BAB II

TINJUAN PUSTAKA

2.1 Tinjauan Tempat Penelitian

2.1.1 Sejarah Singkat dan Perkembangan Umum Perusahaan

1882 Sebuah badan usaha swasta penyedia layanan pos dan telegrap dibentuk pada masa pemerintahan kolonial Belanda.

1906 Pemerintah Kolonial Belanda membentuk sebuah jawatan yang mengatur layanan pos dan telekomunikasi yang diberi nama Jawatan Pos, Telegrap dan Telepon (Post, Telegraph en Telephone Dienst/PTT).

1945 Proklamasi kemerdekaan Indonesia sebagai negara merdeka dan berdaulat, lepas dari pemerintahan Jepang.

1961 Status jawatan diubah menjadi Perusahaan Negara Pos dan Telekomunikasi (PN Postel).

1965 PN Postel dipecah menjadi Perusahaan Negara Pos dan Giro (PN Pos & Giro), dan Perusahaan Negara Telekomunikasi (PN Telekomunikasi).

1974 PN Telekomunikasi selanjutnya dipecah menjadi dua perusahaan milik negara, yaitu Perusahaan Umum

Telekomunikasi (“Perumtel”) untuk menyediakan layanan telekomunikasi domestik dan internasional serta PT Industri Telekomunikasi Indonesia Tbk (“PT INTI”) untuk menyediakan manufacturing peralatan telekomunikasi.

1980 Seluruh saham PT Indonesian Satellite Corporation Tbk (Indosat) diambil alih oleh pemerintah RI menjadi Badan Usaha Milik Negara (BUMN) untuk menyelenggarakan jasa telekomunikasi internasional, terpisah dari Perumtel.

1989 Undang-undang nomor 3/1989 tentang Telekomunikasi, tentang peran serta swasta dalam penyelenggaraan telekomunikasi.

1991 Perumtel diubah menjadi “Persero”, atau perseroan terbatas milik negara dengan tujuan komersial dan diganti namanya menjadi Perusahaan Perseroan (Persero) P.T. Telekomunikasi Indonesia, yang dikenal sebagai TELKOM.

1995 Penawaran Umum perdana saham Telkom (Initial Public Offering/IPO) dilakukan pada tanggal 14 November 1995. Sejak itu saham Telkom tercatat dan diperdagangkan di Bursa Efek Jakarta (BEJ), Bursa Efek Surabaya (BES), New York Stock Exchange (NYSE) dan London Stock Exchange (LSE). Saham Telkom juga diperdagangkan

tanpa pencatatan (*Public Offering Without Listing*/POWL) di Tokyo Stock Exchange.

1996 Kerja sama Operasi (KSO) mulai diimplementasikan pada 1 Januari 1996 di wilayah Divisi Regional I Sumatra – dengan mitra PT Pramindo Ikat Nusantara (Pramindo); Divisi Regional III Jawa Barat dan Banten – dengan mitra PT Aria West International (AriaWest); Divisi Regional IV Jawa Tengah dan DI Yogyakarta – dengan mitra PT Mitra Global Telekomunikasi Indonesia (MGTI); Divisi Regional VI Kalimantan – dengan mitra PT Dayamitra Telekomunikasi (Dayamitra); dan Divisi Regional VII Kawasan Timur Indonesia – dengan mitra PT Bukaka Singtel.

1999 Undang-undang nomor 36/1999, tentang penghapusan monopoli penyelenggaraan telekomunikasi.

2001 Pemerintah mengakhiri hak ekslusif TELKOM sebagai satu-satunya penyelenggara layanan telepon tidak bergerak kabel di Indonesia dan Indosat sebagai satu-satunya penyelenggara layanan sambungan langsung internasional (SLI).

2002 Telkom membeli seluruh saham Pramindo melalui 3 tahap, yaitu 30% saham pada saat ditandatanganinya perjanjian

jual-beli pada tanggal 15 Agustus 2002, 15% pada tanggal 30 September 2003 dan 55% saham sisanya pada tanggal 31 Desember 2004. Telkom menjual 12,72% saham Telkomsel kepada Singapore Telecom, dan dengan demikian Telkom memiliki 65% saham Telkomsel. Sejak Agustus 2002 terjadi duopoli penyelenggaraan telekomunikasi lokal.

2004 Pada tanggal 7 Juni 2004, TELKOM meluncurkan layanan SLI.

2.1.2 Visi dan Misi

2.1.2.1 Visi Telkom

To become a leading InfoCom player in the region

TELKOM berupaya untuk menempatkan diri sebagai perusahaan *InfoCom* terkemuka di kawasan Asia Tenggara, Asia dan akan berlanjut ke kawasan Asia Pasifik.

2.1.2.2 Misi Telkom

TELKOM mempunyai misi memberikan layanan " *One Stop InfoCom* " dengan jaminan bahwa pelanggan akan mendapatkan layanan terbaik, berupa kemudahan, produk dan jaringan berkualitas, dengan harga kompetitif.

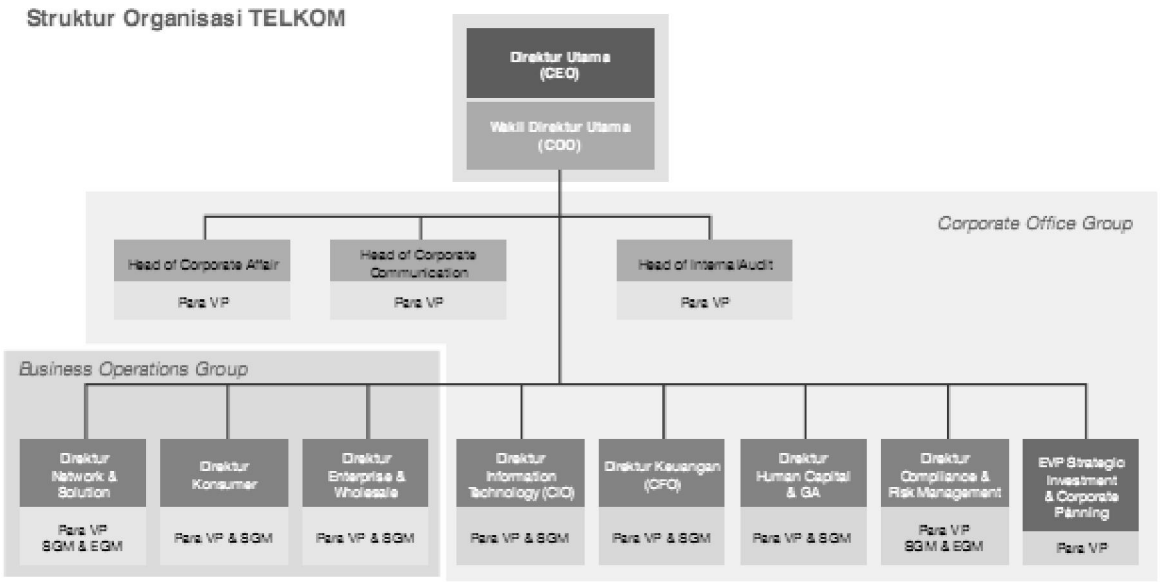
TELKOM akan mengelola bisnis melalui praktek-praktek terbaik dengan mengoptimalisasikan sumber daya manusia yang unggul, penggunaan teknologi yang kompetitif, serta membangun kemitraan yang saling menguntungkan dan saling mendukung secara sinergis.

2.1.3 Struktur Organisasi

Secara umum, organisasi TELKOM pada tahun 2007 terdiri dari Dewan Komisaris, Direksi dan Unit Bisnis. Dewan Komisaris dipimpin oleh Komisaris Utama bertugas mengawasi jalannya operasional perusahaan. Dalam menjalankan tugasnya Dewan Komisaris dibantu oleh beberapa komite yang mencakup Komite Audit, Komite Nominasi dan Remunerasi, serta Komite Pengkajian Perencanaan dan Resiko, Rapat Koordinasi gabungan antara Dewan Komisaris dengan Direksi dilakukan setiap dua minggu sekali.

