PENCARIAN ASOSIASI TOPIK DALAM AYAT AL-QUR'AN DENGAN MENERAPKAN ALGORITMA *MULTIPASS DIRECT HASHING AND PRUNNING* (M-DHP)

SKRIPSI

Sebagai salah satu syarat untuk memperoleh
gelar Sarjana dalam bidang Ilmu Komputer

Disusun Oleh :

ALFIAN ARDHI

NIM. 0810960001

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PERSETUJUAN

PENCARIAN ASOSIASI TOPIK DALAM AYAT AL-QUR'AN DENGAN MENERAPKAN ALGORITMA *MULTIPASS DIRECT HASHING AND PRUNNING* (M-DHP)

SKRIPSI

Sebagai salah satu syarat untuk memperoleh
gelar Sarjana dalam bidang Ilmu Komputer

Disusun oleh :

ALFIAN ARDHI
NIM. 0810960001

Telah diperiksa dan disetujui oleh:

Pembimbing I,	Pembimbing II,
Lailil Muflikhah, S.Kom., M.Sc	**Drs. Marji, M.T.**
NIP. 198002282006041001	**NIP. 196708011992031001**

LEMBAR PENGESAHAN

PENCARIAN ASOSIASI TOPIK DALAM AYAT AL-QUR'AN DENGAN MENERAPKAN ALGORITMA *MULTIPASS DIRECT HASHING AND PRUNNING* (M-DHP)

SKRIPSI

Sebagai salah satu syarat untuk memperoleh
gelar Sarjana dalam bidang Ilmu Komputer

Disusun Oleh:

ALFIAN ARDHI

NIM.0810960001

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 23 Januari 2013

Penguji I	Penguji II
Drs. Achmad Ridok, M.Kom	**Ir. Sutrisno, M.T.**
NIP. 196808251994031002	**NIP. 195703251987011001**

Penguji III

Rekyan Regasari MP, ST.,MT.
NIK. 77041406120253

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP. 196708011992031001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, Januari 2013

Mahasiswa,

Alfian Ardhi
NIM. 0810960001

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT yang telah melimpahkan segala Rahmat, Karunia dan Hidayah-Nya sehingga Penulis dapat menyelesaikan skripsi dengan judul: **"Pencarian Asosiasi Topik Dalam Ayat Al Qur'an Dengan Menerapkan Algoritma *Multipass Direct Hashing and Prunning* (M-DHP)".**

Skripsi ini diajukan sebagai syarat ujian seminar skripsi dalam rangka untuk memperoleh gelar Sarjana Komputer di Universitas Brawijaya. Atas terselesaikannya skripsi ini, Penulis mengucapkan terima kasih kepada:

1. Lailil Muflikhah, S.Kom, M.sc, selaku dosen pembimbing utama yang telah meluangkan waktu untuk memberikan pengarahan dan masukan bagi penulis.
2. Drs. Marji, M.T., selaku ketua program studi Ilmu Komputer dan pembimbing kedua yang telah banyak memberikan bimbingan serta bantuan.
3. Ir. Sutrisno, MT, selaku Ketua Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
4. Reza Andria Siregar, ST. selaku Dosen Penasehat Akademik.
5. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada Penulis selama menempuh pendidikan di Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
6. Segenap staf dan karyawan di Program Teknik Informatika Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya yang telah banyak membantu Penulis dalam pelaksanaan penyusunan skripsi ini.
7. Orang tua Penulis yaitu Bapak Adi dan Ibu Lilik Purwanti serta saudaraku Ahmad Ainul Yaqin atas segala dukungan materi dan doa restunya kepada Penulis.
8. Seluruh Civitas Akademika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.

9. Sahabat-sahabat saya Ardhy Wisdarianto, S.Kom, Maslikha Puspasari, S.Kom, Ardhiyan Syahrullah, S.Kom, M. Shohib Habibi S.Kom, Ikhwan Eko Setiawan, S.Kom, Nasrul Ahmad Hidayat S.Kom, M. Dedi Rudianto S.Kom, dan Faiz Al Huda S.Kom yang senantiasa memberikan semangat, doa dan ilmu-ilmunya selama ini demi terselesaikannya skripsi ini.
10. Teman-teman Ilmu Komputer angkatan 2008 tercinta yang telah banyak memberikan bantuan dan pengalaman selama menjadi mahasiswa di Universitas Brawijaya.
11. Teman-teman KKN Slumbung 2012 tercinta yang telah banyak memberikan semangat dan doa.
12. Semua pihak yang telah membantu terselesaikannya skripsi ini yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa skripsi ini masih banyak kekurangan dan jauh dari sempurna, karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Maka, saran dan kritik yang membangun dari semua pihak sangat diharapkan demi penyempurnaan selanjutnya. Semoga skripsi ini dapat bermanfaat dan berguna bagi semua pihak, baik penulis maupun pembaca, dan semoga Allah SWT meridhoi dan dicatat sebagai ibadah. Amin.

Malang, 28 Desember 2012

Penulis

PENCARIAN ASOSIASI TOPIK
DALAM AYAT AL-QUR'AN DENGAN MENERAPKAN
ALGORITMA *MULTIPASS DIRECT HASHING AND PRUNNING*

ABSTRAK

Seluruh umat islam di seluruh dunia dan tidak terkecuali di Indonesia pasti memiliki keinginan untuk lebih memperdalam pengetahuan yang terkandung di dalam Al Quran, dimana ketika seseorang memperdalam satu topik tertentu maka besar kemungkinan topik tersebut memiliki keterkaitan dengan topik-topik lain yang terdapat pada Al Qur'an. Oleh karena itu, pada penelitian ini dimaksudkan untuk membuat sistem pencarian asosiasi atau keterkaitan topik pada Al Qur'an. Untuk metode yang digunakan pada penelitian Association Rule Mining ini adalah *Multipass Direct Hashing and Prunning* (M-DHP), dimana metode ini untuk mengatasi karakteristik dari text database yang memerlukan ruang memory besar ketika menghitung frequent itemset. Proses pembangkitan rule pada M-DHP ini diperoleh melalui pengolahan data transaksi yang terbentuk dan kemudian menemukan *frequent 1 itemset*. Dari *frequent 1 itemset* ini dilakukan proses partisi yang digunakan untuk menemukan *frequent k itemset*. Selama proses pembentukan *frequent k itemset*, secara bersamaan juga melakukan reduksi database transaksi. Setelah seluruh frequent itemset terbentuk, maka tahapan selanjutnya merupakan pembangkitan rule yang digunakan untuk menemukan pola asosiasi topik pada al qur'an. Pengujian rule pada penelitian ini menggunakan rule-rule dengan tingkat akurasi yang kuat berdasarkan pada *conviction* dan *hiper lift ratio*, untuk partisi dipilih pada partisi 4. Pemilihan k-partisi yang tidak telalu besar memiliki keunggulan yaitu semakin banyak kombinasi rule yang dihasilkan. Hasil pengujian terbaik ketika *minimum support* 8% dan *minimum confidence* 90% dengan hasil 81,8%.

***Kata Kunci** : Association Rule Mining, M-DHP, Al-Qur'an, frequent itemset*

SEARCHING TOPIC ASSOCIATION
IN VERSE OF Al-QUR'AN WITH IMPLEMENT
MULTIPASS DIRECT HASHING AND PRUNNING ALGORITHM

ABSTRACT

All of Muslims in the world and atleast in Indonesia have a desire to further deepen the knowledge contained in the Qur'an, when one deepens one specific topic then likely to be related to the other topics contained in the Al-Qur 'an. Therefore, this study is intended to create an association of topics on the Qur'an. For the methods used in the study Association Rule Mining is Multipass Direct Hashing and prunning (M-DHP), which method to overcome the characteristics of text databases that require large memory space when calculating the frequent itemset. Rule generation process on M-DHP was obtained through the processing of transaction data formed first and then find frequent itemset. Of frequent itemset 1 is done partitioning process is used to find frequent itemset k. During the formation process of frequent itemset k, simultaneously also reducing transaction database. Once all frequent itemset is formed, then the next step is the generation rule is used to find patterns topic association of al-qur'an . Tests of rule in this study using a rule-rule with a strong degree of accuracy based on the conviction and hyper lift ratio, for the selected partition on partition 4. Choose of k-partition is not too big has the advantage that the more combinations of rules. The best test when the minimum support 8% and minimum confidence 90% with the result 81.8%.

***Keywords** : Association Rule Mining, M-DHP, Al-Qur'an, frequent itemset*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SOURCECODE

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Al Qur'an merupakan kitab suci agama Islam sehingga sebagian besar umat atau pemeluk agama Islam banyak yang memperdalam untuk mempelajari berbagai pengetahuan yang terkandung pada ayat-ayat Al Qur'an. Saat ini untuk lebih memperdalam Al Qur'an masyarakat dapat mempelajari dengan memanfaatkan Al Qur'an terjemahan baik dalam bentuk fisik maupun Al Qur'an digital. Pada Al Qur'an terjemahan baik yang dalam bentuk fisik ataupun digital telah dikelompokkan berdasarkan topik dari ayat tersebut, namun belum ada yang menampilkan suatu bentuk keterkaitan atau asosiasi antar topik, padahal berbagai topik yang terkandung di dalam Al Qur'an akan saling terkait satu sama lain.

Sering kali ketika mempelajari kandungan Al Qur'an hanya membaca suatu ayat yang berada dalam satu topik tertentu tanpa mencoba memahami ayat Al Qur'an lain yang topiknya juga berkaitan. Hal ini disebabkan oleh keterbatasan manusia baik dari segi ingatan dan pemahaman, sehingga pada penelitian ini mencoba membangun suatu sistem yang bertujuan untuk mengetahui atau mencari keterkaitan topik-topik yang terkandung dalam ayat dari surat-surat di dalam Al Qur'an. Yang perlu dipahami, pada dasarnya hasil dari penelitian ini diharapkan tidak digunakan sebagai acuan utama dalam mempelajari dan mendalami Al-Qur'an karena penelitian ini merupakan suatu kajian ilmiah yang sifatnya membantu untuk menunjukkan atau menampilkan keterkaitan topik pada Al-Qur'an.

Seperti yang telah dipaparkan sebelumnya bahwa ayat di dalam surat-surat Al Qur'an mengandung topik yang memiliki keterkaitan satu sama lain dan dalam satu surat juga memiliki pola-pola keterkaitan topik satu dengan topik yang lain, sehingga pola-pola ini dapat dibentuk menjadi suatu data transaksi. Adapun untuk topik-topik yang digunakan pada penelitian ini hanya terbatas pada pokok bahasan Ibadah.

Aturan asosiasi (*Association Rules*) atau juga bisa disebut analisis afinitas (*Affinity Analysis*) adalah analisa pertalian atau bagaimana mencari aturan untuk mencari keterkaitan antara dua atau lebih atribut dari suatu transaksi (*DISCOVERING KNOWLEDGE IN DATA An Introduction to Data Mining, 2005*). Adapun aturan ini juga disebut *"Market Based Analysis"* karena awalnya digunakan pada data transaksi pelanggan dan pada perkembanggannya telah diimplementasikan pada berbagai kasus yang berkaitan, oleh karena itu pada penelitian ini mencoba diimplementasikan untuk mengetahui keterkaitan topik di dalam Al Qur'an.

Apriori adalah algoritma pertama untuk aturan asosiasi yang diusulkan oleh Agrawal dan Srikant, dimana digunakan untuk menemukan kombinasi item dengan berdasar pada aturan tertentu kemudian diuji apakah kombinasi item tersebut memenuhi syarat *Support* Minimum *(Minimum Support)*. Kombinasi item yang memenuhi syarat *Support* minimum tersebut disebut *frequent itemeset* dimana nanti dipergunakan untuk membentuk aturan-aturan yang diharapkan memenuhi syarat minimum *Confidence (Fast Algorithms for Mining Association Rules, 1994).*

Algoritma Apriori memiliki kelemahan yaitu harus membangkitkan kandidat *Itemset* dalam jumlah besar, oleh karena itu algoritma Apriori dikembangkan agar lebih efisien dan salah satunya adalah algoritma DHP(*Direct Hashing and Prunning*). Algoritma DHP ini melakukan dua proses untuk menyelesaikan permasalahn tersebut. Pertama menggunakan *Hash Table* untuk mengurangi kandidat *itemset,* sedangkan proses kedua dengan melakukan proses reduksi pada database transaksi (*An effective Hash-Based for MiningAssociation Rules, 1995*).

Menurut John dkk (2000), kedua algoritma tersebut yaitu Apriori dan *Direct Hashing and Prunning* (DHP) tidak cocok untuk menemukan *frequent itemset* pada data transaksi yang terbentuk dari *text database*. Hal ini disebabkan untuk menghitung seluruh kejadian sehingga diperoleh jumlah *large* setiap *frequent itemset* dari kombinasi item (*itemset*) yang terbentuk memerlukan *space memory* yang cukup besar, oleh karena itu algoritma *Direct Hashing and*

Prunning (DHP) dikembangkan agar lebih efisien menjadi *Multipass Direct Hashing and Prunning* (M-DHP) *(John et all, 2000).*

Berdasarkan latar belakang yang telah dipaparkan di atas maka skripsi ini berjudul "Pencarian asosiasi topik dalam ayat Al Qur'an dengan menerapkan algoritma *Multipass Direct Hashing and Prunning* (M-DHP)*"*.

1.2 Perumusan Masalah

Dengan adanya latar belakang di atas, maka dapat dirumuskan permasalahan yang akan dijadikan objek penelitian ini, yaitu:

1. Bagaimana menerapkan algoritma M-DHP *(Multipass Direct Hashing and Prunning)* untuk menemukan keterkaitan atau asosiasi topik pada ayat di dalam Al Qur'an ?
2. Bagaimana pengaruh *Conviction* & *Hiper Lift Ratio* terhadap tingkat akurasi untuk menemukan katerkaitan topik yang terkandung dalam ayat dari surat-surat Al Qur'an dengan menerapkan algoritma *Multipass Direct Hashing and Prunning* (M-DHP) ?
3. Bagaimana pengaruh pembagian atau partisi *frequent itemset* terhadap tingkat kekuatan *rule* yang telah dihasilkan menggunakan *Conviction dan Hiper Lift Ratio* untuk menemukan katerkaitan topik yang terkandung dalam ayat dari surat-surat Al Qur'an dengan menerapkan algoritma *Multipass Direct Hashing and Prunning* (M-DHP) ?

1.3 Tujuan Penelitian

Tujuan yang ingin dicapai dari penelitian dan penulisan skripsi ini adalah:

1. Menerapkan algoritma *Multipass Direct Hashing and Prunning* (M-DHP) untuk menemukan keterkaitan atau asosiasi topik yang terkandung dalam ayat dari surat-surat Al Qur'an.
2. Mengetahui pengaruh *Conviction* & *Hiper Lift Ratio* terhadap tingkat akurasi untuk menemukan katerkaitan topik yang terkandung dalam ayat dari surat-surat Al Qur'an dengan menerapkan algoritma *Multipass Direct Hashing and Prunning* (M-DHP).

3. Mengetahui pengaruh pembagian atau partisi *frequent itemset* terhadap tingkat kekuatan *rule* yang telah dihasilkan menggunakan *Conviction dan Hiper Lift Ratio* untuk menemukan katerkaitan topik yang terkandung dalam ayat dari surat-surat Al Qur'an dengan menerapkan algoritma *Multipass Direct Hashing and Prunning* (M-DHP).

1.4 Batasan Permasalahan

Penelitian yang dilaksanakan dalam penulisan skripsi ini memiliki beberapa batasan sebagai berikut :

1. Algoritma yang digunakan adalah *Multipass Direct Hashing and Prunning* (M-DHP) sehingga hanya membahas topik seputar keterkaitan atau *Association Rule Mining* (ARM).
2. Al Qur'an yang digunakan sebagai sumber data adalah Al Qur'an *digital* yang diperoleh dari *http://www.alquran-digital.com*.
3. Topik-topik yang dicari keterkaitannya dalam skripsi ini hanya topik dalam pokok bahasan Ibadah bersuci, wudhu, mandi besar, tayammum, shalat, zakat, puasa, haji dan umrah, sumpah dan nazar, zikir, dan do'a, adapun daftar topik ditunjukkan secara lengkap pada Lampiran 1.
4. Data (*dataset*) yang dipergunakan dalam penelitian ini terbentuk dari topik-topik yang terkandung dalam ayat dari surat-surat pada Al Qur'an, dimana dari data 114 surat pada Al Qur'an *digital* tersebut diidentifikasi menurut topik-topik pada Batasan Masalah poin 2 sehingga menjadi sebuah data transaksi.

1.5 Manfaat Penelitian

Manfaat yang ingin dicapai dari penulisan skripsi ini adalah untuk menunjukkan atau menemukan pola-pola keterkaitan topik yang terkandung dalam ayat dari surat-surat pada Al Qur'an berdasarkan tingkat kekuatan *rule-rule* yang terbentuk dengan menerapkan Metode *Multipass Direct Hashing and Prunning* (M-DHP).

