yang didapat dari perhitungan dengan *minimal support* yang ditentukan, setelah itu melakukan pengabungan untuk mendapatkan *candidate*. Proses ini akan berhenti ketika pengabungan yang mengasilkan *candidate* tidak lebih kecil dari *minimal support* (Tandel & Soni, 2017).

Pada pseudocode di bawah ini, Algoritma *DC_Apriori* selalu membandingkan nilai *support* dengan minimal *support* kemudian digabungkan dengan setiap *candidate* yang memenuhi minimal *support*. Jika tidak memenuhi akan dihapuskan dari *candidate item sets.* Dan jika memenuhi syarat dari minimal *support* nantinya akan digunakan sebagai *rule* (Tandel & Soni, 2017).

```
Input:Transaction database D.
Minimum support min_sup
Output:Frequent item sets L.
1)for each Transaction t in D{
2)While(getline(strStreamItem,strIt
em,'')){
//get transaction t's item
3)if(t.item==c1.item)
4)then c1.item.Tid.insert(t);
//add item's Tid into transaction t
5)else
6)then c1.insert(item&Tid);
7)}
8)}
9)for each c1 in c1{
//judge the support of each 1-
candidate c1
10)L1={c1€C1|c1.sup>=min_sup};
//Generate frequent 1-item sets L1
11)}
12)for(k=2;Lk-1!=NULL;k++){
//if Lk-1 is empty,stop the
circulation
13)Lk=apriori_gen(Lk-1~L1);
//Generate frequent k-item sets Lk
14)}
15)Return L= Lk-1;
//The last frequent item sets
```

```
While the process of generate
frequent k-itemsets Lk;
1)for all Ik-1€Lk-1
// Ik-1 containing (k-1)elements is
subset of Ik-1
2)for all I1€L1
3)if(lk-1.Tid=l1.Tid)&(lk-
1l1.sup>=minsup
4)then lk=lk-1l1;
//Take the intersection of both lk-1
and l1's transaction st,then add the
item which is generated by Lk-1~L1,
and which support is bigger than
the min_sup into the frequent k-
item sets Lk.
5)else
6)then L1->next
//otherwise l1 movess to the next
frequent l-item set
7)end if
8)end for
9)end for
10)return lk
```

Pseudocode Algoritma DC_Apriori

(Sumber : Tandel & Soni, 2017)