Sejak RUPS Luar Biasa (RUPSLB) pada tanggal 28 Februari 2007, komposisi Direksi TELKOM berubah. Direktur TELKOM menjadi berjumlah delapan orang yang terdiri dari Direktur Utaman (CEO), Direktur Network & Solution, Direktur Konsumer, Direktur Enterprise & Wholesale, Direktur Keuangan (CFO) dan Direktur Human Capital & General Affair, Direktur Information & Supply (CIO) serta Direktur Compliance & Risk Management.

Organisasi TELKOM secara fundamental telah didesain dan dikembangkan untuk mencapai pertumbuhan dan pembangunan jangka panjang yang berfokus kepada kepuasan pelanggan, infrastruktur, penyedia layanan berkualitas, serta pada pendayagunaan sumber daya yang kompeten.



Gambar 2.1 Struktur Organisasi Telkom

2.1.4 Lima Pilar Bisnis

Untuk menjalankan bisnisnya dan memberikan layanan telekomunikasi kepada masyarakat, PT. Telkom memiliki lima pilar bisnis, yaitu :

1. Fixed Phone (TELKOM Phone)
    a. Personal Line
    b. Corporate Line
    c. Wartel & Telum

2. Mobile Phone (TELKOMSEL)
    a. Prepaid Services (simPATI)
    b. Postpaid Services (Halo)
3. Network & Interconnection (TELKOM Intercarier)
    a. Interconnection Services
    b. Network Leased Services
4. Data & Internet
    a. Leased Channel Service (TELKOM Link)
    b. Internet Service (TELKOMNet)
    c. VoIP Service (TELKOM Save & Global 017)
    d. SMS Service (from TELKOMSEL, TELKOMFlexi & TELKOM SMS)
5. Fixed Wireless Access (TELKOM Flexi)
    a. Prepaid Services (Flexi Trendy)
    b. Postpaid Services (Flexi Classy)

2.1.5 Budaya Perusahaan

THE TELKOM WAY 135 **s**ebagai budaya korporasi yang dikembangkan TELKOM merupakan bagian terpenting dari upaya perusahaan untuk meneguhkan hati, merajut pikiran, dan menyerasikan langkah semua Insan TELKOM dalam menghadapi persaingan bisnis InfoCom. Di dalamnya terkandung beberapa unsur, yang secara integral harus menjiwai insan TELKOM, yakni :

a. **1** (satu) asumsi dasar yang disebut Committed 2 U

b. **3** (tiga) nilai inti, mencakup :

1. *Customer Value*
2. *Excellent Service*
3. *Competent People*

c. **5** (lima) langkah perilaku untuk memenangkan persaingan, yang terdiri dari :

1. *Stretch The Goals*
2. *Simplify*
3. *Involve Everyone*
4. *Quality is My Job*
5. *Reward the Winners*

THE TELKOM WAY 135 adalah hasil penggalian dari perjalanan TELKOM dalam mengarungi lingkungan yang terus berubah, dan dikristalisasi serta dirumuskan dengan dirangsang oleh berbagai inspirasi dari perusahaan lain dan berbagai tantangan dari luar. Dengan akar yang kuat pada kesadaran kolektif organisasi, diharapkan *THE TELKOM WAY* 135 dapat cepat tertanam dalam jiwa insan TELKOM. TELKOM berharap dengan tersosialisasinya *THE TELKOM WAY* 135, maka akan tercipta pengendalian kultural yang efektif terhadap cara merasa, cara memandang, cara berpikir, dan cara berperilaku semua insan TELKOM.

2.1.6 Maskot Telkom

Maskot *Be Bee*



Gambar 2.2 Maskot TELKOM

1. Antena Lebah Sensitif terhadap segala keadaan dan perubahan
2. Mahkota Kemenangan
3. Mata yang Tajam dan Cerdas
4. Sayap Lincah dan Praktis
5. Tangan Kuning Memberikan Karya Yang Terbaik

2.2 Landasan Teori

2.2.1 *Knowledge Discovery*

Knowledge Discovery atau sering disebut *Data Mining* secara sederhana dapat dikatakan sebagai proses meng-*extract* atau menggali *(mining)* pengetahuan/informasi yang berharga *(Interesting knowledge)* dari sejumlah besar data baik yang disimpan di dalam *database, datawarehouse* maupun media penyimpanan informasi lainnya (Han, Kamber, 2001). Dari definisi tersebut didapatkan berbagai istilah yang beredar dan memliki arti hampir sama dengan *data mining* yaitu

knowledge mining from database, knowledge extraction, data/pattern analysis, data archeology, data dredging, information harvesting dan *business intelligence.*

2.2.1.1 Data Mining

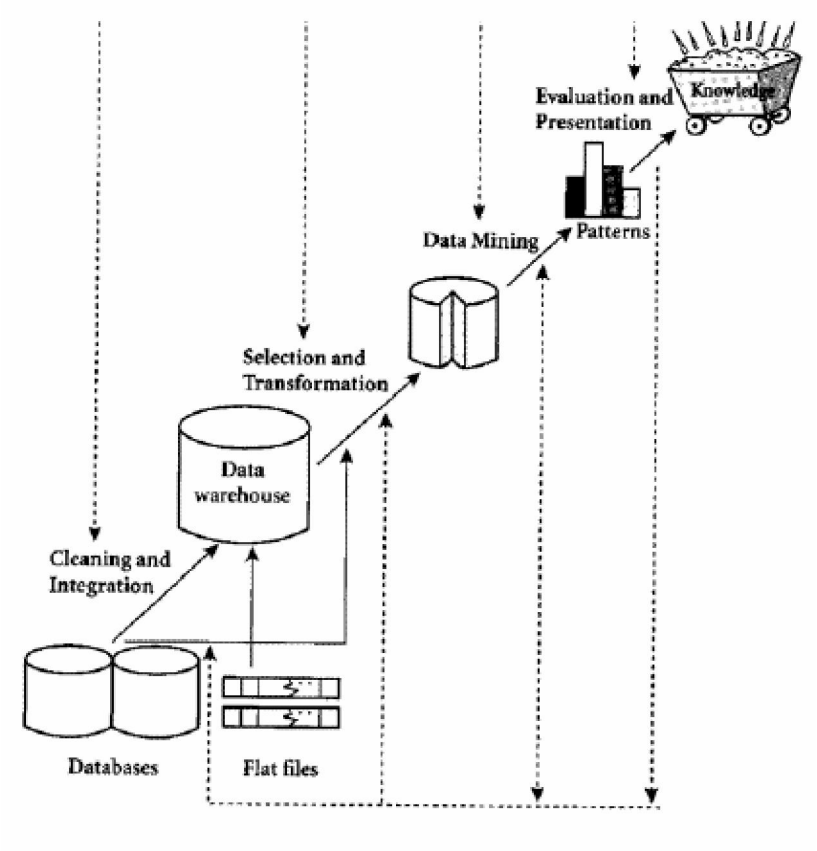
Data Mining yang sering disebut juga *Knowledge Discovery* atau *Intelligent Data Analysis*, merupakan suatu proses pencarian pola dari sekumpulan data. Proses tersebut dapat dilakukan secara *semiautomatic* maupun *automatic*. Pola yang ditemukan haruslah tepat guna dan bermanfaat, sehingga nantinya bisa diperoleh keuntungan dari pola tersebut. Dalam hal ini, yang diharapkan biasanya adalah keuntungan ekonomis, efektifitas dan efisiensi. Untuk dapat menemukan pola-pola tersebut, jumlah data yang tersedia setidaknya harus *sufficient*. Singkatnya, *data mining* bertujuan menghasilkan suatu informasi yang dapat berguna dari sebuah *database* yang besar. *Data mining* secara *semiautomatic* bisa dilakukan dengan melakukan analisa dari data-data masa lalu, menemukan pola-pola dari data tersebut yang nantinya dapat membantu dalam pengalisaan lebih lanjut, atau juga secara *automatic* yaitu dengan suatu alat pengambilan keputusan untuk memperkirakan suatu langkah yang dapat diambil di masa mendatangan sehingga dapat mendatangkan keuntungan.