1.6 Sistematika Penulisan

Pembuatan tugas akhir ini dilakukan dengan sistematika penulisan sebagai berikut:

1. BAB I : PENDAHULUAN

   Bab ini berisi latar belakang penulisan, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan skripsi.

2. BAB II : TINJAUAN PUSTAKA

   Bab ini membahas tentang dasar teori yang terkait dengan topik penulisan skripsi yang diangkat dan menjadi acuan dasar dalam pembuatan sistem asosiasi topik pada ayat Al Qur'an ini.

3. BAB III : METODOLOGI DAN PERANCANGAN

   Pada bab ini menjelaskan metode-metode yang digunakan dalam menyelesaikan masalah, perancangan sistem, serta perancangan evaluasi.

4. BAB IV : HASIL DAN PEMBAHASAN

   Pada bab ini dijelaskan implementasi aplikasi, uji coba sistem, dan analisa hasil.

5. BAB V : PENUTUP

   Bab ini berisi kesimpulan yang diperoleh dari hasil pengujian dan saran untuk pengembangan lebih lanjut.

BAB II
TINJAUAN PUSTAKA

2.1 Al-Qur'an

Menurut khadiri (2005) di dalam bukunya yang berjudul "*Klasifikasi Kandungan Al-Qur'an*", setiap umat islam memahami dan menyadari bahwa Al Qur'an adalah kitab suci yang menjadi pedoman hidup. Al Qur'an bukan sekedar hanya mengatur hubungan manusia dengan tuhanya, namun juga mengatur hubungan manusia dengan manusia beserta dengan alam sekitarnya.

Banyaknya juz di dalam Al Qur'an yaitu 30 juz, dimana terdiri atas 114 surat dan tiap surat terdiri atas kumpulan ayat. Adapun surat terpanjang adalah Al Baqarah dengan 286 ayat dan yang terpendek adalah Al Kautsar, An-Nasr dan Al-'Aṣr yang masing-masing terdiri dari 3 ayat. Apabila berdasarkan dari tempat diturunkannya, dapat dibagi atas surat-surat Makkiyah atau surat Mekkah dan Madaniyah atau surat Madinah. Pembagian ini berdasarkan tempat dan waktu ketika turunnya surat, dengan ketentuan untuk surat-surat yang turun sebelum Rasulullah SAW hijrah ke Madinah digolongkan surat Makkiyah sedangkan setelahnya tergolong surat Madaniyah (Shiddieqy dan Hasbi, 2002).

Seperti yang telah dipaparkan sebelumnya, bahwa Al Qur'an berjumlah 30 juz dan terdiri dari 114 surat. Dari 114 surat tersebut terdiri dari kumpulan ayat-ayat Al Qur'an, yang mana ayat-ayat surat tersebut terbagi dalam topik-topik besar antara lain: *Al-Qur'an, Iman, Ilmu, Ibadah, Akhlak & adab, Hukum privat, Makanan & minuman, Pakaian & perhiasan, Muamalat, Peradilan & hakim, Hukum Pidan (Jinayah), Bangsa-bangsa terdahulu, sejarah dan jihad*. (Anonymous, 2004). Adapun pada penelitian ini hanya fokus pada pokok bahasan atau topik besar *Ibadah*, dimana dari pokok bahasan tersebut diperoleh topik-topik yang lebih terinci dalam hal *Ibadah* seperti yang ditunjukkan pada Lampiran 1.

2.2 Data Mining

2.2.1 Pengertian Data Mining

Data mining adalah proses menemukan hubungan atau keterkaitan yang memiliki arti ataupun pola dengan cara memeriksa kumpulan data dalam jumlah besar dan tersimpan di dalam media penyimpanan dengan menggunakan teknik pengenalan pola, seperti teknik statistik dan matematika(Larose, 2005).

Pada data mining terdapat teknik-teknik untuk pengolahan data, menurut Daniel Larose (2005) dalam Algoritma Data Mining (Kusrini dan Luthfi, 2009), jenis pekerjaan atau permasalahan yang dapat dipecahkan dengan teknik data mining yaitu:

1. Deskripsi

   Deskripsi merupakan cara untuk menggambarkan pola dan kecenderungan yang terdapat dalam data. Sebagai contoh, petugas pengumpulan suara tidak dapat menemukan keterangan atau fakta bahwa siapa yang tidak cukup profesional akan sedikit didukung dalam pemilihan presiden, namun dari deskripsi atau keterangan suatu pola data sering memberikan penjelasan untuk suatu pola atau kecenderungan.

2. Estimasi

   Estimasi hampir sama dengan klasifikasi, namun yang menjadi perbedaan mendasar adalah target estimasi lebih ke arah numerik daripada ke arah kategori. Contoh misalkan melakukan estimasi nilai indeks prestasi kumulatif mahasiswa pascasarjana dapat dilakukan dengan cara melakukan analisa atau melihat nilai indeks prestasi mahasiswa tersebut ketika mengikuti program pascasarjana.

3. Prediksi

   Prediksi hampir sama dengan klasifikasi dan estimasi, namun yang menjadi pembeda adalah hasil dari prediksi merupakan data yang mampu menggambarkan keadaan atau pola di masa mendatang. Beberapa metode dan teknik yang digunakan dalam klasifikasi dan estimasi dapat pula digunakan pada prediksi.

4. Klasifikasi

   Dalam klasifikasi telah memiliki target yang telah memiliki kategori. Sebagai contoh, klasifikasi dalam hal pendapatan dapat dipisahkan ke dalam tiga kategori, yaitu pendapatan tinggi, pendapatan sedang, dan pendapatan rendah.

5. Pengklusteran

   Kluster adalah kumpulan record yang memiliki kemiripan satu dengan yang lainnya dan memiliki ketidakmiripan dengan record-record dalam kluster lain. Pengklusteran berbeda dengan klasifikasi, hal mendasar yang membedakan antara *clustering* dan klasifikasi yaitu tidak adanya variabel target dalam pengklusteran.

6. Asosiasi

   Tugas asosiasi dalam data mining adalah menemukan atribut yang muncul bersama dalam satu waktu sehingga dianggap memiliki keterkaitan antar atribut. Sebagai contoh, menemukan barang didalam supermarket yang dibeli secara bersamaan dan barang yang tidak pernah dibeli secara bersamaan oleh konsumen.

2.2.2 Asosiasi

Menurut Kusrini (2009) di dalam bukunya "*Algoritma Data Mining*" analisis asosiasi atau *association rule mining* adalah teknik data mining untuk menemukan aturan asosiatif antara kombinasi item. Contohnya aturan asosiatif dari analisis pembelian di pasar swalayan sehingga dapat diketahui pola-pola keterkaitan item yang biasanya dibeli secara bersamaan oleh konsumen. Untuk algoritma pada asosiasi antara lain apriori, metode *Generalized Rule Induction* dan Algoritma *Hash Based*. Pada penelitian ini menggunaka metode asosiasi yang menggunakan struktur data *hash* atau Algortima *Hash Based*.

2.2.2.1 Itemset, Support,dan Confidence

Pada *Association Rule Mining* atau aturan asosiasi pada data mining mengenal *Itemset, Support & Confidence*. *Itemset* sendiri adalah suatu himpunan dari sekelompok item yang terbentuk oleh data transaksi. Adapun *Itemset* dengan

jumlah item *k* biasa disebut *k-Itemset*. *Support* (nilai penunjang) adalah persentase dari kombinasi item dari suatu transaksi database atau probabilitas pelanggan membeli beberapa produk secara bersamaan dari seluruh data transaksi (Kusrini dkk, 2009).

Support = $P(X \cap Y)$

$$P(X \cap Y) = \frac{jumla\ h\ transaksi\ \ mengandung\ \ \ x\ dan\ y}{total\ \ jumla\ h\ transaksi} \quad ...(2.1)$$

Confidence (nilai kepastian) merupakan kuatnya hubungan antar-item dalam aturan asosiasi atau probabilitas kejadian dari beberapa produk yang dibeli bersamaan, dimana salah satu produk sudah pasti terbeli.(Kusrini dkk, 2009). Adapun untuk *support* dan *confidence* memiliki nilai antara 0% sampai 100%.

Confidence = $P(X \mid Y)$

$$P(X \mid Y) = \frac{P(X \cap Y)}{P(X)} = \frac{jumla\ h\ transaksi\ \ mengandung\ \ \ x\ dan\ y}{jumla\ h\ transaksi\ \ \ yang\ \ mengandun\ \ g\ x} \quad ...(2.2)$$

2.2.2.2 Metodologi Dasar Association Rule Mining

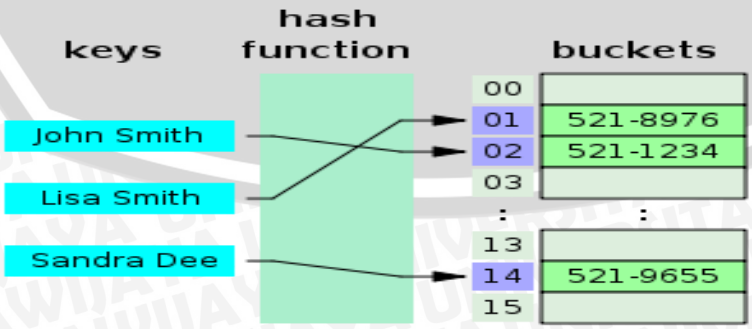
Menurut Agrawal dan Srikant (1994) di dalam jurnalnya yang berjudul *"Fast Algorithm for Mining Association Rules"*, seluruh permasalahan pada Association Rule dapat ditemukan solusinya apabila dibagi menjadi dua metodologi dasar, dimana hal ini dilakukan sebagai langkah dalam memecahkan permasalahn. Adapun langkah-langkah metodologi yang harus dilakukan yaitu:

1. Mencari seluruh *itemset* pada transaksi yang mempunyai nilai *support* di atas *minimum* support yang telah ditentukan. *Itemset* yang memiliki nilai *Support* di atas *minimum support* disebut sebagai *Large* Itemset.
2. Membentuk *rule* dari *Large Itemset* yang telah diperoleh dari proses-proses pencarian *Large Itemset* sebelumnya. Adapun *Large Itemset* adalah *Itemset* yang memiliki *Frequent Itemset*. Rule-rule yang terbentuk harus memenuhi nilai *Confidence* yang telah ditentukan sebelumnya, dimana hal ini bertujuan untuk memperoleh rule yang terpercaya.

2.3 Struktur Data Hash Table

Hash Table dan *Hash Tree* merupakan struktur data yang digunakan pada algortima M-DHP, dimana algoritma ini merupakan pengembangan dari algoritma DHP yang bertujuan agar lebih sesuai diterapkan pada data transaksi dari *text database*. Struktur data *Hash Table* disini digunakan sebagai pemfilter kandidat *itemset* (C_k), sedangkan *Hash Tree* digunakan untuk menyimpan dan menghitung jumlah *support* C_k yang digunakan untuk membentuk *Large Itemset* (L_k).

Menurut Goodrich dan Tamassia (2001) dalam bukunya yang berjudul *Data tructures & Algorithms in Java, Hash Table* adalah suatu struktur data yang mengasosiasikan antara *key* dan *value* atau nilai, dimana *key* biasanya merepresentasikan sebagai alamat dari suatu *value*. Adapun contohnya yang dapat dimisalkan seperti nilai *key* mewakili nama orang sedangkan alamat sebagai nilai *value*. Bagian utama dari *hash table* adalah *Bucket* atau *Bucket Array* dan *Hash Function*. *Bucket Array* merupakan suatu struktur array untuk *hash table* dimana masing-masing sel array tersebut biasa disebut *Bucket* yang didalamnya berisi kumpulan dari pasangan *key* dan *value*, sedangkan *hash function* sendiri adalah suatu fungsi yang digunakan untuk memetakan masing-masing *key* kedalam suatu nilai integer yang merepresentasikans sebagai nilai index pada *Bucket*. Berdasarakan penjelasan tersebut maka dapat menjelaskan mekanisme kerja dari *hash table* yaitu dengan memetakan nilai dari *key* dengan menggunakan *hash function* maka dihasilkan suatu nilai yang disebut nilai hash atau *hash value*. Nilai hash inilah merupakan suatu angka yang merepresentasikan indeks array untuk menentukan lokasi pada *bucket*.



Gambar 2. 1 Hash Table

Seperti pada gambar 2.1, yang disebut nilai *hash* adalah 00, 01, 02 dan selanjutnya, nilai hash ini diperoleh dari *hash function* yang merepresentasikan lokasi dari asosiasi antara *key* dan *value* pada *bucket*.

Metode *hash function* yang banyak digunakan adalah *Division Method*, dimana secara umum dapat dituliskan:

$$h(key) = key \bmod N \quad \text{....(2.3)}$$

Sesuai dengan yang dipaparkan pada persamaan (2.3), N adalah ukuran dari banyaknya ruang yang tersedia atau banyaknya *bucket* yang tersedia di dalam *hash table*. Namun secara *additional*, diperbolehkan untuk mengganti nilai N sebagai bilangan prima terdekat dari jumlah data. Adapun dalam memilih bilangan prima tersebut disarankan memilih bilangan yang lebih besar dari jumlah data yang ingin disimpan di dalam *hash table*, karena apabila N bukan bilangan prima akan banyak menimbulkan kemungkinan pola-pola distribusi nilai hash yang berulang dan ini menimbulkan *collisions*. Oleh karena itu dengan menerapkan *hash function* diharapkan dapat mengurangi kemungkinan terjadi *collisions* (Goodrich, 2001).

Menurut McMillan (2007) pada bukunya yang berjudul "*Data Structures and Algorithms Using C#*", bahwa pada struktur data *hash table* kemungkinan dapat terjadi *collisions* dan *hash function* hanya dapat mengurangi terjadinya *collisions*, sehingga masih diperlukan suatu metode untuk menangani masalah tersebut. Terdapat beberapa metode yang dapat digunakan, antara lain *Bucket Hashing, Open Addressing, Double Hashing* dan masih banyak yang lain. Adapun untuk menangani masalah tersebut maka pada penelitian ini menerapkan metode *Bucket hashing*, yaitu suatu struktur data yang sederhana atau *simple* dimana pada *hash table* dapat menyimpan banyak item, karena di dalam *bucket* tersebut berupa struktur data yang dinamis seperti *ArrayList* ataupun *List*.

2.4 Struktur Data Hash Tree

Hash Tree adalah suatu *tree* yang menerapkan teknik *hashing* dari struktur data *hash table*. Penerapan struktur data *hash tree* pada *Association Rule Mining (ARM)* adalah untuk menyimpan kandidat *k-itemset* (C_k) bersama dengan jumlah dari setiap C_k atau *counting candidate itemset*. Pada *hash tree*, untuk daftar dari

kandidat *k-itemset* (C_k) selalu berada pada *leaf node*, sedangkan *interior node* dapat dianggap sebagai representasi dari *hash table* yang merujuk ke setiap *node-node* dibawahnya. Terdapat dua hal yang perlu diperhatikan pada struktur *hash tree* ini yaitu *hash function* dan *maximum node size*. *Maximum node size* adalah ukuran data *maximum* pada setiap node atau lebih tepatnya pada *leaf node* dan apabila ukuran dari *leaf node* melebihi dari ukuran *maximum node size* maka akan membentuk cabang baru dan *leaf node* tersebut menjadi *interior node* (Agrawal dan Srikant, 1994). Adapun menurut Veeramalai dkk (2010) dalam jurnalnya yang berjudul *"Efficient Web Log Mining Using Enhanced Apriori Algorithm with Hash Tree and Fuzzy"* bahwa *root* dari *hash tree* tidak berisi data atau kosong dan untuk kedalaman (*depth*) dari *tree* sendiri adalah k atau dapat dikatakan sesuai dengan ukuran dari *kandidat itemset* yang dibentuk saat itu (*k-itemset*).

Menurut Bell dan Deen (1984) dalam jurnalnya yang berjudul *"Hash Trees Versus B-Trees"*, dikemukakan bahwa di *hash tree* untuk melakukan pengaksesan suatu data merupakan kombinasi antara *hashing* dan *indexing*. Hal ini juga sesuai dengan yang dipaparkan oleh Orlando dkk (2001) dalam jurnalnya yang berjudul *"Enhancing the Apriori algorithm for Frequent Set Counting"* bahwa *hash tree* menggunakan *hash function* untuk melakukan *insert* dan pencarian (*search*) nilai C_k secara langsung (*direct*). Adapun *hash function* yang digunakan untuk *hash tree*, seperti yang ditunjukkan pada persamaan (2.4).