Data mining secara *automatic* lebih banyak dipergunakan orang karena dapat melakukan *mining* dengan lebih baik daripada usaha analisa manusia biasa. Jika semua pelaku bisnis dapat mengolah data mereka

dengan baik, maka persaingan dalam pasar bisnis otomatis akan meningkat. Lebih jauh lagi, bila *data mining* yang dilakukan dapat akurat, cepat, dan dapat menghasilkan sesuatu yang tidak terpikirkan sebelumnya, maka hal ini nantinya dapat berdampak banyak pada ekonomi.

2.2.1.2 Komponen-komponen dan Tahapan D*ata Mining*

Sebenarnya *data mining* merupakan salah satu tahap yang terdapat di dalam *knowledge discovery*. Sebagai sebuah proses, *knowledge discovery* dapat digambarkan seperti gambar 2.3 di bawah ini [5] :



Gambar 2.3 Tahap-tahap dalam proses *knowledge discovery*

a. *Data cleaning*:

Untuk menghilangkan data *noise* (data yang tidak relewan/berhubungan langsung dengan tujuan akhir proses *data mining*, misal: *data mining* yang bertujuan untuk menganalisa hasil penjualan, maka data-data dalam kumpulan seperti ”nama pegawai”, ”umur”, dan sebagainya dapat di*ignore*) dan tidak konsisten.

b. *Data integration*

Untuk menggabungkan *multiple data source*

c. *Data selection*

Untuk mengambil data yang sesuai untuk keperluan analisa

d. *Data transformation*

Untuk mentransformasikan data ke dalam bentuk yang lebih sesuai untuk di *mining.*

e. *Data mining*

Proses terpenting dimana metode tertentu diterapkan untuk menghasilkan *data pattern*.

f. *Pattern evaluation*

Untuk mengidentifikasi apakah *interenting patterns* yang didapatkan sudah cukup mewakili *knowledge* berdasarkan perhitungan tertentu.

g. *Knowledge presentation*

Untuk mempresentasikan *knowledge* yang sudah didapatkan dari *user*

Arsitektur utama dari sebuah sistem *data mining* pada umumnya mengandung unsur-unsur sebagai berikut :

a. *Database, data warehouse,* atau media penyimpanan informasi

Media dalam hal ini bisa jadi berupa *database, data warehouse, spreadsheets,* atau jenis-jenis penampung informasi lainnya. *Data cleaning* dan *data intregration* dapat dilakukan pada data-data tersebut.

b. *Database* atau *data warehouse server*

Database atau *data warehouse server* bertanggung jawab untuk menyediakan data yang relevan berdasarkan permintaan dari *user* pengguna *data mining.*

c. *Data mining engine*

Yaitu bagian dari *software* yang menjalankan program berdasarkan algoritma yang ada

d. *Pattern evaluation module*

Yaitu bagian dari *software* yang berfungsi untuk menemukan *pattern* atau pola-pola yang terdapat di dalam *database* yang diolah sehingga nantinya proses *datamining* dapat menemukan *knowledge* yang sesuai

e. *Graphical user interface*

Bagian ini merupakan sarana antara *user* dan sistem *data mining* untuk berkomunikasi, dimana *user* dapat berinteraksi dengan sistem melalui *data mining query*, untuk menyediakan informasi yang dapat membantu dalam pencarian *knowledge*. Lebih jauh lagi, bagian ini mengijinkan *user* untuk melakukan *browsing* pada *database* dan *data warehouse*, mengevaluasi *pattern* yang telah dihasilkan, dan menampilkan *pattern* tersebut dengan tampilan yang berbeda-beda [2].

2.2.1.3 Association Rules

Association rules mining adalah suatu prosedure untuk mencari hubungan antar *item* suatu *data set* yang ditentukan. (Han, Kamber, 2001).

Seperti yang telah disebutkan sebelumnya, *association rules mining* mencari dan menemukan hubungan antar *item* yang ada pada suatu *dataset*. Penerapan *data mining* dengan *association rule* bertujuan menemukan informasi *item-item* yang saling berhubungan dalam bentuk *rule*, dengan demikian *association rule* paling tepat diterapkan pada *market basket analysis.* Berikut adalah penjelasan lebih mendalam mengenai *association rules.*

2.2.1.3.1 Konsep dasar pengertian *association rules*

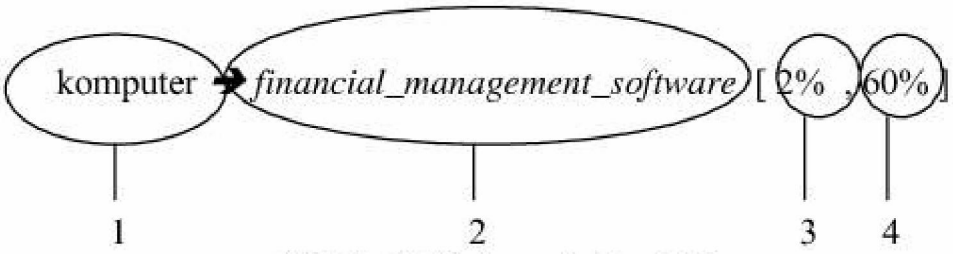
Jika diasumsikan bahwa semesta sebagai suatu kumpulan *item* yang tersedia di toko, maka setiap *item* tersebut memiliki *Boolean variable* yang merepresentasikan keberadaan atau ketidakberadaan dari *item* tersebut. Setiap transaksi kemudian dapat dipresentasikan oleh suatu nilai *Boolean* yang diberikan pada setiap variable. Nilai *Boolean* tersebut bisa dianalisa untuk pola belanja yang menunjukkan *item* yang sering dibeli bersamaan. Pola tersebut dapat dipresentasikan dengan bentuk *association rules*. Sebagi contoh, *customer* yang biasanya membeli komputer juga membeli *financial_management_software* ditunjukan sebagai berikut :

komputer à *financial_management_software*[*suppor*=2%,*conficende*=60%]

Dalam pencarian *association rules,* diperlukan suatu variabel ukuran yang dapat ditentukan oleh *user,* untuk mengatur batasan sejauh mana dan sebanyak hasil *output* yang diinginkan oleh *user*. *Support* dan *confidence* adalah dua ukuran kepercayaan yang menunjukkan kepastian dan tingkat kegunaan suatu *rule* yang ditemukan. *Support=2%* berarti bahwa sebanyak 2% dari semua jumlah transaksi yang dianalisis menunjukkan bahwa computer dan *financial_management_software* dibeli bersamaan. Sedangkan

confidence=60% berarto bahwa 60% dari *customers* yang membeli sebuah computer pasti juga membeli *software*.