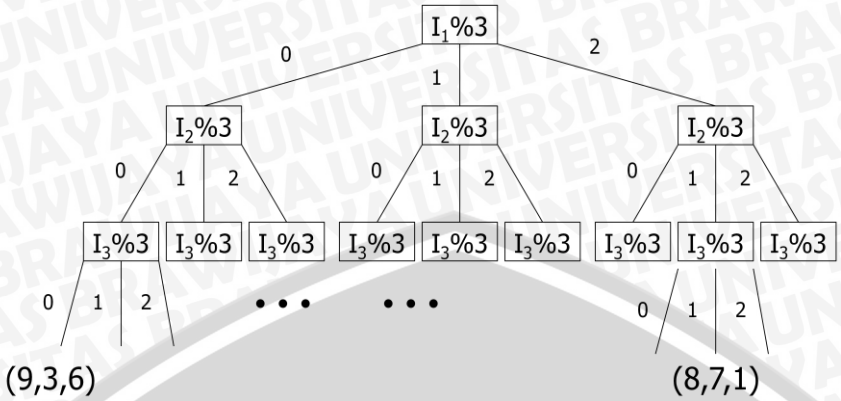
$$hash(I) = I \bmod H, H < m \qquad \text{.....(2.4)}$$

Dimana:

I : *item* ke-i dari *k-itemset*.

H : banyaknya pembagian cabang *tree*.

Dengan ketentuan bahwa nilai dari H kurang dari m, dimana m adalah jumlah dari *items*. Untuk lebih memperjelas proses dari *insert* pada *hash tree*, maka dapat dilihat pada gambar 2.2



Gambar 2. 2 Hash Tree

Sesuai dengan *hash tree* pada gambar 2.2 tersebut, maka akan melakukan *insert* data *3-itemset* yaitu {(9,3,6),(8,7,1)} dengan ketentuan menggunakan *hash function* h(I)= I mod 3. Pada penerapannya dimisalkan untuk kedalaman mulai 1 hingga 2 ukuran dari jumlah *itemset* pada *node* melebihi dari ukuran *maximum node size,* maka diperlukan pembentukan cabang baru sampai kedalaman 3 dan melakukan *insert* sampai ke kedalaman 3 dari *tree*. Dimisalkan proses *insert itemset* (9,3,6), adapun untuk proses pertama adalah perhitungan menggunakan *hash function* h(I)= I mod 3, maka pada *level*-1: I_1%3= 9%3= 0 artinya adalah *itemset* (9,3,6) *insert* pada cabang 0 dengan kedalaman 1, karena ukuran dari jumlah *itemset* pada node melebihi dari ukuran *maximum node size* maka memerlukan pembentukan cabang baru pada level 2. Pada *level*-2: I_2%3= 3%3= 0 artinya adalah *itemset* (9,3,6) *insert* pada cabang 0 dengan kedalaman 2, karena ukuran dari jumlah *itemset* pada node melebihi dari ukuran *maximum node size* maka memerlukan pembentukan cabang baru pada level 3, dan pada pada *level*-3: I_3%3= 6%3= 0 artinya adalah *itemset* (9,3,6) *insert* pada cabang 0 dengan kedalaman 3, karena ukuran dari jumlah *itemset* pada node tidak melebihi dari ukuran *maximum node size* maka pada kedalaman 3 tidak memerlukan pembentukan cabang baru pada level berikutnya dan untuk ukuran kedalaman pada *hash tree* adalah 3 karena merupakan *hash tree* untuk *3-itemset*.

2.5 DHP (Direct Hashing and Prunning)

DHP *(Direct Hashing and Prunning)* merupakan metode yang dikemukakan oleh Joon, Chen dan Philips, dimana tujuan dari pengembangan metode ini untuk mengatasi permasalahn klasik dari ARM *(Association Rules Mining)* yaitu mencari *frequent itemset*. Pengembangan metode klasik seperti *Apriori* sangat diperlukan, karena semakin lama data-data yang perlu dilakukan proses *mining* bukan bertambah sedikit, namun semakin banyak sehingga dengan adanya metode ini dapat meningkatkan efisiensi dalam proses ARM.

Menurut Joon dan Philips (1995) di dalam jurnalnya yang berjudul "*An Effective Hash-Based Algorithm for Mining Association Rules*", pada algoritma DHP ini terdapat 2 proses yang membedakan dengan metode ARM terdahulu seperti misalnya algoritma apriori. Pertama menggunakan *hash table* yang bertujuan untuk mengurangi kandidat *itemset* dan proses kedua yaitu untuk melakukan reduksi database transaksi. Disebabkan metode DHP ini menggunakan struktur data *hash table* maka dalam implementasinya juga menggunakan *hash function*. Pada implementasinya menururt Orlando dkk (2001) pada jurnal yang berjudul "*Enhancing the Apriori algorithm for Frequent Set Counting*" bentuk umum dari persamaan untuk *hash function* adalah sesuai dengan yang ditunjukkan pada persamaan (2.5).

$$hn(x1, \dots, xn) = \left(\sum_{i=1}^{n} xi \,.\, A^{k-(i-1)}\right) mod\ s \qquad \text{....(2.5)}$$

Dimana:

h_n : merupakan h_{k+1}

k : k- *itemset*

x_i : item ke-i dari tiap *itemset*

A : jumlah item

n : k+1

s : ukuran dari *bucket*

Secara umum metode DHP terbagi kedalam 3 bagian utama, dimana pada DHP bagian kedua dan ketiga memiliki sub-proses.(Joon dkk, 1995).

Berikut ini merupakan *pseudocode* metode DHP bagian pertama, kedua dan ketiga beserta dengan seluruh sub proses di dalamnya.

2.5.1 Algortima DHP bagian pertama

Pada tabel 2-1 merupakan *pseudocode* untuk menemukan kandidat *1-itemset*, frequent 1-*itemset* (L_1) dan membentuk *2-itemset* yang akan dilakukan penyimpanan pada *hash table* (H_2).

Tabel 2. 1 Pseudocode DHP bagian pertama

```
1  s = minimum support
2  set all the buckets of H2 to zero /* hash table */
3  forall transaction t ∈ D do begin
4  insert and count 1-items occurrences in a hash tree;
5  forall 2-subset x of t do
6  H2 [h2(x)]++;
7  end
8  L1 = {c|c.count ≥ s, c is in a leaf node of the hash tree};
```

Dimana:

s : *minimum support*.

H_2 : 2-*itemset* pada *hash table*.

h_2 : *hash function* (2.5).

t : database transaksi.

c : kandidat-*itemset*.

L_1 : *frequent 1-itemset*.

2.5.2 Algortima DHP bagian kedua

Tabel 2-2 merupakan *pseudocode* DHP bagian kedua yang terdiri dari prosedur *gen candidate*, *count support* dan *make hash*. Adapun algoritma ini berfungsi untuk menemukan kandidat *k-itemset*, membentuk database transaksi yang baru setelah dilakukan reduksi database transaksi, *k+1-itemset* yang disimpan pada struktur data *hash table* (H_{k+1}), dan frequent k-*itemset* (L_k).

Tabel 2. 2 Pseudocode DHP bagian kedua

1	k = 2;
2	D_k = D; /*database untuk *large* k-itemset*/
3	while (|{x|H_k[x] ≥ s} | ≥ Large){
4	/*Membuat sebuah *hash table**/
5	gen candidate (L_{k-1}, H_k, C_k);
6	set seluruh bucket dari H_{k+1} sama dengan kosong
7	D_{k+1} = ϕ;
8	forall transactions t ∈ D_k do begin
9	count support(t, C_k, k, $\hat{t}$); /* $\hat{t} \subseteq$ t*/
10	if (|$\hat{t}$| > k) then do begin
11	make hash ($\hat{t}$, H_k, k, H_{k+1}, $\tilde{t}$)
12	if (|$\tilde{t}$| > k) then D_{k+1} = D_{k+1} U {$\tilde{t}$}
13	end
14	end
15	L_k = { c ∈ C_k | c.Count ≥ s };
16	k++;
17	}

Dimana:

D_k : Database ke-k.

k : variabel *k-itemset*.

s : *minimum support*.

$|\{x|H_k[x] \geq s\}|$: Jumlah data *itemset* pada *hash bucket* dari *hash table* (H_k) yang memenuhi syarat *minimum support*.

Large : merupakan *threshold* untuk melakukan DHP bagian dua.

L_k : *frequent k-itemset*.

H_k : *hash table* untuk k-*itemset*.

C_k : kandidat k-*itemset*.

t : database transaksi.

$\hat{t}$: transaksi tereduksi hasil *count_support*.

$|\hat{t}|$: jumlah $\hat{t}$.

$\tilde{t}$: transaksi tereduksi hasil *make_hash*.

$|\tilde{t}|$: jumlah $\tilde{t}$.

2.5.2.1 Gen Candidate

Seperti yang ditunjukkan pada tabel 2-3 adalah *pseudocode* dari prosedur *gen candidate* yang berfungsi untuk menemukan kandidat *k-itemset* (C_k) dan dilakukan penyimpanan pada struktur data *hash tree*.

Tabel 2. 3 Pseudocode gen candidate

1	Procedure gen candidate (L_{k-1}, H_k, C_k)
2	$C_k = \phi$;
3	forall c=Cp [1] ... Cp[k-2]. Cp[k-1]. Cq[k-1],
4	Cp, Cq $\in$ L_{k-1}|Cp $\cap$ Cq| = k-2 do
5	if($H_k[h_k$ (c)] $\geq$ s) then
6	$C_k = C_k$ U {c}; /*insert c ke dalam *hash tree**/
7	end procedure

Dimana:

c : hasil join L_{k-1} dengan L_{k-1}.

L_{k-1} : *frequent* k-1-*itemset*.

H_k : *hash table* untuk k-*itemset*.

C_k : kandidat k-*itemset*.

h_k : *hash function* (2.5).

2.5.2.2 Count Support

Tabel 2-4 adalah *pseudocode* dari prosedur *count support* yang berfungsi untuk melakukan reduksi database transaksi sehingga item-item transaksi yang kemungkinan besar tidak berguna didalam proses dapat dihilangkan.

Tabel 2. 4 Pseudocode count support

1	Procedure count_support(t, C_k, k, $\hat{t}$)
2	forall c such that c $\in$ Ck and c (= ti1 ... tik) $\in$ t do
3	begin
4	c.count ++;
5	for (j=1; j$\leq$k; j++) a[ij]++;
6	end;
7	for(i=0; j=0; i<|t|; i++)
8	if(a[i]$\geq$k) then do begin $\hat{t}$j = ti; j++; end
9	end procedure

Dimana:

C_k : kandidat k-*itemset*.

t : database transaksi.

$\hat{t}$: transaksi tereduksi hasil *count_support*.

2.5.2.3 Make Hash

Berikut ini, pada tabel 2-5 adalah *pseudocode* dari prosedur *make hash* yang berfungsi untuk membentuk *k-itemset* yang disimpan pada struktur data *hash table* (H_{k+1}) dan pada prosedur ini juga melakukan reduksi database transaksi.

Tabel 2. 5 Pseudocode make hash

1	Procedure make_hast($\hat{t}$, H_k, k, H_{k+1}, $\ddot{t}$)
2	forall (k+1)-subsets x (= $\hat{t}$i1 ... $\hat{t}$ik) of $\hat{t}$ do
3	if (for all k-subsets y of x, $H_k[h_k(y)] \geq s$) then do
4	begin
5	$H_{k+1}[h_{k+1}(x)]$++;
6	for(j=1; j≤k+1; j++) a[ij]++;
7	end
8	for (i=0, j=0; i<| $\hat{t}$ |; i++)
9	if(a[i] > 0) then do begin $\ddot{t}$j= $\hat{t}$i; j++; end
10	end procedure

Dimana:

H_k : *hash table* untuk k-*itemset.*

$\hat{t}$: transaksi tereduksi hasil *count_support*.

$|\hat{t}|$: jumlah $\hat{t}$.

$\ddot{t}$: transaksi tereduksi hasil *make_hash*.

2.5.3 Algortima DHP bagian ketiga

Tabel 2-6 merupakan *pseudocode* DHP bagian ketiga. Pada algoritma DHP bagian ketiga ini hampir sama dengan DHP bagian kedua namun terdapat sedikit perbedaan yaitu tidak mebentuk *hash table* untuk *itemset* (H_k). Pada DHP bagian ketiga ini juga terdiri dari prosedur *gen candidate*, *count support* dan *apriori gen*. Adapun algoritma ini berfungsi untuk menemukan kandidat *k-itemset*, membentuk database transaksi yang baru setelah dilakukan reduksi database transaksi, dan frequent k-*itemset* (L_k).

Tabel 2. 6 Pseudocode DHP bagian ketiga

1	gen_candidate (L_{k-1}, H_k, C_k)
2	while(|C_k| >0) {
3	D_k = ϕ;
4	forall transactions t ∈ D_k do begin
5	count_support(t, C_k, k, $\hat{t}$)
6	if(| $\hat{t}$ | > k) then then D_{k+1} = D_{k+1} U { $\hat{t}$}

7	end
8	L_k = {c ∈ C_k|c.count ≥ s};
9	if(|D_{k+1}|=0) then break;
10	C_{k+1}=apriori_gen(L_k);
11	k++;
12	}

Dimana:

D_k : Database ke-k.

L_k : *frequent k-itemset.*

H_k : *hash table* untuk k-*itemset*.

C_k : kandidat k-*itemset*.

$\hat{t}$: transaksi tereduksi hasil *count_support*.

$|\hat{t}|$: jumlah $\hat{t}$.

2.5.3.1 Apriori Gen

Pada tabel 2-7 adalah *pseudocode* dari prosedur *apriori gen* yang berfungsi untuk membentuk kandidat *k+1-itemset* (C_{k+1}).

Tabel 2. 7 Pseudocode apriori gen

1	**join** L_{k-1} with L_{k-1}
2	insert into C_k
3	select p.item1, p.item2, ... , p.itemk-1, q.itemk-1
4	from L_{k-1} p, L_{k-1} q
5	where p.item1 = q.item1, ..., p.itemk-2 = q.itemk-1 < q.itemk-1;
6	**prune step**, delete all itemset c ∈ C_k that (k-1)-subset of c is not in L_{k-1}
7	forall itemset c ∈ C_k do
8	forall (k-1)-subsets s of c do
9	if (s ∉ L_{k-1}) then
10	delete c from C_k;

Dimana:

p, q : L_{k-1} atau *frequent* k-1 *itemset*.

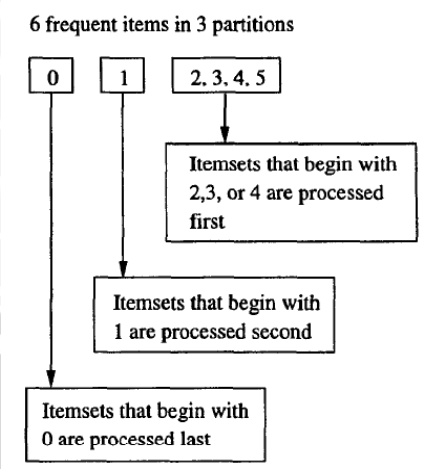
C_k : kandidat k-*itemset*.

2.6 M-DHP (*Multipass Direct Hashing and Prunning*)

Algoritma M-DHP *(Multipass Direct Hashing and Prunning)* dikembang-kan oleh John D. Holt dan Soon M. Chung. Algoritma M-DHP ini dikembangkan untuk mencoba mengatasi permasalahan *Association Rule Mining* (ARM) pada *text database*. Hal ini dikarenakan menurut John dan Soon (2000) di dalam jurnalnya yang berjudul "*Efficient Mining of Association Rules in Text Databases*" bahwa karakteristik dari *text database* sedikit memiliki perbedaan dari data transaksi perdagangan sehingga menyebabkan besarnya ruang memori yang diperlukan untuk menghitung *frequent itemset*, oleh karena itu algoritma mining untuk asosiasi yang telah ada misalnya Apriori ataupun DHP tidak cocok untuk mengatasi permasalahan *text database* tersebut.

M-DHP *(Multipass Direct Hashing and Prunning)* sendiri merupakan pengembangan dari DHP *(Direct Hashing and Prunning)* dimana metode ini mereduksi *frequent 1-itemset* (L_1) secara *partition*. Adapun secara alur dasar algoritma tersebut memang menerapkan algoritma dari DHP namun terdapat proses tambahan yaitu *partition* apabila telah diperoleh L_1. Berikut ini akan dijelaskan tahapan–tahapan dari algortima M-DHP yang sesuai dengan gambar 2.3:

1. Hitung seluruh kombinasi dari masing-masing item di dalam database transaksi dan temukan *frequent 1-itemset* (L_1).
2. Partisi *frequent 1-itemset* ke dalam p partisi P_1, P_2, ... , P_p.
3. Gunakan algoritma DHP untuk menemukan seluruh *frequent itemset* di dalam masing-masing partisi, dimana akan diproses secara urut mulai P_p, P_{p-1}, ... , P_1. Ketika P_p diproses kita dapat menemukan seluruh *frequent itemsets* yang terdapat pada P_p. Ketika partisi selanjutnya yaitu P_{p-1} juga diproses maka akan dapat menemukan seluruh itemset yang merupakan anggota dari item P_{p-1} dan P_p. Hal ini karena ketika P_{p-1} diproses, *itemset* yang ditemukan dari P_p diperluas dengan *itemset* dari P_{p-1} yang kemudian dihitung. Proses ini dilakukan sampai P_1.



Gambar 2. 3 Partisi k-itemset pada M-DHP

2.7 Rule Generate

Algortima M-DHP hanya akan menemukan *itemset* yang memenuhi dari *minimum support*, sedangkan untuk menemukan rule menggunakan algortima selain M-DHP. Untuk menemukan atau membentuk rule mengunakan algortima *Faster Algorithm*. Menggunakan algortima *Faster Algorithm* karena pada penelitian ini memerlukan *consequent* yang lebih dari satu item, sedangkan apabila menggunakan algoritma lain hanya menghasilkan *consequent* satu item.

2.7.1 Faster Algorithm

Tabel 2. 8 Pseudocode rule generate

1	// Faster Algorithm
2	forall large k-itemsets L_k, k ≥ 2 do begin
3	H_1 = {consequent of rules derived from L_k with one item in the consequent };
4	call ap-genrules(L_k, H_1);
5	End

Penjelasan dari *psedeucode* pada tabel 2-8 adalah akan dibentuk rule apabila nilai k dari k-*itemset* lebih dari sama dengan 2. Kemudian akan membentuk H_1 yang merupakan *consequent* dari L_k (*large k-itemset*) dengan hasil satu item dan selanjutnya memanggil prosedur *ap-genrules*.

2.7.2 Ap-Genrules

Tabel 2. 9 Pseudocode ap-genrules

1	procedure ap-genrules(L_k : large k-itemset, Hm: set of m-item consequents)
2	if(k > m+1) the begin
3	H_{m+1} = apriori-gen(H_m);
4	forall h_{m+1} ∈ H_{m+1} do begin
5	conf = support (L_k) / support (L_k - h_{m+1});
6	if (conf ≥ minconf) then
7	output the rule (L_k - h_{m+1}) => (h_{m+1}) with confidence = conf and support = support(L_k)
8	Else
9	delete h_{m+1} from H_{m+1};
10	End
11	call ap-genrules(L_k, H_{m+1});
12	End

Prosedur *apriori –gen* pada tabel 2-9 akan diproses apabila nilai k > m+1 dimana dengan inisialisasi m=1, kemudian mencari H_{m+1} atau h_{m+1} yang merupakan *itemset* untuk *consequent* hasil dari proses pada prosedur apriori-gen. Proses selanjutnya adalah melakukan pengecekan apakah conf (nilai *confidence*) ≥ minconf (*minimum confidence*), apabila memenuhi maka akan membentuk rule dan apabila tidak memenuhi maka h_{m+1} akan dihapus dari H_{m+1}. Proses dari prosedur ini akan berjalan secara rekursif hingga tidak memenuhi syarat k > m+1.

2.8 Measures

2.8.1 Conviction

Conviction adalah salah satu dari beberapa *measures* yang selama ini ada, dimana metode ini untuk mengatasi kelemahan dari *confidence* dan *lift*. Tidak seperti *lift*, *conviction* lebih akurat, adapun persamaan dari *conviction* seperti yang ditunjukkan pada persamaan (2.6):

$$conv(A \rightarrow B) = \frac{1-\sup(B)}{1-conf\ (A \rightarrow B)} \qquad (2.6)$$

Dimana:

A, B : *item.*

Support(B) : nilai *support* dari *item* B.

Conf(A → B) : nilai *confidence* aturan asosiatif $A \rightarrow B$.

Nilai range pada *conviction* ini, berada pada nilai 0.5,...,1,...∞ dengan ketentuan *conviction* dianggap memiliki nilai tak terhingga (*infinite*) apabila nilai dari $conf(A \rightarrow B)$ sama dengan 1. Sama seperti *lift*, apabila *conviction* menghasilkan nilai rule yang menjauh dari 1 maka akan dianggap akurat atau *rule* tersebut memiliki tingkat kekuatan yang baik (Azevedo dkk, 2006).

2.8.2 Hiper Lift

Terdapat *measures* lain yang digunakan untuk mengatasi kelemahan dari *lift ratio*, metode ini adalah *hiper lift*. Akurasi *hiper lift* merupakan pengembangan dari *lift ratio*, dimana persamaan *hiper lift* seperti yang ditunjukkan pada persamaan (2.7) berikut ini:

$$hyper\ lift\delta(X \rightarrow Y) = \frac{cXY}{Q\delta(Cxy)} = \frac{conf(X \rightarrow Y)}{\delta\left(support(Y)\right)}$$

$$= \frac{support\ (x \cup y)}{\delta(support\ \ (x)\ support\ (y))} \qquad (2.7)$$

Dimana:

δ : 0.99.

Conf$(X \rightarrow Y)$: nilai *confidence* aturan asosiatif $X \rightarrow Y$.

Support(Y) : nilai *support* dari *item* Y.

Pada *hiper lift* hampir sama seperti *lift*, yaitu apabila nilai *hiper lift ratio* dari suatu rule > 1 maka dianggap akurat atau tingkat kekuatan *rule* yang dihasilkan baik. Adapun mengapa pada *hiper lift* menggunakan nilai δ = 0.99 atau 99%, hal ini bertujuan agar hanya akan menghasilkan rule yang bersifat tidak terpercaya *(independent)* tidak lebih dari 1% dari setiap kasus (Hashler dkk, 2006).

2.9 Akurasi

Kualitas dari aturan asosiasi dapat dievaluasi menggunakan rumus akurasi. Akurasi dapat dihitung berdasarkan persentase *error* yang terjadi. Akurasi didefinisikan sebagai berikut :

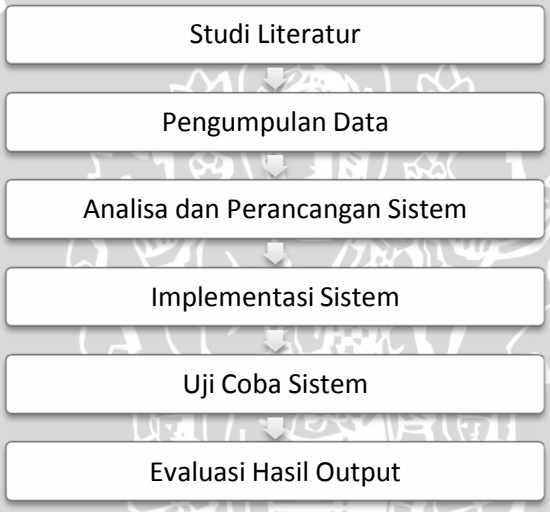
$$Akurasi = 100\% - Error \quad \text{.....(2.8)}$$

Dimana,

$$Error = (prediksi\ salah/total\ \ prediksi) * 100\% \quad \text{.....(2.9)}$$

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan sistem ini, akan membahas tahapan-tahapan yang dilakukan dalam pembuatan program "Pencarian asosiasi topik dalam ayat Al Qur'an dengan menerapkan algoritma *M-DHP (Multipass Direct Hashing and Prunning)*". Gambar 3.1 menunjukkan tahapan-tahapan tersebut secara umum.



Gambar 3. 1 Tahapan Metodologi dan Perancangan Sistem

Penjelasan dari langkah – langkah yang ditunjukkan pada Gambar 3.1 ini, adalah sebagai berikut:

1. Mencari dan mempelajari literatur-literatur yang berkaitan dengan metode *M-DHP (Multipass Direct Hashing and Prunning)*, dimana merupakan salah satu metode *Association Rule Mining* (ARM) dengan cara membaca buku, jurnal ataupun melakukan *browsing*.
2. Mengumpulkan data surat-ayat Al-Qur'an beserta topiknya dalam bentuk file **.txt* untuk digunakan sebagai data transaksi setelah melalui tahapan *preprocessing*.

3. Melakukan analisis dan perancangan sistem Pencarian asosiasi topik dalam ayat Al-Qur'an dengan menggunakan metode *M-DHP*.
4. Melakukan implementasi yaitu pembuatan program pencarian asosiasi topik dalam ayat Al-Qur'an menggunakan metode *M-DHP* sesuai dengan analisis yang telah dilakukan.
5. Melakukan ujicoba terhadap sistem yang telah dibuat dengan menggunakan data transaksi yang terbentuk.
6. Melakukan evaluasi rule yang dihasilkan oleh sistem.

3.1 Studi Literatur

Pada penelitian *Association Rules Mining* ini dibutuhkan suatu studi literatur, baik itu sumber materi yang berasal dari membaca buku ataupun jurnal. Hal ini dilakukan untuk menyelesaikan berbagai permasalahan yang muncul setelah melakukan analisa, sehingga diharapkan implementasi dari sistem ini dapat sesuai dengan tujuan yang diinginkan. Adapun mengenai materi ataupun teori yang digunakan pada penelitian ini seperti teori tentang DHP *(Direct Hashing and Prunning)*, *hash table*, *hash trees*, ataupun M-DHP *(Multipass Direct Hashing and Prunning)* diperoleh dari mempelajari berbagai jurnal, buku ataupun melakukan browsing di internet.

3.2 Pengumpulan Data

Data (*dataset*) yang dipergunakan dalam penelitian ini terbentuk dari 114 surat pada Al Qur'an yang kemudian setiap ayat surat diidentifikasi menurut topik-topik besar atau pokok bahasan: Al-Qur'an, Iman, Ilmu, Ibadah, Akhlak & adab, Hukum privat, Makanan & minuman, Pakaian & perhiasan, Muamalat, Peradilan & hakim, Hukum Pidana (Jinayah), Bangsa-bangsa terdahulu, sejarah dan jihad sehingga menjadi data transaksi yang sesuai dengan sistem, namun pada penelitian ini hanya menggunakan data dari pokok bahasan Ibadah. Adapun sumber dari Al-Qur'an digital yang digunakan berasal dari *http://www.alquran-digital.com*.

Data transaksi yang terbentuk merupakan data yang digunakan untuk menemukan pola-pola asosiasi, dimana bentuk dari data transaksi pada penelitian

ini terdiri dari TID dan *itemset*. Adapun TID merupakan representasi dari nomor surat Al Qur'an sedangkan untuk *itemset* merupakan berbagai topik yang terkandung di ayat-ayat pada setiap surat Al-Qur'an.

3.3 Analisa dan Perancangan Sistem

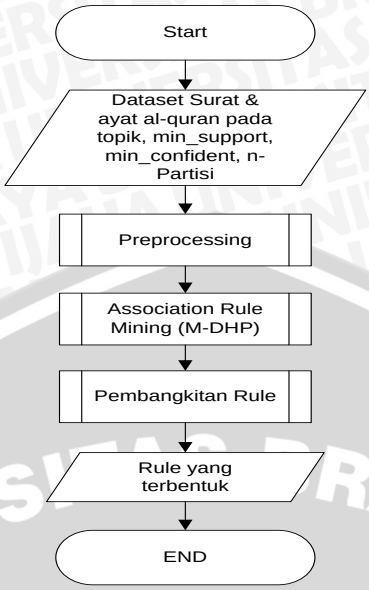
3.3.1 Deskripsi Sistem

Sistem yang akan dibuat merupakan program yang dapat menemukan asosiasi topik dari ayat-ayat surat dalam Al-Qur'an dengan menerapkan metode M-DHP *(Multipass Direct Hashing and Prunning)*. Dalam sistem ini nantinya akan menghasilkan *rule-rule* yang merepresentasikan keterkaitan atau asosiasi antar topik berdasarkan dari bentuk-bentuk transaksi pada setiap surat di dalam Al-Qur'an. Berdasar dari rule-rule tersebut maka dapat diketahui ayat-ayat mana saja yang juga terkait dimana sebagai pelengkap apabila dilihat dari segi informatif suatu sistem, namun dengan tetap berdasarkan rule-rule asosiasi topik yang dihasilkan.

Pada awal dari menjalankan sistem ini tentunya memerlukan berbagai inputan antar lain: input data transaksi, *nilai minimum support*, *minimum confidence*, dan jumlah partisi untuk *multipass*.

Dataset atau data transaksi pada sistem ini terdiri dari TID dan *itemset*, dimana TID merepresentasikan id transaksi yang berupa nomor surat dan *itemset* yeng berupa kumpulan topik-topik pada tiap surat yang terkandung di dalam Al-Qur'an. Adapun jumlah maksimal *record* pada database transaksi tentunya sama dengan jumlah surat di dalam Al-Qur'an yaitu berjumlah 114, sehingga jumlah transaksi pada data transaksi mksimal juga berjumlah 114 *record*.

Untuk menerapkan metode M-DHP pada sistem pencarian asosiasi topik dari ayat-ayat surat yang terkandung di dalam Al-Qur'an, terdapat beberapa langkah proses yang harus dilakukan agar menghasilkan suatu *rule* yang baik dan terpercaya. Alur pencarian asosiasi topik dari ayat-ayat surat pada Al-Qur'an secara umum dapat dilihat langkah-langkahnya pada gambar 3.2 dibawah ini



Gambar 3. 2 Flowchart sistem secara umum.

Sesuai diagram alur di atas, penjelasan dari *flowchart* pada gambar 3.2 adalah sebagai berikut :

1. Input Data dan Parameter Sistem

   Memasukkan dataset dokumen-dokumen topik (*.txt) yang berisi data surat-ayat Al-Qur'an yang dibutuhkan dalam proses sistem ini, dimana nantinya dari data tersebut akan terbentuk data transaksi. Adapun keterangan untuk data transaksi dapat dilihat pada sub-bab pengumpulan data. Untuk parameter lain dari sistem yang perlu diinputkan seperti *minimum support*, *minimum confident*, dan n-partisi.

2. *Preprocessing*

   Inputan dokumen-dokumen topik (*.txt) yang berisi data surat-ayat Al-Qur'an dibaca satu-persatu oleh *system* dan dilakukan *preprocessing* yang meliputi proses menemukan nomor topik berdasarkan inputan dokumen topik yang digunakan untuk membentuk data transaksi, *tokenizing*, dan penyusunan data transaksi.

3. Metode M-DHP

   Menerapkan metode *M-DHP (Multipass Direct Hashing and Prunning)* untuk mengolah data transaksi topik dari ayat-ayat di dalam surat Al-Qur'an sehingga menghasilkan *k-itemset* yang nantinya dibentuk menjadi *rule*.

4. Pembangkitan Rule

   Hasil dari proses M-DHP merupakan *k-itemset* yang pada tahapan ini dibentuk menjadi *rule-rule* yang baik dan terpercaya.

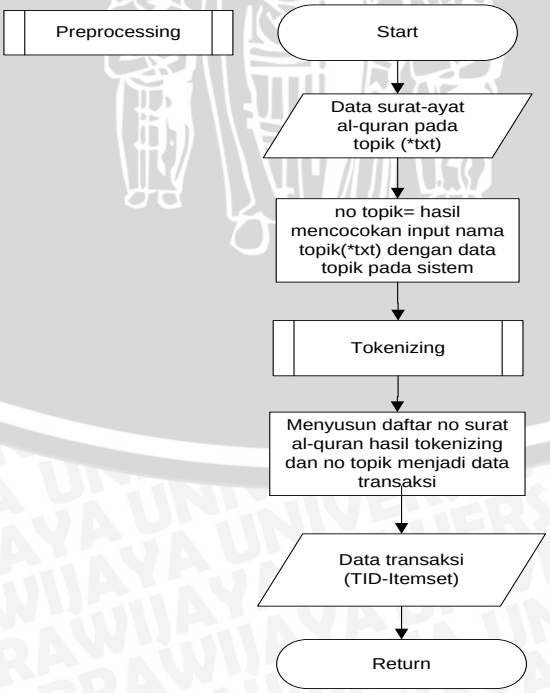
5. Output Data atau Hasil Uji

   Sistem ini menghasilkan keluaran berupa rule-rule keterkaitan topik pada ayat Al-Qur'an, dimana nantinya *rule-rule* tersebut dapat digunakan untuk mengetahui ayat-ayat mana saja yang saling terkait berdasarkan dari rule yang dihasilkan.

3.3.2 Perancangan Proses

Setelah melakukan analisis pada deskripsi sistem maka selanjutnya adalah melakukan perancangan terhadap proses sesuai dengan analisis yang telah dilakukan sebelumnya. Pada bagian ini akan menjelaskan secara rinci mengenai tahapan-tahapan dari proses yang akan diimplementasikan pada sistem asosiasi topik dari ayat-ayat surat dalam Al-Qur'an dengan menerapkan metode M-DHP *(Multipass Direct Hashing and Prunning)* ini.