Pada gambar 2.4 berikut ini, ditunjukkan istilah-istilah yang sering dipakai untuk mempresentasikan tiap bagian dari *association rules* :



Gambar 2.4 Associaiton Rules

Association rules di atas dapat di baca seraca sederhana menjadi "**IF** *buys komputer* **THEN** *buys financial_management_software in 2% of rows, in 60% of cases"*

Keterangan istilah:

1: *Antecedent, left-hand side (LHS), body*

2: *Consequent, right-hand side (RHS), head*

3: *Support, frequency* (Sebagaimana besar bagian data pada *left-hand side* dan *right-hand side* muncul secara bersamaan)

4: C*onfidence, strength* (jika *left-hand side* muncul, seberapa besar *right-hand side* muncul)

2.2.1.3.2 Konsep dasar pencarian *association rules*

a. Misalkan I={i_1, i_2,.....,i_m} adalah sekumpulan *items*. Contoh: I={i_1,i_2,i_3,i_4,i_5}. Secara umum notasi I dapat dibaca sebagai semua jenis *items* yang oernah dibeli dalam berbagai transaksi selama periode tertentu dengan m sebagai nilai yang menunjukkan berapa macam *items* yang ada.

b. Misalkan juga D adalah *database transactions* dimana tiap transaksi T terdiri dari sekumpulan *items* sehingga $T \subseteq l$. Jadi D datapa berupa D={T100,T200,T300,T400,T500.T600,T700,T800,T900} dan sebagai contoh salah satu transaksi T100={i1,i2,i5}. Tanda $T \subseteq l$ (baca*T* merupakan himpunan bagian atau *subset* dari semua anggota *l*) berarti bahwa semua anggota T harus merupakan sebagian atau seluruh dari semua anggota l. Di sini dapat dilihat bahwa {i1,i2,i5} $\subseteq$ {i1,i2,i3,i4,i5}. Contoh dari *database transaction* D dapat dilihat pada tabel 2.1

c. Setiap transaksi T dikatakan mengandung A jika dan hanya jika $A \subseteq T$. Sebuah *association rules* merupakan sebuah implikasi dari bentuk A à B, di mana $A \subseteq l$, $B \subseteq l$ dan $A \cap B = \varnothing$. Sebagai contoh sebuah transaksi T100 yang memiliki *items* {i1,i2,i5} dapat memiliki sebuah *association rule* dalam bentuk i1^i2 à i5 dimana :

1. A={i1,i2}, dengan catatan A$\subseteq l$ atau semua anggota A harus merupakan sebagian dan tidak pernah mencapai seluruh dari semua anggota l.
2. B={i5}, dengan catatan $B \subseteq l$ atau semua anggota B harus merupakan sebagian dan tidak pernah mencapai seluruh dari smeua anggota l.
3. {i1,i2}$\cap${i5}=$\varnothing$ karena berdasarkan syarat terakhir yaitu A$\cap$B=$\varnothing$ yang artinya tidak boleh ada satuoun anggota dari A dan B yang sama.

d. *Rule* A à B berada pada sekumpulan transaksi D dengan *support* s, di mana s adalah persentase transaksi di dalam D yang mengandung (A $\cup$B) sehingga hal ini merupakan probabilitas P(A$\cup$B). *Rule* A à B juga mempunyai *confidence c* di dalam sekumpulan transaksi D, di mana c adalah persentasi transaksi di dalam D jika transaksi tersebut mengandung A juga mengandung B sehingga ini merupakan probabilitas bersyarat, P(B|A).

Pada umumnya *association rules* yang ditemukan dikatakan menarik apabila *rule* tersebut memenuhi baik *minimum support* maupun *minipun confidence* dapat digambarkan sebagai berikut :

$$support(A \Rightarrow B) = P(A \cup B)$$
$$= \left(\frac{jumlah\ kemunculan\ (A \cup B)}{jumlah\ transaksi}\right) * 100\%$$

$$confidence(A \Rightarrow B) = P(B|A) = \left(\frac{jumlah\ kemunculan\ (A \cup B)}{jumlah\ kemunculan\ A}\right) * 100\%$$

Rule yang memenuhi baik *minimum support* maupun *minimum confidence* disebut juga *strong rule*. *Support* dan *confidence* dituliskan dengan nilai antara 0% sampai 100%. Sekumpulan *items* disebut juga *itemsets*. Sebuah *itemsets* yang mengandung k *items* adalah k-*itemset*. Sekumpulan *items* {komputer,*financial_management_software*} adalah 2-*itemsets*. Jumlah kejadian muncul *itemset* tersebut, atau disebut juga *frequency, support* count atau count dari itemset. Sebuah *itemset* akan memenuhi *minimum support*, jika jumlah kejadian munculknya *itemset* tersebut lebih besar atau sama dengan *minimum support* yang telah ditentukan. Jika suatu *itemset* memenuhi *minimum support*, maka *itemset* tersebut disebut juga *frequent itemset*. Suatu set dari *frequent* k-*itemset* pada umumnya dinotasikan dengan L_k.

Langkah-langkah untuk melakukan perhitungan untuk menemukan *association rules* terdiri dari dua langkah utama, yaitu :

Langkah 1: temukan semua *frequent itemsets*, yaitu *itemsets* yang memenuhi *minimum support* yang telah ditentukan

Langkah 2:gunakan *frequent itemsets* tersebut untuk menghasilkan *association rules*.

2.2.1.4 Algoritma Apriori

Apriori adalah suatu Algoritma yang sudah sangat dikenal dalam melakukan pencarian *frequent itemset* untuk mendapatkan *association rules.* Sesuai dengan namanya, algoritma ini menggunakan *prior knowledge* mengenai *frequent itemset properties* yang telah diketahui sebelumnya, untuk memproses informasi selanjutnya. Apriori menggunakan pendekatan secara *iterative* yang disebut juga sebagai *level-wise search* dimana k-*itemset* digunakan untuk mencari (k+1)-*itemset*. Pertama-tama dicari *set* dari *frequent* 1-*itemset, set* ini dinotasikan sebagai L_1. L_1 digunakan untuk menemukan L_2, kemudian set dari *frequent* 2-*itemset,* digunakan untuk menemukan L_2, kemudian set dari *frequent* 2-*itemset* digunakan untuk menemukan L_3, dan seterusnya, sampai tidak ada lagi *frequent* k-*itemset* yang dapat ditemukan. Proses untuk menemukan setiap L_k membutuhkan satu kali pemeriksaan menyeluruh pada *database,* yang artinya jika k=4, maka pemeriksaan terhadap *database* dilakukan sebanyak empat kali.

Untuk meningkatkan efiseinsi dari pencarian k-*itemset,* maka digunakan suatu metode yang dinamakan *apriori property,* metode ini dapat mengurangi lingkup pencarian sehingga waktu pencarian dapat dipersingkat. Pengertian *Apriori property* adalah suatu kondisi di mana semua *subset* dari *frequent itemset* yang tidak kosong haruslah juga *frequent, frequent* yang dimaksud disini adalah semua *itemset* yang memenuhi *minimum support* yang ditentukan oleh *user*. *Apriori property*

didasarkan pada observasi berikut. Berdasarkan definisi yang disebutkan tadi, jika suatu *itemset l* tidak memenuhi *minimum support* (min_sup) yang ditentukan, maka *l* tidak *frequent*, atau dinotasikan dengan *P(l)<min_sup.* Jika suatu *item A* dimasukan ke dalam *l*, maka hasil *itemset* (I∪A) tidak dapat terjadi lebih daripada jumlah *itemset l*. Oleh karena itu I ∪ A tidak *frequent* juga, dengan kata lain, *P(l∪A)<min_sup*. Secara sederhana dapat dikatakan jika suatu *itemset* tidak memenuhi syarat, maka semua *superset* (*itemset* lain yang dibentuk dengan mengandung unsur *itemset* tersebut) dari *itemset* tersebut tidak mungkin akan memenuhi syarat juga.

Berikut penjelasan lebih lanjut bagaimana *apriori property* digunakan dalam algoritma ini. Penjelasan berikut merupakan proses yang terdiri dari dua langkah, langkah pertama adalah *join* dan kedua adalah *prune* mengenai bagaimana *frequent* (k-1)-*itemsets* (L_{k-1}) digunakan untuk menemukan *frequent* k-*itemsets* (L_k).