3.3.2.1 Preprocessing



Gambar 3. 3 Flowchart Preprocessing

Pada *Preprocessing* dilakukan beberapa tahapan proses yang bertujuan untuk menyiapkan inputan dokumen topik (*.txt) yang berisi surat-ayat Al-Qur'an agar terbentuk menjadi data transaksi dan siap diolah di tahapan-tahapan selanjutnya pada metode *M-DHP (Multipass Direct Hashing and Prunning)*. Data Transaksi yang terbentuk dari tahapan ini sebagian digunakan sebagai data transaksi latih (data transaksi) dan sisanya digunakan sebagai data uji. Sesuai dengan gambar 3.3, maka tahapan *Preprocessing* ini meliputi:

1. Proses mencari atau menemukan nomor topik berdasar inputan dokumen-dokumen topik(*.txt). Adapun prosesnya yaitu dengan cara mencocokan antara inputan dokumen-dokumen topik (*.txt) dengan data topik pada sistem. Misal, diketahui input dokumen topik (*.txt) memiliki judul Akhlaq dan adab (Akhlaq dan adab.txt) yang berisi data "Menyampaikan amanat: 2:283, 3:75, 4:2, 4:58, 8:27, 23:8, 70:32". Selanjutnya dilakukan proses mencocokkan judul inputan dokumen tersebut dengan data pada sistem dan dimisalkan hasilnya diperoleh angka 2 (dua), maka nomor topik adalah 2.
2. T*okenizing.*

   Proses pada tahapan ini telah dijelaskan pada sub-bab **3.3.2.1.1 *Tokenizing***, dimana hasilnya adalah "2, 3, 4, 4, 8, 23, 70" dan daftar tersebut merupakan nomor surat.
3. Membentuk atau menyusun nomor topik *(Itemset)* dan output dari tahapan *Tokenizing* yang berupa daftar nomor surat *(TID)* menjadi suatu data transaksi yang sesuai dan dapat digunakan pada sistem ini. Maka hasil dari dua tahapan pada poin 1 dan 2 adalahseperti yang ditunjukkan pada table 3.1 sebagai berikut:

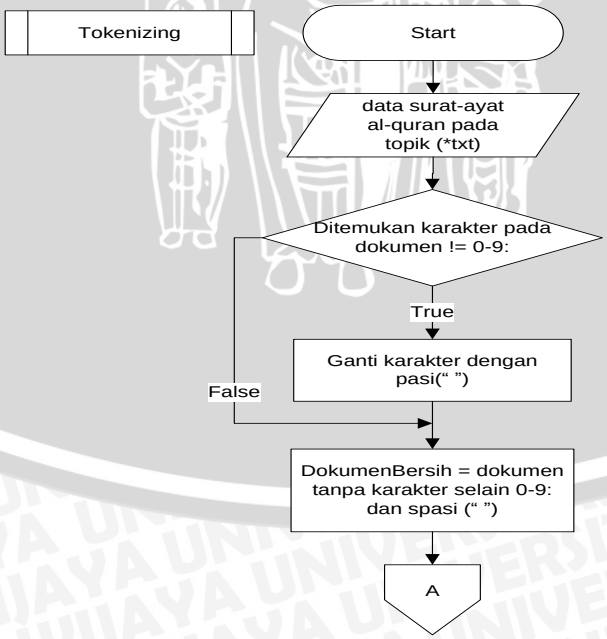
Tabel 3. 1 Data Transaksi (Contoh)

TID	***Itemset***
2	2,..
3	2,..
4	2,..
8	2,..
23	2,..
70	2,..

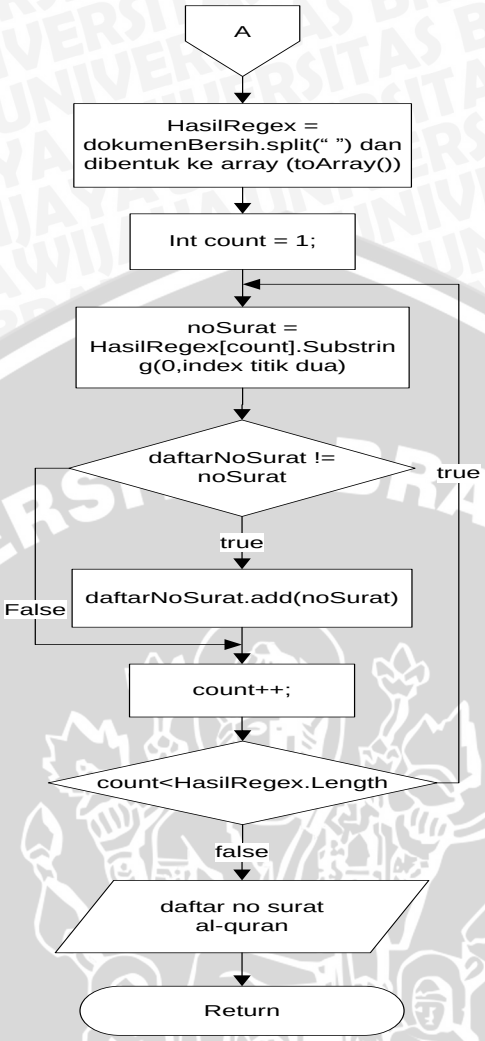
3.3.2.1.1 *Tokenizing*

Pada tahapan ini semua baris kalimat di dalam dokumen dilakukan pemilahan menjadi daftar nomor surat Al-Qur'an. Adapun berdasarkan flowchart proses *Tokenizing* yang dapat dilihat pada gambar 3.4, langkah-langkah penerapan dari proses *Tokenizing* adalah sebagai berikut:

1. Memasukkan (input) dokumen-dokumen topik (*.txt) yang berisi surat-ayat Al-Quran. Contohnya: "Menyampaikan amanat: 2:283, 3:75, 4:2, 4:58, 8:27, 23:8, 70:32"
2. Melakukan proses *scan* kata didalam kalimat-kalimat dokumen yang bertujuan untuk menghapus karakter selain angka yang bertanda khusus titik dua [0-9:], dan hasilnya menjadi "2:283, 3:75, 4:2, 4:58, 8:27, 23:8, 70:32" yang berupa nomor surat : ayat.
3. Pada proses selanjutnya yaitu tahapan dimana hanya mengambil nomor surat saja dengan menggunakan fungsi *Substring*, adapun hasilnya menjadi "2, 3, 4, 4, 8, 23, 70" dan daftar tersebut merupakan nomor surat.
4. Untuk output dari proses ini berupa daftar nomor surat Al-Qur'an, dimana nanti akan dibentuk menjadi suatu data transaksi.



Gambar 3. 4 Flowchart Tokenizing (1)



Gambar 3. 5 Flowchart Tokenizing (2)

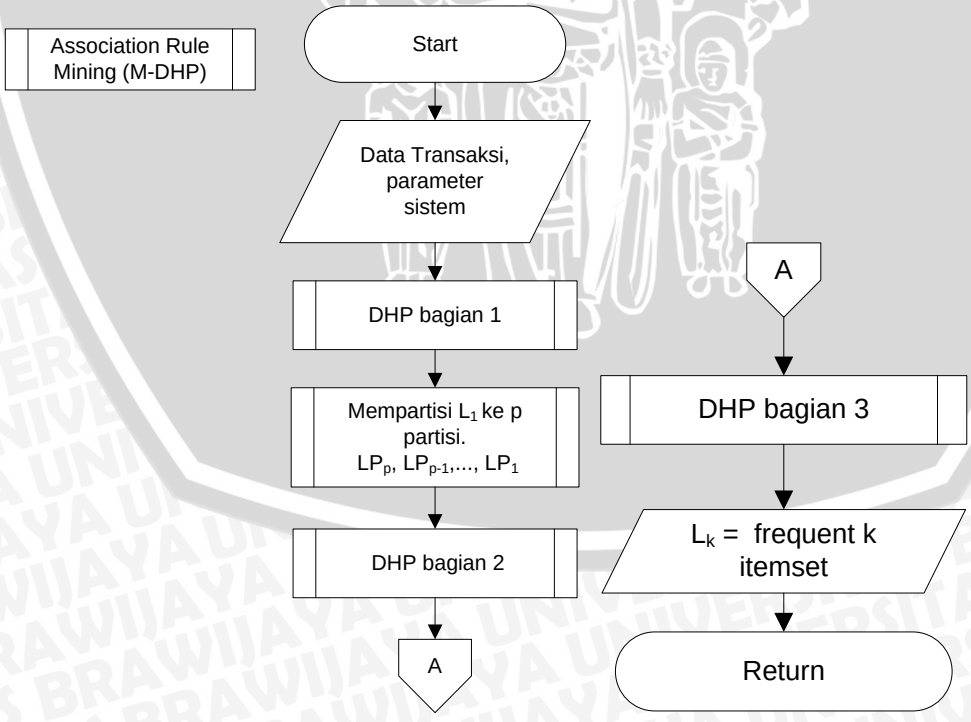
3.3.2.2 Metode M-DHP

Data masukan yang berupa data-data transaksi dan parameter yang dibutuhkan oleh sistem dimasukkan sebagai inputan, kemudian melakukan proses DHP bagian 1, Multipass, DHP bagian 2, dan DHP bagian 3. Adapun hasil dari proses tahapan ini adalah *frequent k-itemset* yang nantinya digunakan untuk melakukan proses pembangkitan *rule*. Berdasarkan dari gambar 3.6 di bawah ini, langkah-langkah dari penerapan metode M-DHP adalah sebagai berikut:

1. Memasukkan (input) dataset atau data transaksi dari hasil *preprocessing* dan parameter-parameter inputan yang dibutuhkan oleh sistem.
2. Melakukan proses DHP bagian pertama, dimana pada proses ini menghasilkan kandidat *1-itemset* (C_1) yang disimpan ke dalam *Hash Tree* dan menentukan

2-itemset untuk proses selanjutnya ke dalam *Hash Table* (H_2). Pada proses ini juga menghasilkan *frequent 1-itemset* (L_1).

3. Pada proses *partition* ini dilakukan partisi yang dimulai dari LP_p, LP_{p-1},..., LP_1.
4. Pada DHP bagian Kedua ini, terdapat 3 sub proses, yaitu proses mencari kandidat *itemset* (C_k) pada prosedur *Gen Candidate*, proses mereduksi database transaksi yang ditunjukkan pada prosedur *Count Support* dan terakhir proses membentuk *k-itemset* pada *hash table* untuk H_{k+1} dengan menggunakan prosedur *Make Hash*. Adapun hasil yang diperoleh pada DHP bagian kedua ini adalah *frequent k-itemset* (L_k).
5. Pada algoritma DHP bagian ketiga ini melakukan proses pencarian kandidat k-*itemset* (C_k) menggunakan prosedur *gen candidate* dan dilakukan juga reduksi database transaksi pada prosedur *count support*. Untuk menemukan kandidat *k+1 – itemset* pada proses ini menggunakan prosedur *apriori gen*. Pada DHP bagian ketiga ini juga menghasilkan *frequent k-itemset* (L_k).
6. Output dari tahapan ini adalah *frequent k-itemset* (L_k) yang nantinya digunakan untuk membentuk *rule* pada tahapan pembangkitan *rule.*



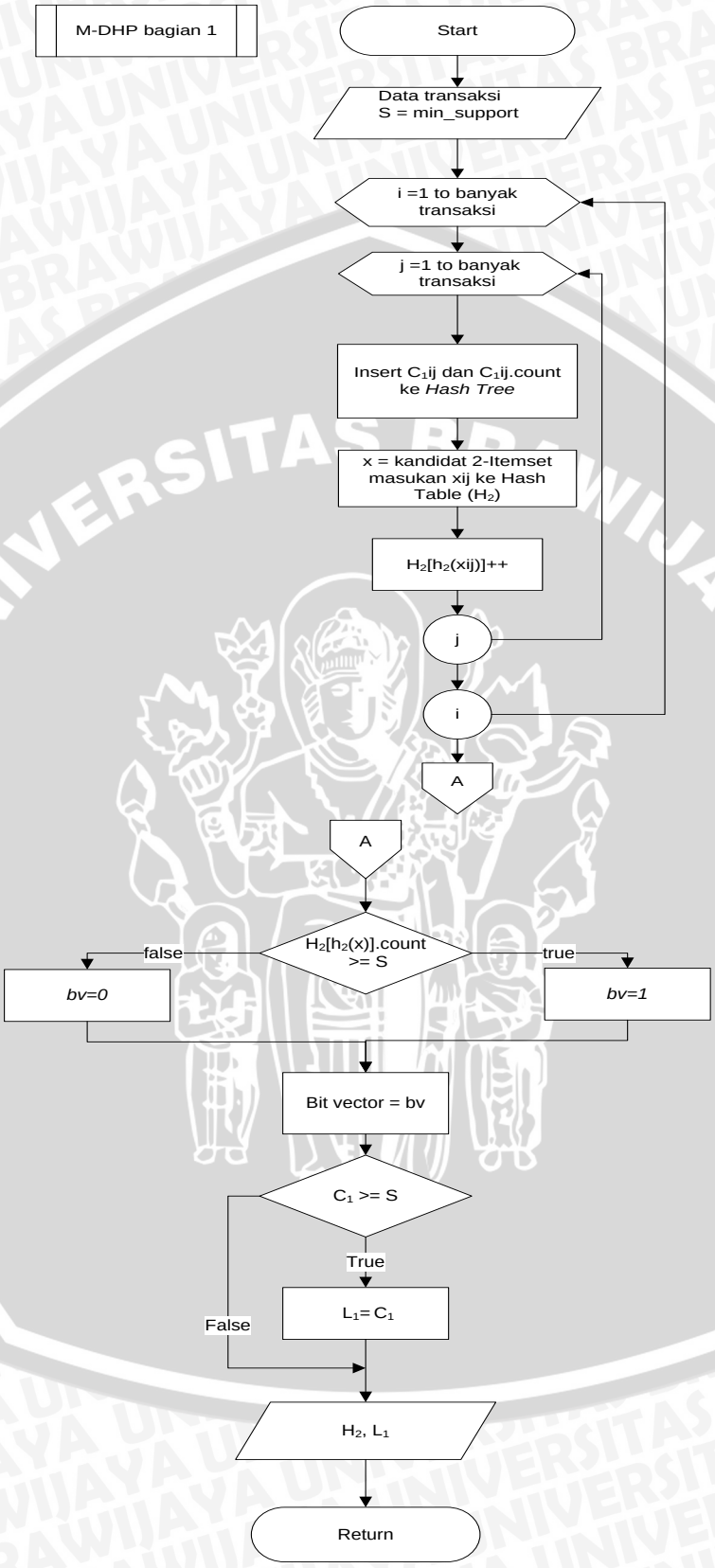
Gambar 3. 6 Flowchart metode M-DHP

3.3.2.2.1 DHP bagian Pertama

Pada DHP bagian Pertama merupakan proses yang mencari kandidat *1-itemset* (C_1) yang dihasilkan ke dalam *Hash Tree* dan menentukan 2-itemset untuk proses selanjutnya ke dalam Hash Table (H_2). Dalam melakukan *insert* atau memasukkan ke dalam *hash tree* terdapat dua hal yang harus diperhatikan yaitu menentukan *hash function* dan *max node* yang merupakan jumlah maksimum node pada *hash tree*. Pada tiap node selain berisi nilai dari *1-itemset* juga berisi nilai jumlah *support*. Secara bersamaan pada alur ini juga membentuk 2-itemset yang disimpan di adalah *hash table* (H_2). Dalam proses memasukkan itemset ke dalam *hash table* menggunakan *hash function* (2.5). Output dari proses pada alur ini berupa H_2 dan L_1, dimana L_1 merupakan C_1 yang memenuhi dari syarat *minimum support* yang telah di tentukan.

Berdasarkan dari gambar 3.6 di bawah ini, langkah-langkah dari penerapan metode DHP bagian pertama adalah sebagai berikut:

1. Input dataset atau data transaksi dan *minimum support* sebagai data masukan.
2. Melakukan *iterasi* sebanyak database transaksi latih untuk mencari kandidat *1-itemset* (C_1) beserta jumlah *1-itemset* yang ada, kemudian di-*insert* ke dalam *hash tree* dengan menggunakan *hash function* (2.4). Secara bersamaan juga membentuk *2-itemset* dan jumlah dari *2-itemset* yang ditemukan kemudian disimpan di dalam *hash table* (H_2), dimana untuk menyimpan di dalam *hash table* menggunakan *hash function* (2.5) atau $h_2(x)$.
3. Pada *bucket* H_2 terdapat jumlah dari tiap elemen, kemudian jumlah elemen tersebut akan diubah dengan syarat apabila memenuhi *minimum support* maka *bit vector* bernilai 1 dan apabila tidak memenuhi maka bernilai 0.
4. Untuk output dari proses ini adalah *2-itemset* yang tersimpan pada *hash table* (H_2) dan L_1, dimana ini diperoleh dari C_1 yang memenuhi syarat minimum dari *minimum support*.