1. Langkah *join* : untuk menemukan L_k, suatu *set* dari *candidate* k-*itemset* di hasilkan dengan cara men-*join*-kan L_{k-1} dengan dirinya sendiri. Set *candidate* ini dihasilkan sebagai C_k. Misalkan l_1 dan l_2 adalah *itemsets* di dalam L_{k-1}. Notasi $l_i[j]$ menunjukkan *item* ke-j yang ada pada *l*. Sebagai contoh l_1[k-1] menunjukkan *item* kedua terakhir yang ada pada *l1*. Perlu ditekankan bahwa algoritma ini mempunyai aturan bahwa *item* di dalam transaksi atau *itemset* telah diurutkan secara *lexicograohic order* terlebih dahulu.

Selanjutnya *join* yang dinotasikan dengan $L_{k-1} \bowtie L_{k-1}$ dijalankan, di mana anggota-anggota L_{k-1} yang berupa *itemset-itemset* dikatakan dapat di-*join* jika *firts* (k-2)-*items* dari L_{k-1} sama. Jadi anggota l_1 dan l_2 dari L_{k-1} akan di-*join* jika $(l_1[1]=l_2[1])\wedge(l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$. Kondisi $(l_1[k-1] < l_2[k-1])$ bertujuan untuk memastikan tidak ada *itemset* kembar yang dihasilkan dari *join* l_1 dan l_2 yaitu dalam format $l_1[1]l_1[2]\ldots.l_1[k-1]l_2[k-1]$.

2. Langkah *prune* : Ck adalah *superset* dari L_k dimana setiap anggotanya bisa *frequent* ataupun tidak, tetapi semua *frequent* k-*itemset* termasuk dalam C_k. Proses *scan* terhadap *database* yang dilakukan untuk menentukan jumlah kemunculan setiap *candidate* yang ada di dalam C_k akan menentukan L_k (sebagai contoh, semua *candidate* yang memiliki *support count* tidak kurang dari *minimum support count* yang ditentukan disebut *frequent*, yang artinya juga memenuhi syarat untuk masuk menjadi L_k). ada kemungkinan C_k dapat mengandung *candidate* dengan jumlah yang sangat besar, sehingga dapat menyebabkan proses penghitungan C_k menjadi berat. Untuk mengurangi jumlah *candidate* C_k, *Apriori property* digunakan sebagai berikut. Semua (k-1) *itemset* yang tidak *frequent* dinyatakan tidak dapat menjadi *subset* dari sebuah *frequent* k-*itemset*. Oleh karena itu, jika ada (k-1)-*subset* dari sebuah *candidate* k-*itemset* tidak termasuk dalam L_{k-1}, maka *candidate*

tersebut tidak mungkin *frequent* juga dan oleh karena itu dapat di-*remove* dari $C_{k.}$

Berikut adalah algoritma *apriori* yang akan digunakan untuk mencari *frequent itemsets* menggunakan iterasi. *Output* yang akan dihasilkan oleh program adalah semua *frequent itemset* yang memenuhi *minimum support*. Adapun *input* yang harus disediakan oleh *user* antara lain:

a. *Database* dari suatu transaksi, dalam algorithma akan dinotasikan sebagai D
b. *Minimum support* yang ditentukan oleh *user*, dinotasikan sebagai *min_sup*, hanya *item* yang memenuhi *minimum support* ini saja yang akan diperhitungkan sebagai *frequent itemset*.

```
(1) L1 = find_frequent_1=itemsets(D);
(2)  for (k = 2;Lk-1≠∅;k++)
(3)  {
(4)   Ck = apriori_gen(Lk-1,min_sup);
(5)   for each transaction t ∈ D // scan D for counts
(6)    {
(7)     Ct = subset(Ck,t); //get the subsets of t that
   are candidates
(8)       for each candidate c ∈ Ct
(9)       c.count++;
(10)   }
(11)   Lk = {c ∈ Ck|c.count ≥ min_sup}
(12)}
(13)return L = ∪kLk;
```

```
procedure apriori_gen(L_{k-1}:frequent (k-1)-itemsets; min_sup:
minimum support threshold)
   (1) for each itemset l_1 ∈ L_{k-1}
   (2)   for each itemset l_2 ∈ L_{k-1}
   (3)    If (l_1[1] = l_2[1]) ∧ (l_1[2] = l_2[2]) ∧ … ∧ (l_1[k-
      2] = l_2[k-2]) ∧ (l_1[k-1] < l_2[k-1]) then
   (4)    {
   (5)      c =l_1 ⋈ l_2; //join step:generate candidates
   (6)      if has_infrequent_subset(c,L_{k-1}) then
   (7)        delete  c;  //prune  step:remove  unfruitful
      candidate
   (8)      else add c to C_k;
   (9)    }
   (10)return C_k;
procedure has_infrequent_subset(c:candidate k –
itemset;L_k:frequent(k-1)-itemsets); // use prior knowledge
  (1)  for each (k-1)-subset s of c
  (2)       if s ∉ L_{k-1} then
  (3)          Return TRUE;
  (4)  Return FALSE;
```

Algoritma Apriori. Sumber : Han, Jiawei, and Micheline Kamber. *Data Mining* : *Concepts and Techniques.* Morgan Kaufmann, 2001, p.235(telah dional kembali)

Berikut ini diberikan penjelasan lebih lanjut melalui contoh, pemakaian algoritma *apriori* untuk menemukan *association rule* dengan *Market Basket Analysis*. Contoh tabel berikut adalah merupakan *transaction database* D dari sebuah toko. Ada sembilan transaksi di dalam *database* ini.

Tabel 2.1 contoh Tabel *Transaction Database* D

Kode Transaksi	***Item* yang dibeli**
T100	L1,l2,l5
T200	L2,l4
T300	L2,l3
T400	L1,l2,l4
T500	L1,l3
T600	L2,l3
T700	L1,l3
T800	L1,l2,l3,l5
T900	L1,l2,l3

Sumber: Han, Jiawei, and Micheline Kamber. *Data Mining* : *Concepts and Techniques*. Morgan Kaufmann, 2001, p.232(telah dional kembali)

Berikut adalah langkah-langkah algoritma Apriori untuk menemukan *frequent itemsets*:

a. Pada iterasi dari algorithma, setiap *item* adalah anggota *set* dari *candidate* 1-*itemset*, C_1. Algoritma akan secara langsung memeriksa semua transaksi yang ada di dalam *database* D untuk dapat menghitung kejadian munculnya setiap *item*.

b. Jika diasumsikan bahwa *minimum support count* yang dibutuhkan adalah 2 (dalah hal ini *min_sup*=2/9=22%). Set dari *frequent* 1-*itemsets*, L_1 dapat diperoleh yaitu semua *candidate* 1-*itemset* atau C_1 yang memenuhi *minimum support*. Dalam contoh ini kebetulan semua anggota C_1 memiliki *support count* ≥ *minimum support count*. Jadi tidak ada anggota C_1 yang di-*remove* pada L_1.

c. Untuk menemukan *frequent* 2-*itemset* atau $L_{2,}$ pertama-tama algoritma ini menggunakan join $L_1 \bowtie L_1$ untuk menghasilkan *candidate* 2-*itemsets* atau C_2. C_2 merupakan hasil kombinasi join dari L_1. Jumlah *itemset* yang ada pada C_2 yaitu $\left(|\frac{L_1}{2}|\right) = \binom{5}{2} = \frac{5!}{(5-2)!2!} = 10$

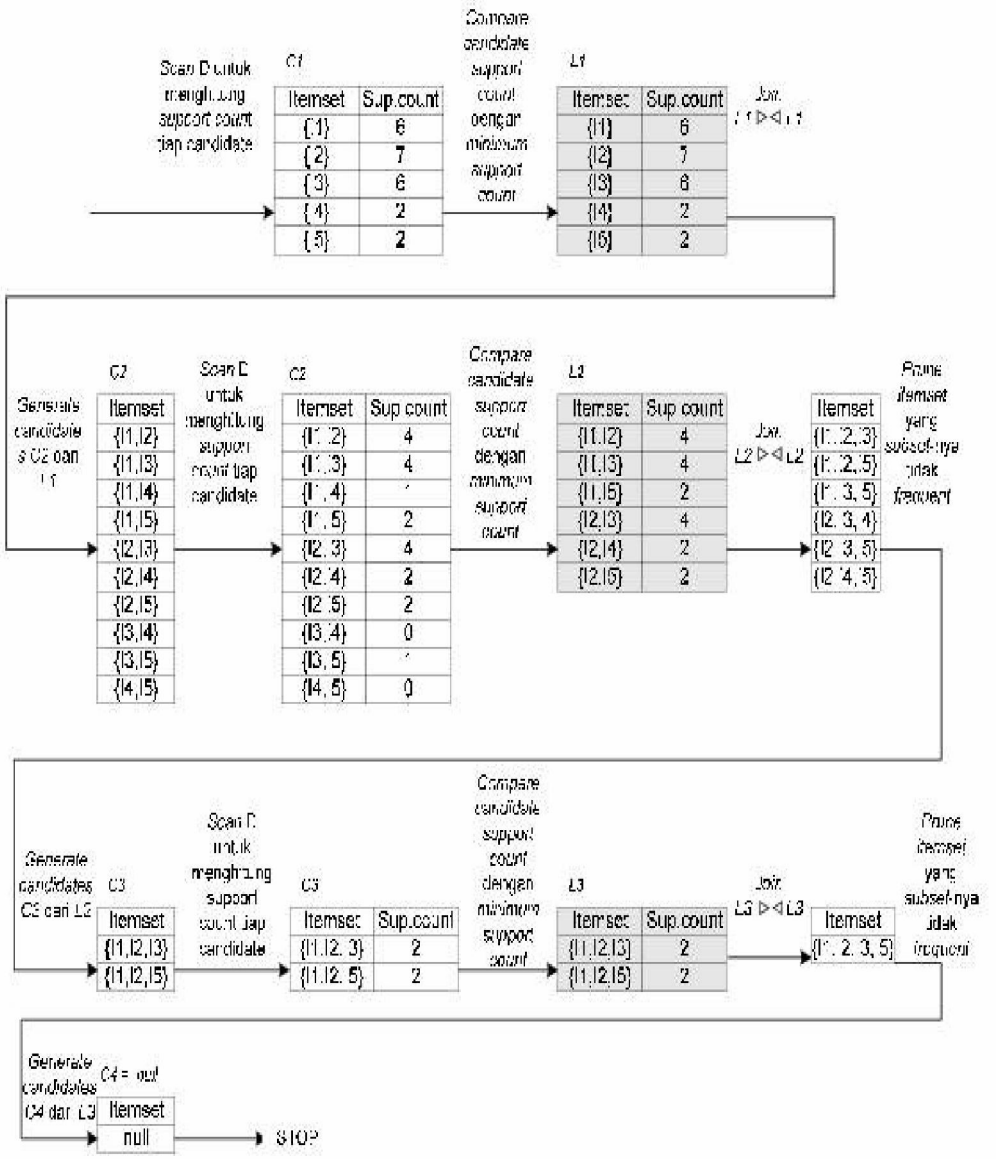
d. Kemudian, transaksi yang ada pada *database* D diperiksa atau di-*scan* dan *support count* dari tiap *candidate itemset* yang ada di C_2 ditambahkan, seperti yang ditunjukkan pada gambar 2.5 di bawah ini. Dapat dilihat pada kolom sebelah kanan tiap *candidate itemset* yang ada dalam C_2 tertera *support count* dari masing-masing *candidate* 2-*itemset*. (lihat tabel pada posisi tengan gambar 2.5)

e. Set dari 2-*itemset* atau L_2, dapat ditentukan, yaiktu semua *candidate* 2-*itemsets* yang memenuhi *minimum support*. Dapat dilihat pada gambar 2.5 bahwa *itemset-itemset* pada C_2 yang memiliki *support count* lebih kecil cari 2 di-*remove* sehingga yang tersisa adalah *itemset* yang memiliki *support count* ≥ *minimum support count*

f. Proses untuk menghasilkan atau meng-*generate set* dari candidate 3-*itemsets* atau $C_{3,}$ dijelaskan secara lebih detail dalam dua langkah yaitu *join* dan *prune* di dalam *apriori property*.

Join step: Langkah pertama di dalam mendapatkan C_3, yaitu dengan mengkombinasikan L_2 dengan L_2 melalui proses *join* $L_2 \bowtie L_2$, menghasilkan {{l1,l2,l3}, {l1,l2,l5}, {l1,l3,l5}, {l2,l3,l4}, {l2,l3,l5}, {l2,l4,l5}}. Detail dari $L_2 \bowtie L_2$ dapat dilihat pada gambar 2.6.

Prune step : Berdasarkan pada ketentuan *apriori property* yang menagtakan bahwa semua *subset* dari masing-masing *frequent itemset* diatas juga harus *frequent*, maka dapat ditentukan bahwa keempat *candidate* terakhir yaitu {{l1,l3,l5}, {l2,l3,l4}, {l2,l3,l5}, {l2,l4,l5}} tidak mungkin *frequent*. Oleh karena itu, keempat *candidate* tadi harus di-*remove* dari C_3. Dengan demikian dapat menghemat waktu yang tidak dperlukan untuk melakukan *scan* terdapat *database*, ketika akan menentukan L_3. Perlu diingat, bahwa ketika diperoleh *candidate* k-*itemset* atau C_k setelah langkah *join*, maka perlu dicek terlebih dahulu, apakah (k-1)-*subsets* dari masing-masing anggota C_k adalah *frequent*, sehubungan dengan algoritma Apriori menggunakan strategi *level-wise search*. Jadi *apriori property* ini dilakukan secara *level-wise search* pada L_k dimana k dimulai dari level 2 yaitu L_2 dan seterusnya.



Gambar 2.5 contoh pencarian *candidate* dan *frequent itemsets* menggunakan Apriori