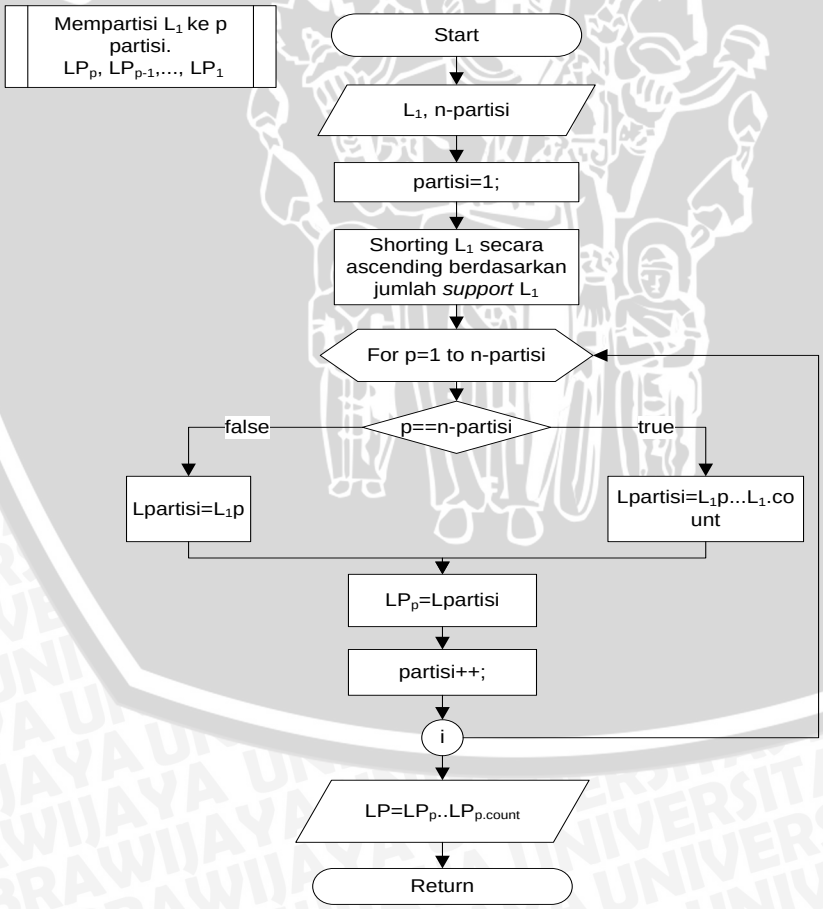


Gambar 3. 7 Flowchart metode DHP bagian Pertama.

3.3.2.2.2 Partisi

Proses *partition* dapat dikatakan sebagai salah satu proses penting pada sistem ini. Pada proses ini inputan yang diperlukan adalah *frequent 1 –itemset* (L_1) dan banyaknya jumlah partisi yang diinginkan (n-partisi). *Frequent 1-itemset* akan dipartisi sebanyak p *partitions* sesuai inputan n-partisi yang diharapkan dan bisa diilustrasikan banyak partisinya menjadi LP_p, LP_{p-1}, ... , LP_1 partisi. Adapun alur proses dari *partition* seperti yang ditunjukkan pada gambar 3.8. berikut ini:

1. Masukan (input) L_1 dan banyak partisi (n-partisi).
2. Data L_1 disorting dari nilai kecil ke nilai terbesar berdasarkan jumlah *support* dari L_1.
3. Melakukan iterasi sebanyak n-partisi dengan syarat jika i tidak sama dengan n-partisi maka hanya satu nilai yang diambil, namun apabila i sama dengan n-partisi akan diambil mulai index i sampai n data.



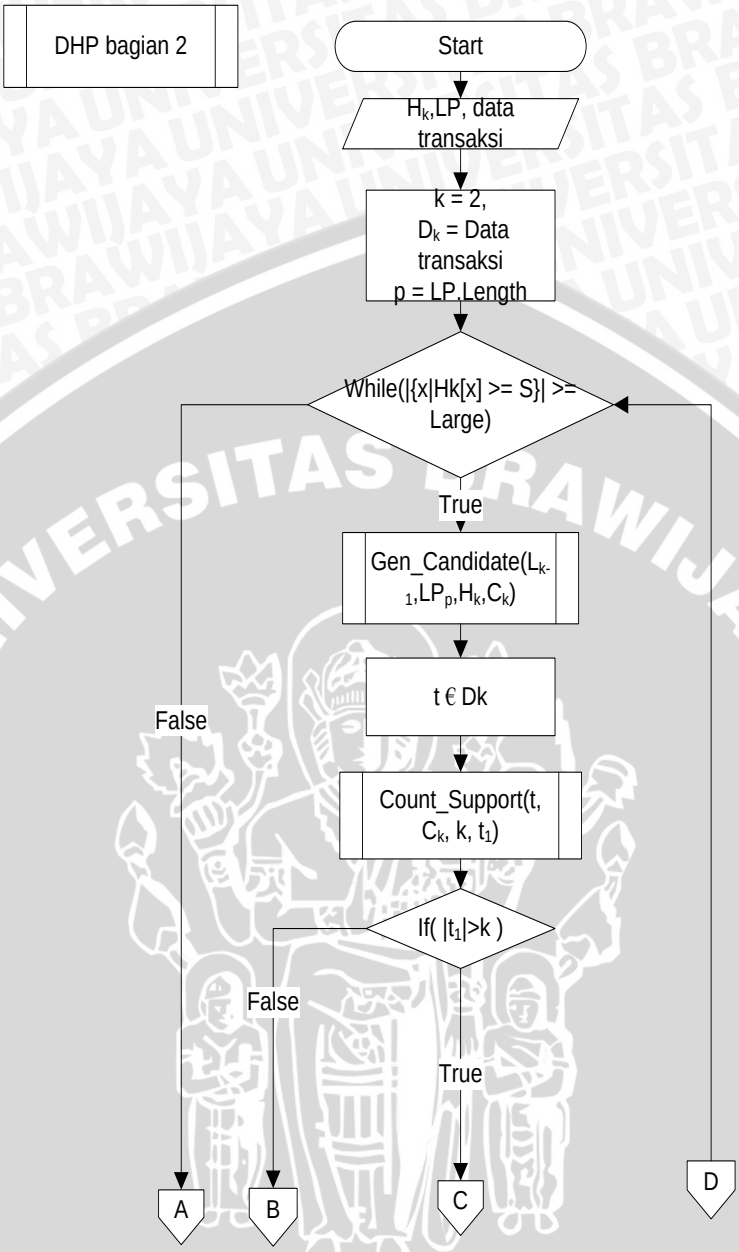
Gambar 3. 8 Flowchart Multipass

3.3.2.2.3 DHP bagian Kedua

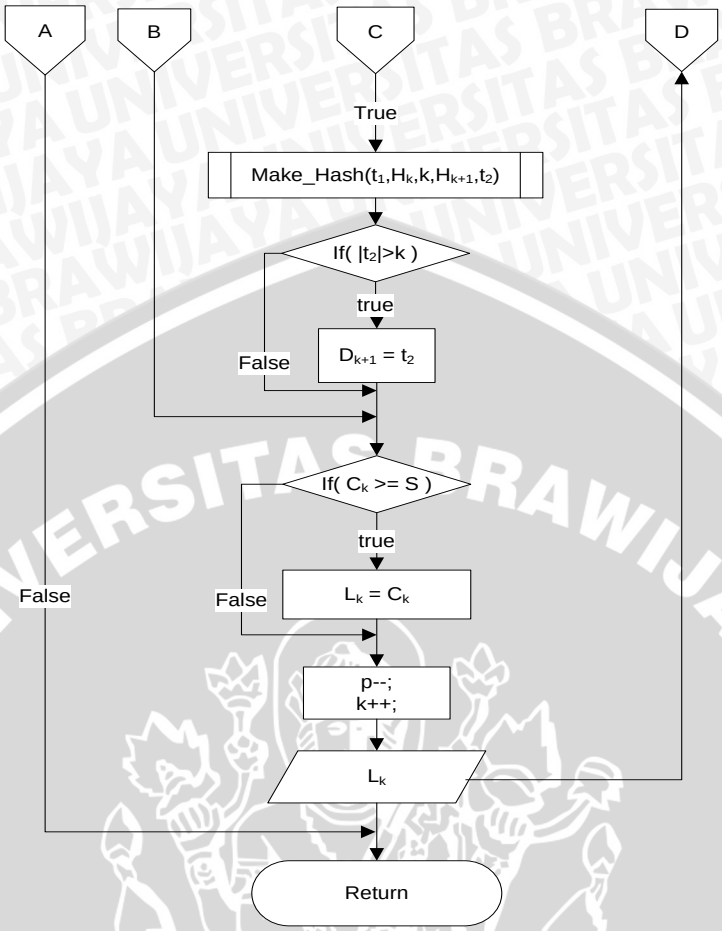
Pada DHP bagian kedua ini terdapat 3 sub proses, yaitu proses mencari kandidat *k-itemset* (C_k) pada prosedur *Gen Candidate*, proses mereduksi database transaksi yang ditunjukkan pada prosedur *Count Support* dan terakhir proses membentuk *itemset* pada *hash table* (H_{k+1}) dengan menggunakan prosedur *Make Hash*. Adapun hasil yang diperoleh pada DHP bagian kedua ini adalah *frequent k-itemset* (L_k).

Berdasarkan gambar 3.9 dan 3.10 dibawah ini, langkah-langkah dari penerapan metode DHP bagian kedua adalah sebagai berikut:

1. *Input* dari proses ini adalah H_k dari proses DHP bagian pertama, LP yang merupakan L_1 yang telah dilakukan proses *multipass* dan data transaksi. Untuk besar nilai *Large* sama dengan *k-itemset* saat itu. Selanjutnya dilakukan inisialisasi nilai k=2. Algoritma DHP bagian kedua ini akan terus berjalan apabila jumlah data pada *bucket hash table* (H_k) yang memenuhi syarat *minimum support* lebih dari sama dengan nilai *Large*.
2. Proses selanjutnya adalah pencarian kandidat *itemset* pada prosedur *Gen Candidate*.
3. Untuk langkah selanjutnya dilakukan proses reduksi database transaksi pada prosedur *Count Support* sehingga diperoleh data transaksi yang baru yaitu t_1. Untuk proses selanjutnya mengecek apakah jumlah data transaksi pada t_1 lebih dari nilai k, apabila memenuhi maka akan dilakukan proses pada prosedur *Make Hash* untuk memperoleh H_{k+1}.
4. Pada prosedur *Make Hash* selain membentuk H_{k+1} juga melakukan reduksi database data transaksi, sehingga diperoleh data transaksi tereduksi yang baru yaitu t_2. Proses selanjutnya membentuk D_{k+1} yang diperoleh dari data transaksi t_2, dimana harus memenuhi syarat untuk jumlah data transaksi t_2 lebih dari nilai k.
5. Output pada proses ini untuk memperoleh data L_k, oleh karena itu proses terakhir pada *flow* ini adalah menentukan anggota *frequent k-itemset* (L_k). Prosesnya yaitu dengan cara mengecek apakah setiap anggota C_k memenuhi syarat *minimum support* (S), apabila memenuhi syarat tersebut maka menjadi anggota L_k.



Gambar 3. 9 Flowchart metode DHP bagian Kedua (1)



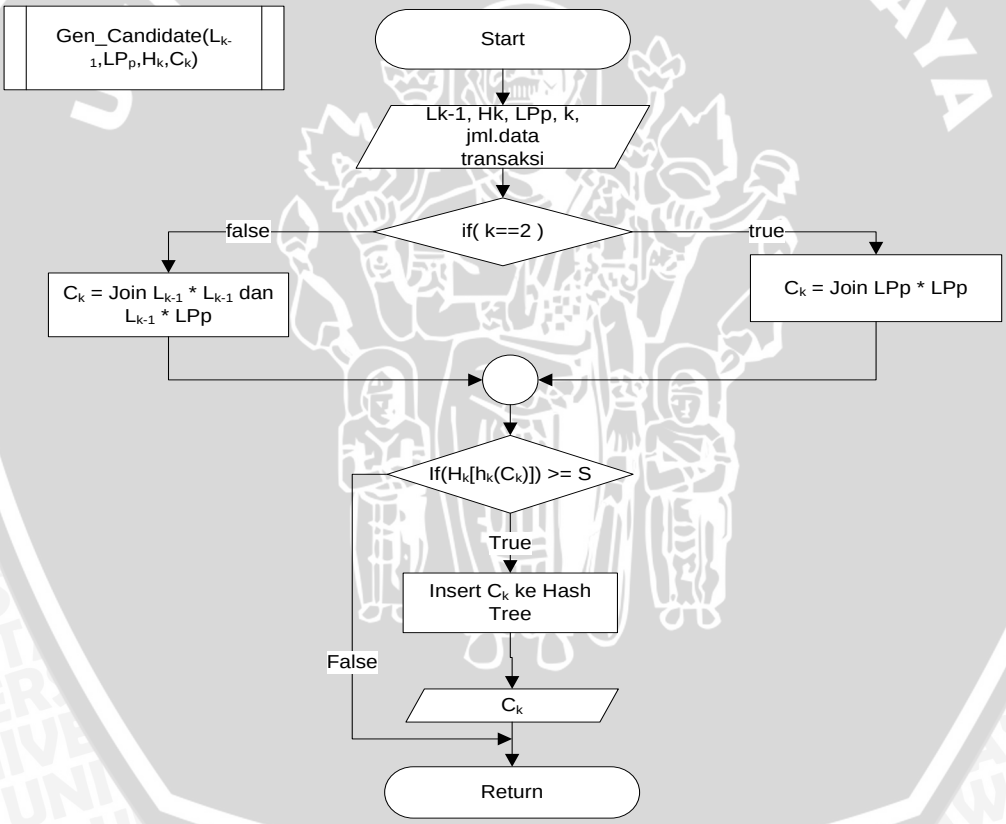
Gambar 3. 10 Flowchart metode DHP bagian Kedua (2)

3.3.2.2.3.1 Gen Candidate

Untuk memperoleh kandidat *k-itemset* pada algoritma M-DHP tetap menggunakan algortima dari DHP karena memang metode M-DHP merupakan pengembangan dari DHP oleh karena itu prosesnya secara garis besar hampir sama. Adapun sesuai dengan yang ditunjukkan pada gambar 3.11, langkah-langkah pada prosedur *gen candidate* adalah sebagai berikut:

1. Masukan terdiri dari L_{k-1}, H_k dan LP_p, k dan jumlah data transaksi. LP_p disini merupakan L_1 yang telah dilakukan proses *Multipass* atau telah dilakukan proses partisi. Proses selanjutnya yaitu *join* untuk membentuk kandidat *k-itemset*, adapun langkahnya dengan melakukan pengecekan apabila nilai k= 2 maka yang dilakukan join adalah LPp dengan LPp, namun apabila nilai k tidak sama dengan 2 maka dilakukan proses join L_{k-1} dengan L_{k-1} serta L_{k-1} dengan LP_p.

2. Pada proses ini output yang ingin dicari adalah C_k, oleh karena itu dilakukan suatu pengecekan dengan cara melihat setiap anggota C_k tersebut apakah telah di *hash* ke dalam *hash table* menggunakan *hash function* (2.5) dan memiliki *bit vector* sama dengan 1 atau memenuhi syarat *minimum support* yang telah ditentukan, apabila memenuhi maka C_k telah terbentuk dan selanjutnya melakukan *insert* ke dalam *hash tree* dengan menggunakan *hash function* (2.4). Bit vector pada *hash table* hanya sebagai penanda apabila bernilai 1 maka jumlah data pada *bucket* dari *hash table* memenuhi *minimum support* dan apabila bernilai 0 maka tidak memenuhi syarat *minimum support*.