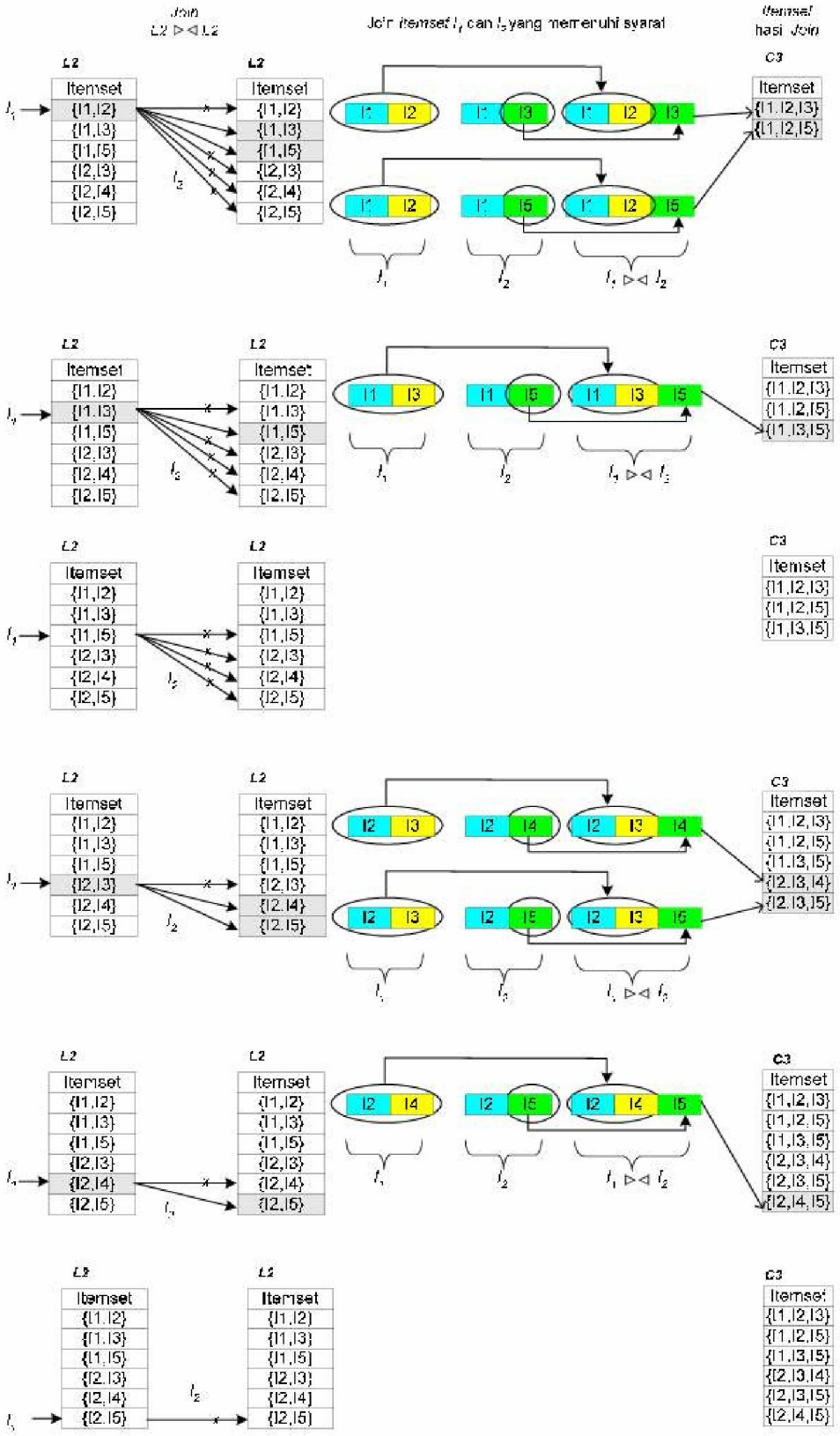
Sumber : Han, Jiawei, and Micheline Kamber. *Data Mining : Comcepts dan Techniques.* Morgan Kaufmann, 2001, P.233 (telah diolah kembali)

g. Semua transaksi di dalam D diperiksa atau di-*scan* untuk menentukan L_3 yaitu terdiri dari *candidate* 3-*itemsets* di dalam C_3 yang memenuhi *minimum support* yang sudah ditentukan

h. Algoritma akan melakukan kombinasi antara L_3 dan L_3 untuk menghasilkan set dari *candidate* 4-*itemsets*, C_4. Walaupun hasil *join* adalah {l1,l2,l3,l5}, *itemset* ini di-*prune* karena salah satu (k-1)-subset-nya yaitu {l2,l3,l5} tidak *frequent*. Dengan demikian $C_4 = \varnothing$, dan jalannya proses algoritma ini digentikan karena telah menemukan semua *frequent itemsets*. Pada gambar 2.5 semua *frequent itemsets* dapat dilihat pada tabel dengan warna gelap.

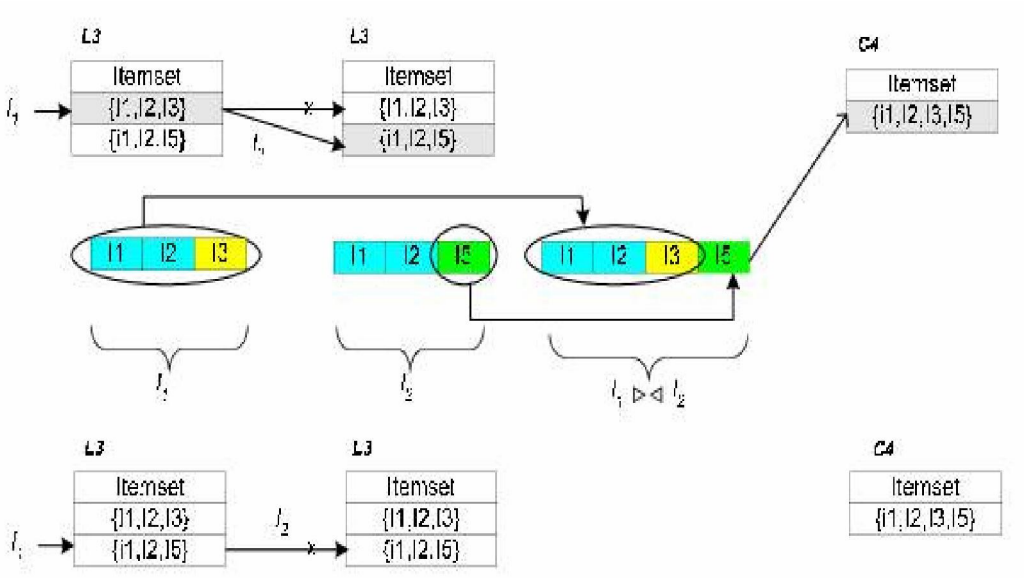
Di atas telah disinggung mengenai *Apriori property*, dua langkah di bawah ini adalah gambaran bagaimana *Apriori property* digunakan untuk membantu mempercepat proses menemukan *frequent itemset*, dengan cara mengurangi *candidate itemset*. Dua langkah tersebut dapad dilihat pada penjelasan di bawah ini :

1. Langkah *join* : $C_3 = L_2 \bowtie L_2$= {{l1,l2}, {l1,l3}, {l1,l5}, {l2,l3}, {l2,l4}, {l2,l5}} $\bowtie$ {{l1,l2}, {l1,l3}, {l1,l5}, {l2,l3}, {l2,l4}, {l2,l5}} = {{l1,l2,l3}, {l1,l2,l5}, {l1,l3,l5}, {l2,l3,l4}, {l2,l3,l5}, {l2,l4,l5}}. Untuk detail dari langkah *join* dapat dilihat pada gambar :



Gambar 2.6 Contoh Langkah *join* $L_2 \bowtie L_2$

Pada contoh gambar 2.6 *join* $L_2 \bowtie L_2$ di atas dapat dilihat di mana proses *join* dilaksanakan jika memenuhi syarat $(l_1[1]=l_2[1])\wedge(l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$. Kotak biru menunjukkan kondisi $(l_1[1]=l_2[1])\wedge(l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2])$. Kotak kuning pada l1 menunjukkan li[k-1] dan kotak hijau pada l2 menunjukkan l2[k-1]. Secara sederhana, jika sebuah *itemset* l1 dan sebuah *itemset* l2 memenuhi syarat (lihat baris l1 dan l2 yang berlatar belakang abu-abu), maka hasil *join* dilakukan hanya dengan memindahkan kota hijau tadi ke bagian paling kanan dari itemset li (lihat kotak hijau dengan garis tepi putus-putus). Demikian juga misalnya untuk $L_3 \bowtie L_3$ dapat dilihat pada gambar :



Gambar 2.7 contoh langkah join $L_3 \bowtie L_3$

Perlu di catat bahwa C_3 atau C_4 hasil *join* diatas berjumlah *candidate* yang final karena belum melalui langkah *prune*.

2. Langkah *prune* menggunakan *apriori property* dengan syarat semua *nonempty subsets* dari sebuah *frequent itemset* juga harus *frequent*. Di bawah ini merupakan *itemset* dari hasil *join* $L_2 \bowtie L_2$ yaitu C_3 yang masih belum final di mana akan dilakukan *prune* jika tidak memenuhi syarat di atas.
   a. 2-*item subsets* dari {I1,I2,I3} adalah {I1,I2}, {I1,I3} dan {I2,I3}. Semua 2-*item subsets* dari {I1,I2,I3} adalah anggota dari L_2. Oleh karena itu, {I1,I2,I3} pantas masuk C_3.
   b. 2-*item subsets* dari {I1,I2,I5} adalah {I1,I2}, {I1,I5} dan {I2,I5}. Semua 2-*item subsets* dari {I1,I2,I5} adalah anggota dari L_2. Oleh karena itu, {I1,I2,I5} pantas masuk C_3.
   c. 2-*item subsets* dari {I1,I3,I5} adalah {I1,I3}, {I1,I5} dan {I3,I5}. {I3,I5} bukan anggota dari L_2, dan oleh karena itu tidak *frequent*. Maka {I1,I3,I5} tidak pantas masuk C_3.
   d. 2-*item subsets* dari {I2,I3,I4} adalah {I2,I3}, {I2,I4} dan {I3,I4}. {I3,I4} bukan anggota dari L_2, dan oleh karena itu tidak *frequent*. Maka {I1,I3,I4} tidak pantas masuk C_3.
   e. 2-*item subsets* dari {I2,I3,I5} adalah {I2,I3}, {I2,I5} dan {I3,I5}. {I3,I5} bukan anggota dari L_2, dan oleh karena itu tidak *frequent*. Maka {I2,I3,I5} tidak pantas masuk C_3.
   f. 2-*item subsets* dari {I2,I4,I5} adalah {I2,I4},{ I2,I5} dan {I4,I5}. {I4,I5} bukan anggota dari L_2, dan oleh karena itu tidak *frequent*. Maka {I2,I4,I5} tidak pantas masuk C_3.

3. Oleh karena itu C_3 yang tersisa setelah proses *pruning* adalah {{l1,l2,l3}, {l1,l2,l5}}. Inilah C_3 yang final setelah melalui langkah *prune*. [3]

2.2.2 Data

Data adalah Kenyataan yang menggambarkansuatu kejdian-kejadian dankesatuan kenyataan. Dan merupakan suatu istilah yang berbentuk jamak dari kata "*datum*" yang berarti fakta atau bagian dari fakta yang mengandung arti yang menghubungkan dengan kenyataan, simbol-simbol, gambar-gambar, kata-kata, angka-angka, huruf-huruf yang menunjukan suatu ide, objek, kondisi dan situasi.

Menurut *the liang gie*: "Data atau bahan keterangan adalah hal atau peristiwa kenyataan lainnya apapun yang mendukung suatu pengetahuan untuk dijadikan dasar guna penyusunan keterangan pembuatan kesimpulan atau penetapan keputusan, atau data ibarat bahan mentah yang melalui pengolahannya tertentu lalu menjadi keterangan (informasi)".

2.2.3 *Database*

Database adalah kumplan data yang saling berkaitan, berhubungan yang disimpan secara bersama-sama sedemikian rupa tanpa pengulangan yang tidak perlu, untuk memenuhi berbagai kebutuhan. Data-data ini harus mengandung semua informasi untuk mendukung semua kebutuhan sistem.

Proses dasar yang dimiliki oleh database ada 4, yaitu:

1. Pembuatan data-data baru *(create database)*

2. Penambahan data *(insert)*

3. Mengubah data *(update)*

4. Menghapus data *(delete)*.

Sistem Manajemen Database *(Database Management System)* merupakan sistem pengoprasian dan sejumlah data pada komputer. Dengan sistem ini dapat merubah data, memperbaiki data yang salah dan menghapus data yang tidak dapat dipakai.

Sistem manajemen *database* merupakan suatu perluasan *software* sebelumnya mengenai *software* pada generasi komputer yang pertama. Dalam hal ini data dan informasi merupakan kesatuan yang saling berhubungan dan berkerja sama yang terdiri dari: peralatan, tenaga pelaksana dan prosedur data. Sehingga pengolahan data ini membentuk sistem pengolahan data. Peralatan dalam hal ini berupa perangakat keras *(hardware)* yang digunakan, dan prosedur data yaitu berupa perangakat lunak *(software)* yang digunakan dan dipakai untuk mengalokasikan dalam pembuatan sistem informasi pengolahan *database*.

2.3 Metodologi Yang Digunakan

Pada Sub bab ini akan dijelaskan metodologi yang akan digunakan. Metodologi yang digunakan adalah Object Oriented dengan menggunakan UML

2.3.1 Pengertian UML

Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. [http://id.wikipedia.org/wiki/UML]

2.3.2 Jenis Diagram UML

1. **Use Case Diagram**

   Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang atau sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

   Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

   Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case*

yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

2. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut atau properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda atau fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama dan stereotype
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

- *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
- *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
- *Public*, dapat dipanggil oleh siapa saja

3. Statechart Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan dari satu *state* ke *state* lainnya suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu satu *class* dapat memiliki lebih dari satu *statechart diagram*.

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring.

4. Activity Diagram

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka

berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem dan interaksi antar subsistem secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

5. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem termasuk pengguna, *display*, dan sebagainya berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu.

Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

6. Collaboration Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*.

Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

7. Component Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil.

8. Deployment Diagram

Deployment atau physical *diagram* menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak pada mesin, server atau piranti keras apa, bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal [6]

2.4 Software Pendukung Aplikasi

2.4.1 Sekilas Mengenai Visual Basic

Menurut artikel yang terdapat pada http://wikipedia.org/wiki/Visual_Basic, Microsoft Visual Basic merupakan sebuah bahasa pemrograman yang bersifat *event driven* dan menawarkan *Integrated Development Environment* (IDE) visual untuk membuat program aplikasi berbasis sistem operasi Microsoft Windows dengan menggunakan model pemrograman Common Object Model (COM). Visual Basic merupakan turunan bahasa BASIC dan menawarkan pengembangan aplikasi komputer berbasis grafik dengan cepat, akses ke basis data menggunakan Data Access Objects (DAO), Remote Data Objects (RDO), atau ActiveX Data Object (ADO), serta menawarkan pembuatan kontrol ActiveX dan objek ActiveX. Beberapa bahasa skrip seperti Visual Basic for Applications (VBA) dan Visual Basic Scripting Edition (VBScript), mirip seperti halnya Visual Basic, tetapi cara kerjanya yang berbeda.

Para programmer dapat membangun aplikasi dengan menggunakan komponen-komponen yang disediakan oleh Microsoft Visual Basic Program-program yang ditulis dengan Visual Basic juga dapat menggunakan Windows API, tapi membutuhkan deklarasi fungsi eksternal tambahan.

Dalam pemrograman untuk bisnis, Visual Basic memiliki pangsa pasar yang sangat luas. Dalam sebuah survey yang dilakukan pada tahun 2005, 62% pengembang perangkat lunak dilaporkan menggunakan berbagai bentuk Visual Basic, yang diikuti oleh C++, JavaScript, C#, dan Java.

2.4.2 Microsoft SQL Server 2000

Menurut artikel yang terdapat pada http://id.wikipedia.org/wiki/Microsoft_SQL_Server, Microsoft SQL Server adalah sebuah sistem manajemen basis data relasional (RDBMS) produk Microsoft. Bahasa kueri utamanya adalah Transact-SQL yang merupakan implementasi dari SQL standar ANSI/ISO yang digunakan oleh Microsoft dan Sybase. Umumnya SQL Server digunakan di dunia bisnis yang memiliki basis data berskala kecil sampai dengan menengah, tetapi kemudian berkembang dengan digunakannya SQL Server pada basis data besar.

Microsoft SQL Server dan Sybase/ASE dapat berkomunikasi lewat jaringan dengan menggunakan protokol TDS (*Tabular Data Stream*).

Selain dari itu, Microsoft SQL Server juga mendukung ODBC (*Open Database Connectivity*), dan mempunyai driver JDBC untuk bahasa pemrograman Java. Fitur yang lain dari SQL Server ini adalah kemampuannya untuk membuat basis data *mirroring dan clustering*. Pada versi sebelumnya, MS SQL Server 2000 terserang oleh cacing komputer SQL Slammer yang mengakibatkan kelambatan akses Internet pada tanggal 25 Januari 2003.