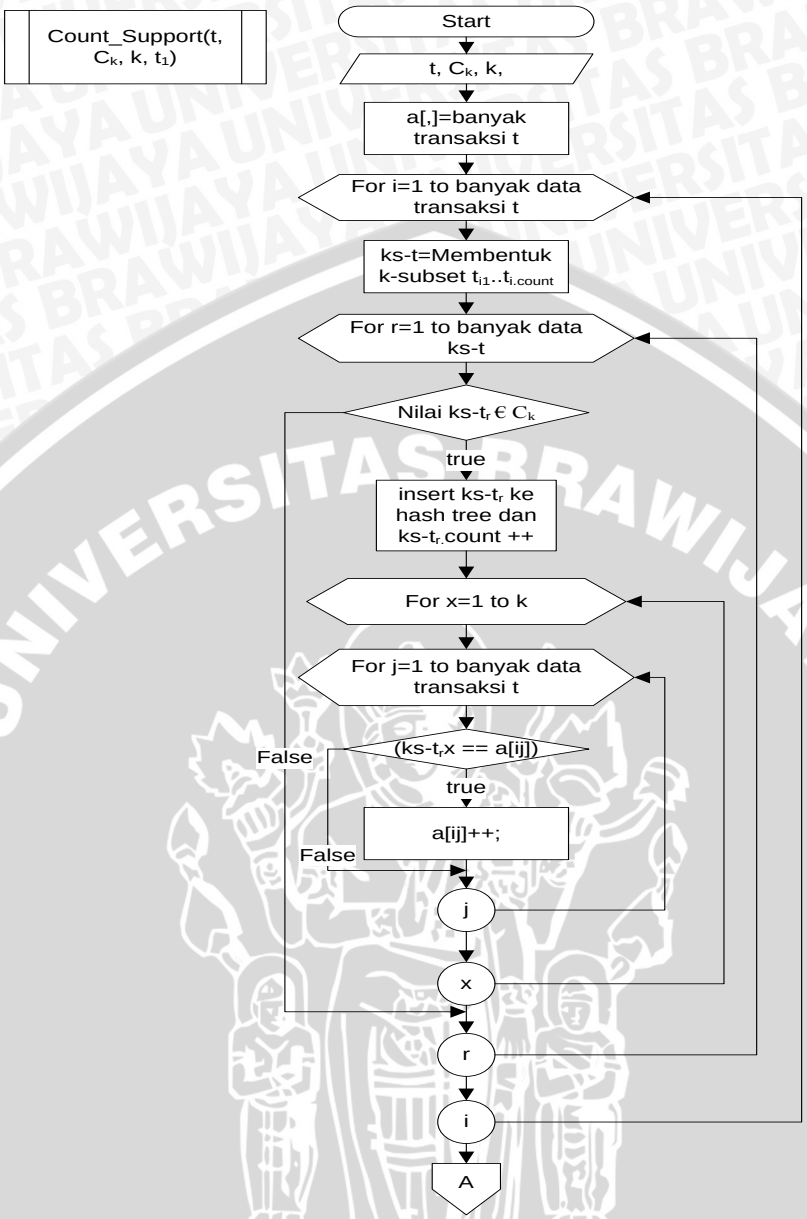


Gambar 3. 11 Flowchart Gen Candidate

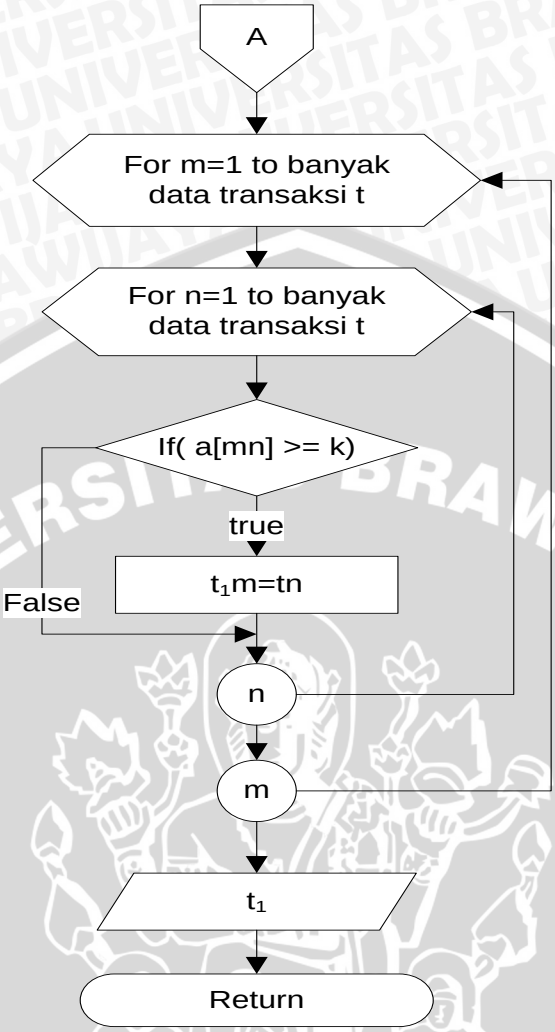
3.3.2.2.3.2 Count Support

Setelah memperoleh kandidat *k-itemset* (C_k) maka proses selanjutnya yaitu mereduksi database transaksi, dimana tahapan ini sesuai dengan alur pada DHP bagian kedua. Berdasarkan gambar 3.12 dibawah ini, langkah-langkah dari penerapan metode pada prosedur *count support* adalah sebagai berikut:

1. Pada proses ini masukan terdiri dari data transaksi (t), kandidat *k-itemset* (C_k) dan nilai k=2, serta *inisialisasi* array a.
2. Selanjutnya melakukan *looping* sebanyak data transaksi (t) dan membentuk *k-subset* dari data transaksi. Setelah *k-subset* terbentuk maka melakukan iterasi sebanyak *k-subset* (ks-t) dan melakukan pencarian ke dalam *hash tree* menggunakan *hash function* (2.4), apakah *k-subset* yang terbentuk sama dengan C_k di dalam *hash tree* dan apabila sama atau ditemukan maka *k-subset* tersebut di-*insert* ke dalam *hash tree* menggunakan *hash function* (2.4) dan sekaligus nilai *count supporting* C_k yang sama dengan *k-subset* bertambah satu.
3. Tahapan selanjutnya adalah melakukan iterasi sebanyak nilai k, kemudian mengecek apakah setiap *item* ke-x dari *k-subset* yang telah di-*insert* pada *hash tree* (ks-t_rx) sama dengan *item* a[ij] dari data transaksi, apabila benar maka item a[ij] yang merepresentasikan data transaksi tersebut akan bertambah satu jumlahnya.
4. Setelah seluruh item pada data transaksi yang diwakili oleh array a[ij] telah diketahui jumlahnya, maka proses selanjutnya adalah pereduksian database transaksi. Proses reduksi database transaksi dilakukan dengan cara melakukan iterasi sebanyak database transaksi, kemudian mengecek jumlah dari tiap item a[mn] data transaksi yang kurang dari nilai k maka akan dilakukan reduksi. Adapun hasil atau output dari proses *count support* ini akan diperoleh database transaksi yang baru (t_1) yang merupakan hasil reduksi database data transaksi setelah tahapan DHP bagian pertama(t).



Gambar 3. 12 Flowchart Count Support (1)



Gambar 3. 13 Flowchart Count Support (2)

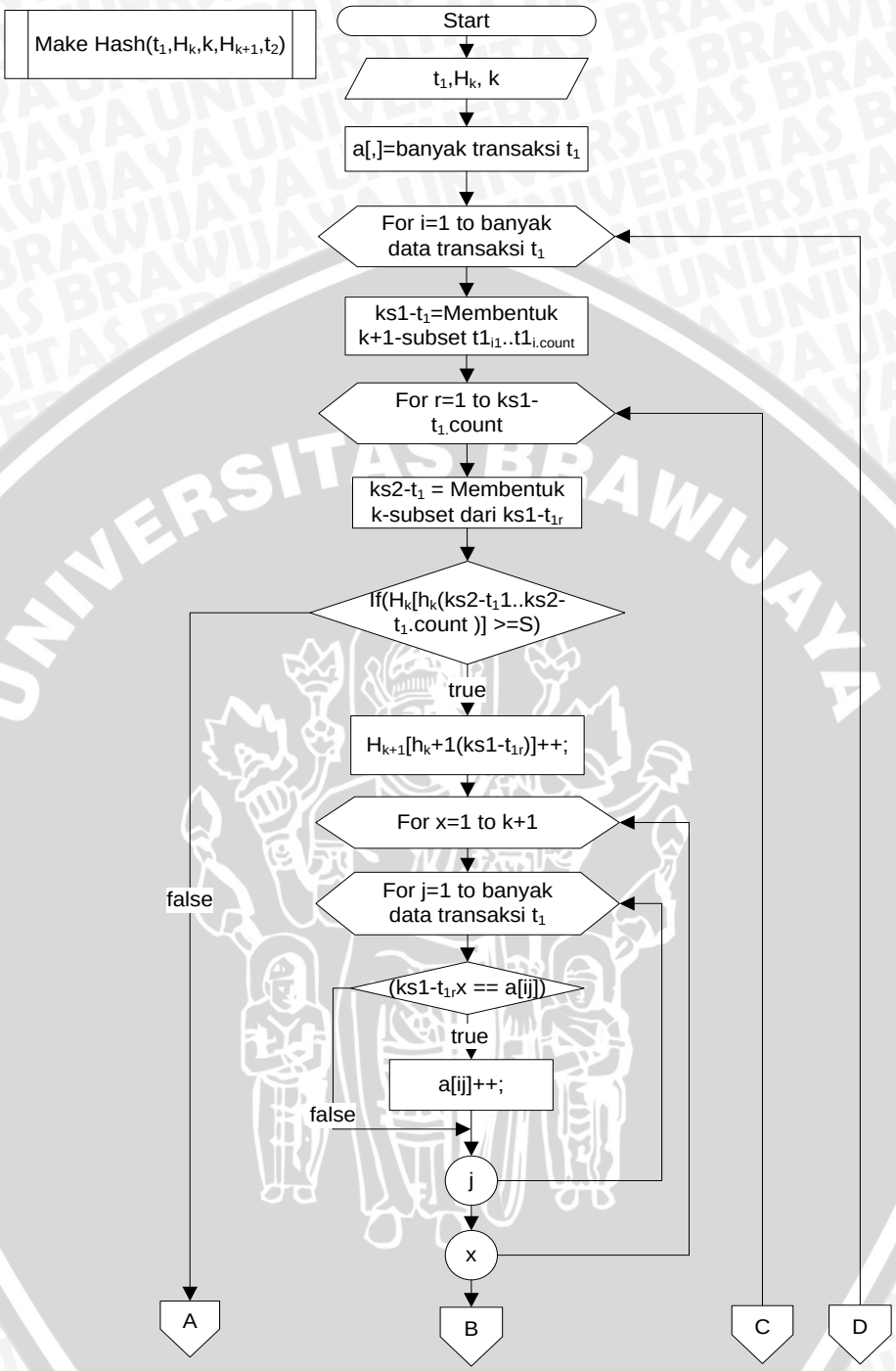
3.3.2.2.3.3 Make Hash

Tahapan terakhir pada DHP bagian kedua adalah Make Hash, dimana pada prosedur ini membentuk H_{k+1} untuk proses-proses selanjutnya. Berdasarkan gambar 3.14 dan 3.15 dibawah ini, langkah-langkah dari penerapan prosedur *make hash* adalah sebagai berikut:

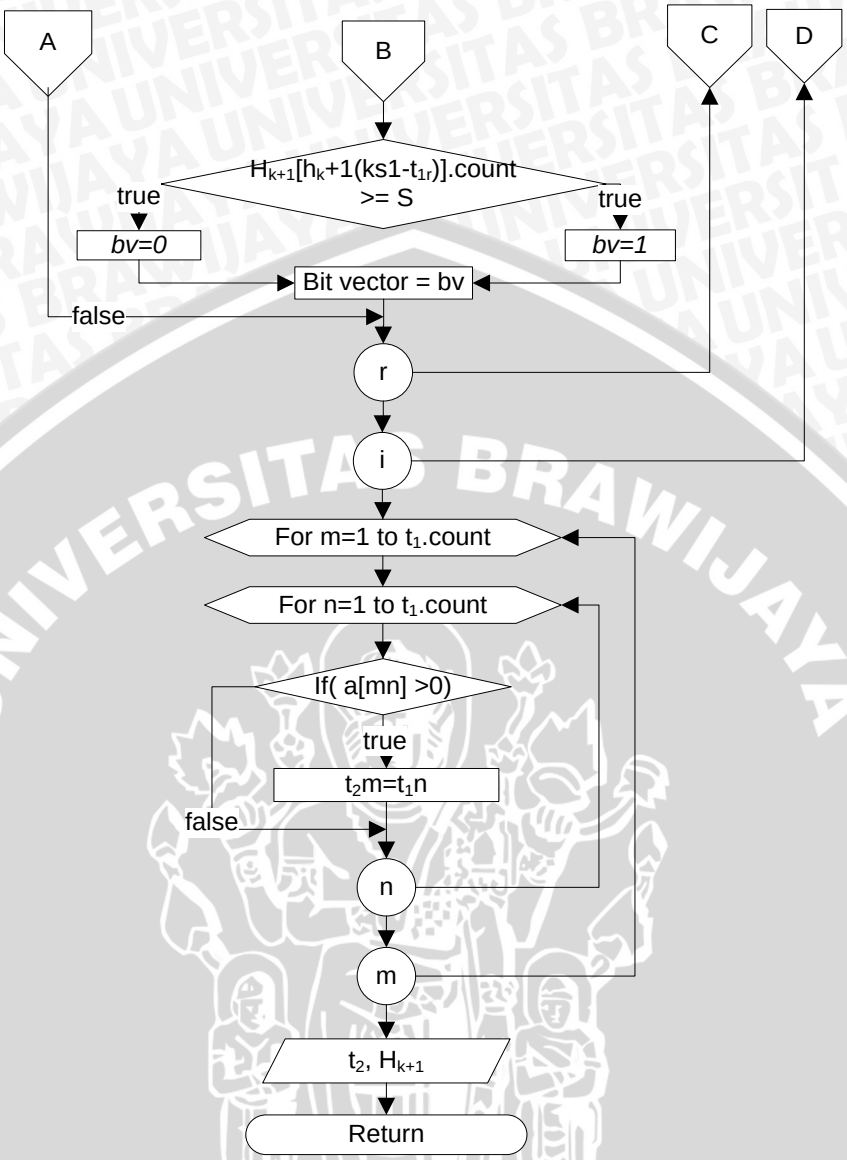
1. Masukan dari proses *make hash* ini terdiri dari H_k, k dan t_1 atau database transaksi latih yang telah tereduksi pada *count support*, serta serta *inisialisasi* array a.
2. Selanjutnya dilakukan iterasi sebanyak database data transaksi t_1 dan membentuk *k+1-subset* (ks1-t_1) dari t_1, kemudian dari setiap *k+1-subset* yang terbentuk diubah menjadi k-*subset*. Dari *k-subset* yang terbentuk dicek

apakah k-*subset* tersebut terdapat pada H_k dan apakah nilai *bit vector*-nya sama dengan 1 atau memenuhi syarat *minimum support* menggunakan *hast function* (2.5), apabila benar maka *k+1-subset* yang terbentuk disimpan pada *hash table* H_{k+1} menggunakan *hast function* (2.5), secara bersamaan juga mencari berapa jumlah data pada H_{k+1} tersebut.

3. Tahapan selanjutnya adalah melakukan iterasi sebanyak nilai k+1, kemudian mengecek apakah setiap *item* ke-x dari *k+1-subset* yang telah disimpan pada *hash table* (ks1-t_{1r}x) sama dengan *item* a[ij] dari data transaksi, apabila benar maka item a[ij] yang merepresentasikan data transaksi tersebut akan bertambah satu jumlahnya. Pada tahap ini juga melakukan pengecekan apabila jumlah data dari *k+1-subset* yang berada pada H_{k+1} melebihi *minimum suppport* maka bit vector sama dengan 1 apabila sebaliknya maka bit vector bernilai 0.
4. Selain proses membentuk H_{k+1}, pada *make hash* ini juga melakukan reduksi database data transaksi sehingga diperoleh database transaksi yang baru (t_2). Proses reduksi database data transaksi pada prosedur *make hash* sedikit berbeda dengan reduksi pada *count support*, adapun letak perbedaannya yaitu data transaksi t_1 akan tereduksi apabila jumlah item dari tiap data transaksi t_1 kurang dari 0. Dengan kata lain minimal item dari data transaksi $t_1$berjumlah 1 saja tidak akan tereduksi.



Gambar 3. 14 Flowchart Make Hash (1)

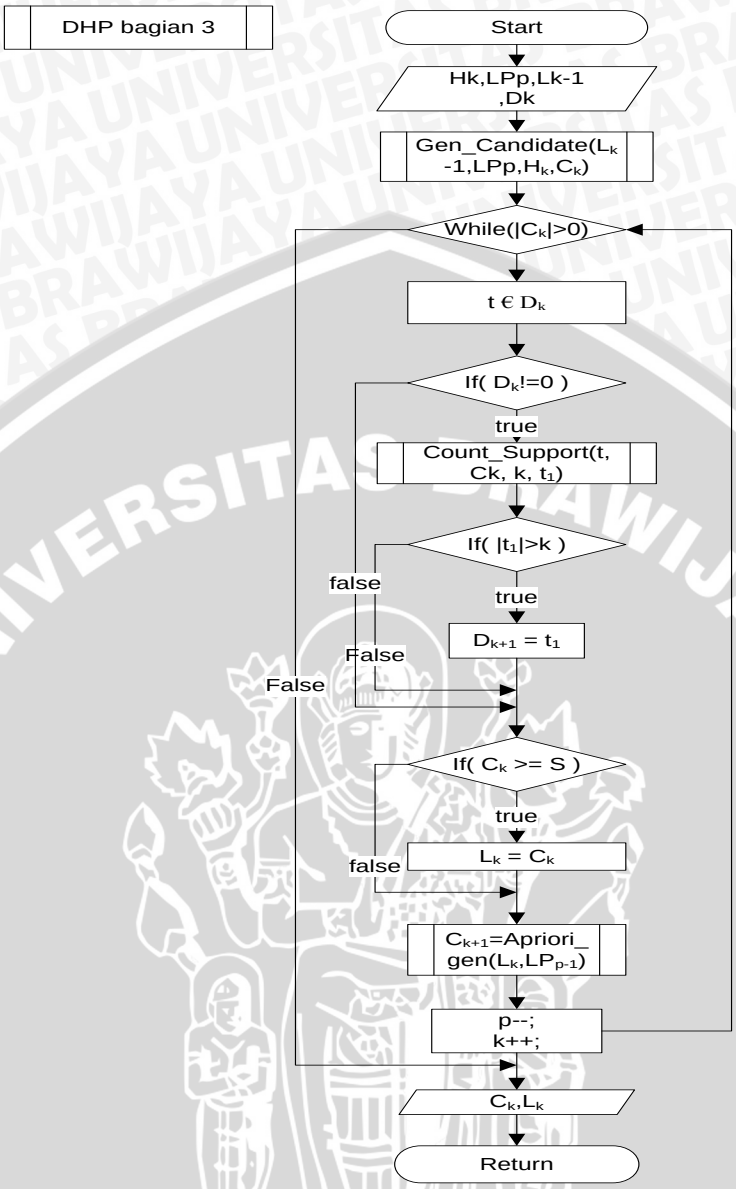


Gambar 3. 15 Flowchart Make Hash (2)

3.3.2.2.4 DHP bagian Ketiga

Pada algoritma DHP bagian ketiga ini merupakan bagian dari metode M-DHP, dimana proses pada bagian ini hampir sama dengan algortima DHP bagian kedua. DHP bagian ketiga ini juga terdapat proses pencarian kandidat k-*itemset* (C_k) menggunakan prosedur *gen candidate* dan algoritma bagian ketiga ini akan terus berjalan selama terdapat C_k, maka berdasarkan gambar 3.16 dibawah ini, langkah-langkah dari penerapan metode DHP bagian ketiga adalah sebagai berikut:

1. Masukan pada proses ini adalah H_k, LP_p, dan $L_{k-1,}$ $D_{k,}$ dimana D_k merupakan data transaksi dari hasil tahapan DHP bagian ke dua.
2. Selanjutnya sama seperti DHP bagian kedua akan dicari kandidat *k-itemset* menggunakan prosedur *gen candidate*.
3. Proses selanjutnya akan dilakukan reduksi database transaksi pada prosedur *count support* dan menghasilkan transaksi t_1, apabila jumlah data transaksi t_1 memenuhi syarat lebih dari nilai k maka t_1 sebagai databse transaksi yang baru D_{k+1}.
4. Untuk memperoleh anggota L_k maka mengecek jumlah *support* dari C_k, apakah memenuhi syarat dari *minimum support* yang telah ditentukan sebelumnya. Adapun apabila memenuhi, maka C_k tersebut menjadi anggota dari L_k.
5. Perbedaan pada algoritma ini dengan DHP bagian kedua yaitu pada proses *generate* kandidat *k-itemset* tidak menggunakan *make hash* atau dengan kata lain bukan menggunakan struktur data *hash table* tetapi menggunakan prosedur *apriori gen*.
6. Output pada algortima DHP bagian ketiga ini menghasilkan L_k dan C_k. Adapun untuk langkah 3, 4, dan 5 akan dilakukan secara terus menerus selama terdapat C_k yang dihasilkan.



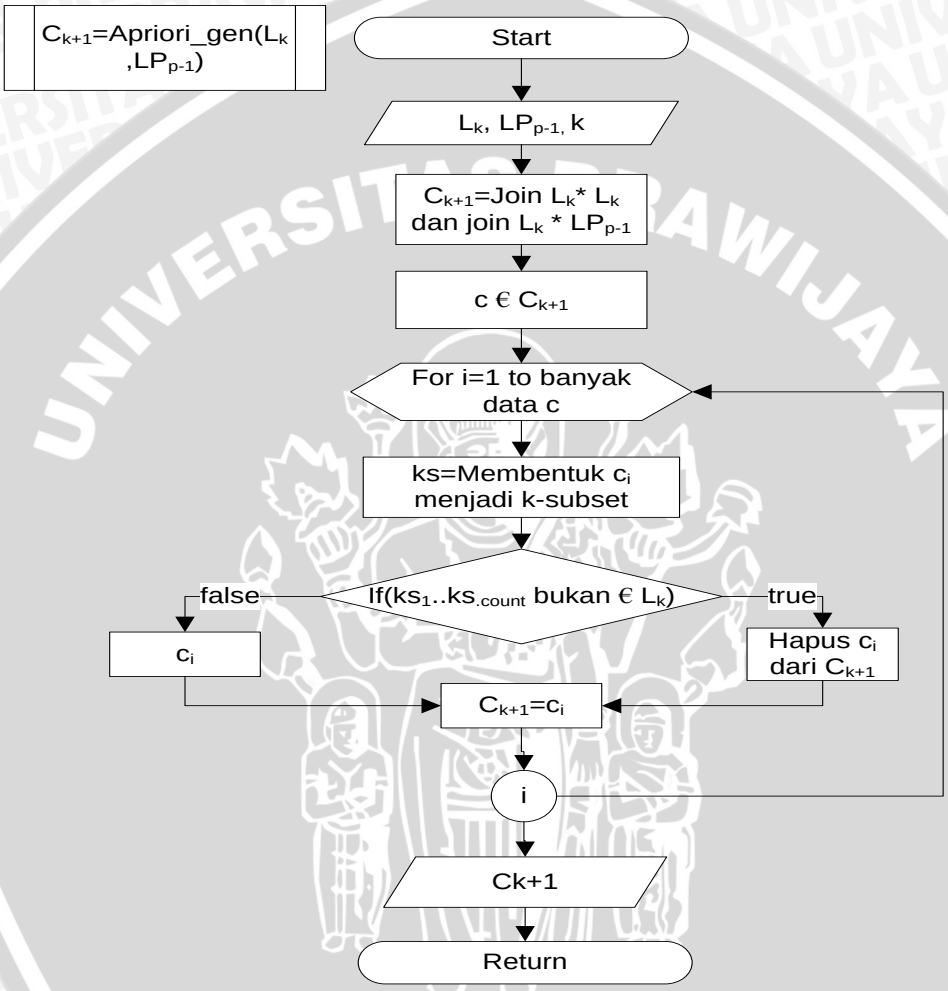
Gambar 3. 16 Flowchart DHP bagian Ketiga

3.3.2.2.4.1 Apriori Gen

Seperti yang telah dijelaskan pada 3.3.2.2.4 untuk membentuk kandidat itemset pada DHP bagian ketiga ini tidak menggunakan struktur hash table seperti pada DHP bagian kedua. Adapun untuk proses pada prosedur ini berdasarkan gambar 3.17 adalah sebagai berikut:

1. Masukan data adalah L_k dan $LP_{p-1,}$ dan k (k-itemset).
2. Proses selanjutnya melakukan join L_k dengan L_k dan L_k dengan LP_{p-1} sehingga menghasilkan C_{k+1}.

3. Kemudian setelah C_{k+1} terbentuk, masih perlu melakukan pengecekan yaitu dengan cara C_{k+1} tersebut akan dibentuk menjadi k-subset dan dicek apakah k-subset tersebut merupakan elemen dari L_k, apabila merupakan anggota elemen dari L_k maka akan menghasilkan C_{k+1}.



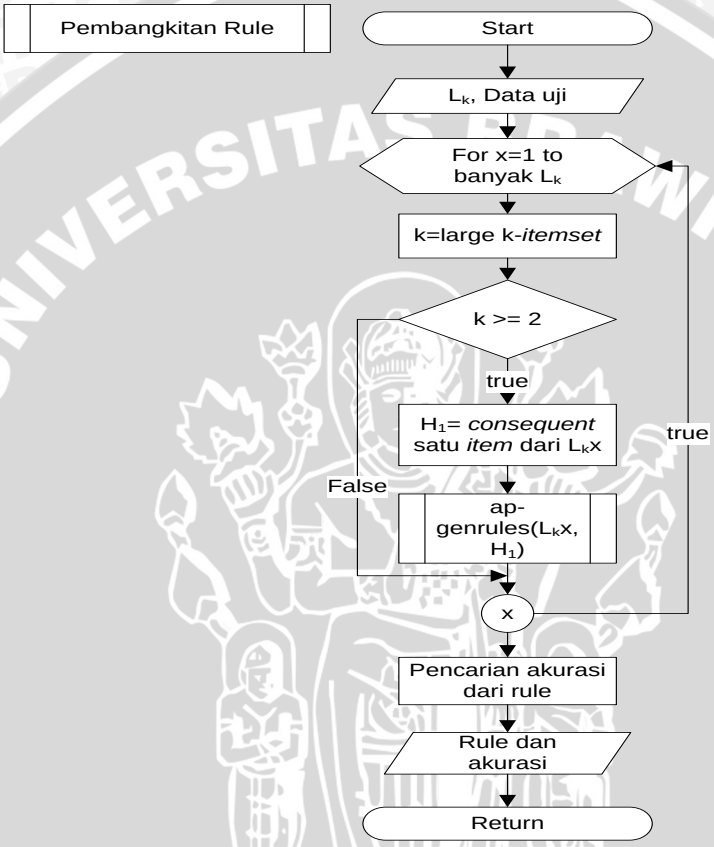
Gambar 3. 17 Flowchart Apriori Gen

3.3.2.3 Generate Rules

Proses generate rule pada penelitian ini menggunakan *Faster Algorithm*. Menggunakan metode ini karena pada penelitian ini memerlukan metode generate rule yang mampu menghasilkan *consequent* lebih dari satu item.

Berdasarkan gambar 3.18 dibawah ini, langkah-langkah dari penerapan metode *faster algorithm* adalah sebagai berikut:

1. Masukan berupa L_k dan data uji. Nilai dari k disini merupakan nilai k-*itemset*.
2. Dilakukan pengecekan jika k ≥2 maka akan melakukan proses H_1 dan menjalankan prosedur *ap-genrules*.
3. Setelah seluruh *rule* terbentuk, maka tahapan akhir mencari akurasi dari *rule* yang terbentuk

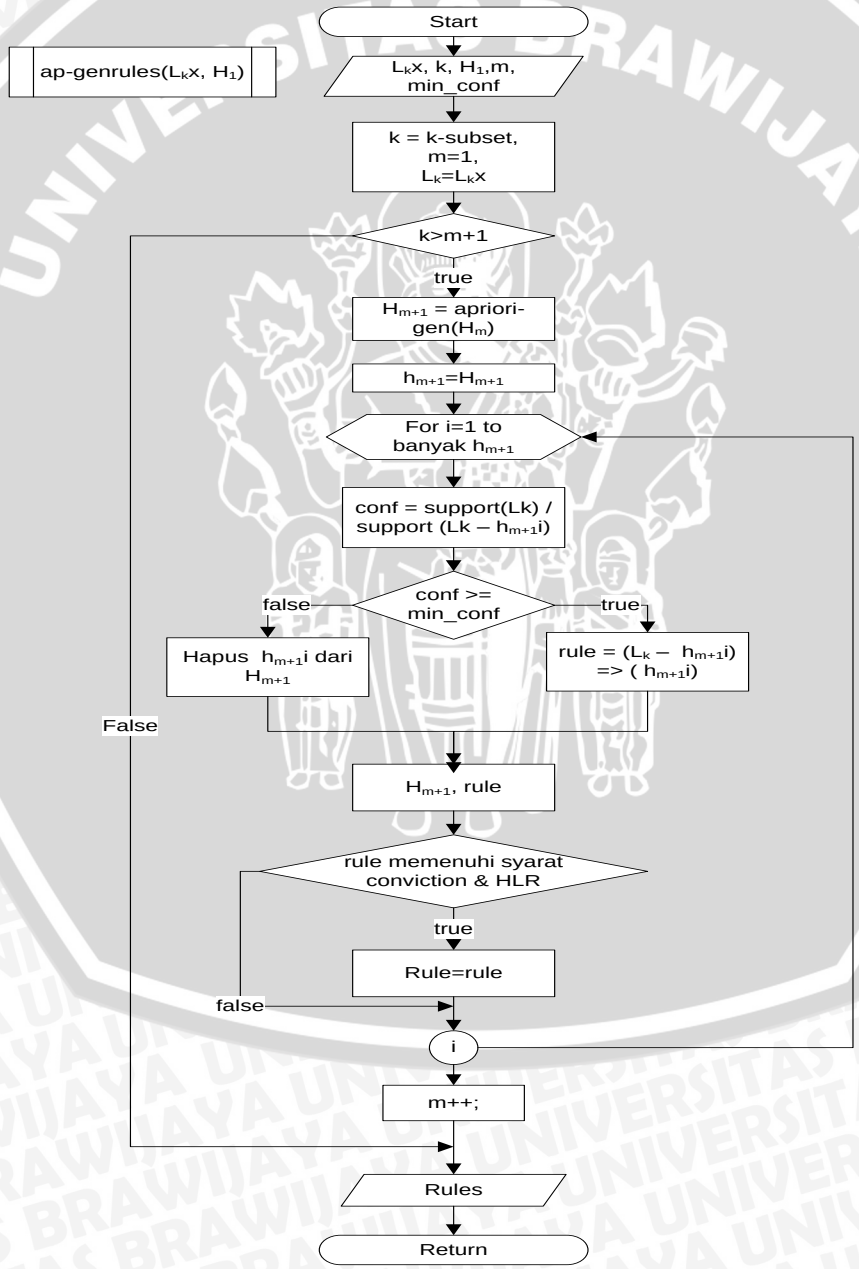


Gambar 3. 18 Flowchart Faster Algorithm

Berdasarkan gambar 3.19, langkah-langkah dari penerapan metode prosedur *ap-genrules* adalah sebagai berikut:

1. Masukan atau input L_k, k, H_1, m, dan *minimum confident*.
2. Inisialisasi k yang merupakan nilai k-*subset* dan m=1;
3. Melakukan pengecekan apabila nilai k > m+1 maka membentuk H_{m+1} dengan menjalankan prosedur apriori-gen, dimana H_{m+1} adalah *itemset* untuk *consequent*.

4. Melakukan inisialisasi $h_{m+1} = H_{m+1}$, kemudian melakukan *iterasi* untuk mencari nilai *confident* dan melakukan pengecekan, jika nilai *confident* yang dicari (conf) memenuhi syarat dari inputan *minimum confident* maka melakukan proses *generate rule*.
5. Pada proses metode ini output yang dihasilkan adalah rule dengan *consequent* lebih dari satu item dan merupakan *rule* dengan tingkat kakuatan yang baik berdasarkan syarat dari *conviction* dan *hiper lift ratio*.



Gambar 3. 19 Flowchart ap-genrules

3.4 Perhitungan Manual menggunakan algoritma Multipass Direct Hashing and Prunning (M-DHP)

Tabel 3. 2 Data Topik-topik Al Qur'an

Kode Item	Item
1	Keutamaan bersuci
2	Membersihkan bejana air
3	Kebersihan pakaian
4	Haid
5	Najis
6	Disyariatkannya wudhu
7	Hukum wudhu
8	Rukun wudhu
9	Yang membatalkan wudhu
10	Hukum mandi besar
11	Yang diharamkan dan diperbolehkan orang yang junu
12	Disyariatkannya tayammum
13	Rukun tayammum
14	Cara tayammum

Tabel 3. 3 Dataset topik pada surat-surat Al Qur'an

TID	ITEMSET
105	Keutamaan bersuci, Membersihkan bejana, Haid, Rukun tayammum
106	Keutamaan bersuci, Membersihkan bejana, Haid, membatalkan wudhu, Rukun tayammum
107	Keutamaan bersuci, Membersihkan bejana, Haid, Najis, Disyariatkannya wudhu
108	Keutamaan bersuci, Membersihkan bejana, Haid, Najis, Disyariatkannya tayammum, Rukun tayammum
109	Keutamaan bersuci, Membersihkan bejana, Disyariatkannya tayammum, Rukun tayammum

Tabel 3. 4 Dataset topik surat-surat Al Qur'an yang berupa TID (surat) dan *itemset* (kode item topik) hasil dari tahapan Preprocessing

TID	ITEMSET
105	1,2,4,13
106	1,2,4,9,13
107	1,2,4,5,6
108	1,2,4,5,12,13
109	1,2,12,13

Dimana:

TID : nomor transaksi.

Itemset: hasil dari pemetaan *itemset* pada tabel 3.3 dengan kumpulan topik pada tabel 3.2 berdasarkan dari kode itemnya.

Langkah 1: Inisialisasi dari *minimum support*, *minimum confidence* dan *n-partisi*. *Minimum Support* ≥ 40%, dengan asumsi bahwa (40/100)*5 = 2 maka jumlah *support* (*support count*) untuk memenuhi syarat dari *minimum support* harus ≥ 2. *Minimum Confidence* ≥ 50%. *n-partisi*= 3.

Langkah 2: Menemukan kandidat *1-itemset* (C_1), *hash table 2-itemset* (H_2) dan *large 1-itemset* (L_1) pada DHP bagian pertama.

Proses pertama adalah menemukan kandidat *1-itemset* (C_1) dan hasilnya terdiri dari *itemset* {1, 2, 4, 5, 6, 9, 12, 13}, namun C_1 tersebut belum memiliki jumlah *support*. Oleh karena itu setelah menemukan kandidat *1-itemset* (C_1) maka proses selanjutnya adalah mencari jumlah *support* dari setiap C_1. Cara untuk mencari jumlah *support* dari setiap anggota kandidat *1-itemset* (C_1) yaitu dengan melakukan *insert* kandidat *1-itemset* (C_1) ke dalam *hash tree* sekaligus dengan melakukan proses mencari jumlah dari setiap item yang sama. Seperti yang ditunjukkan pada gambar 3.20 bahwa setiap item dari *1-itemset* memiliki jumlah *support*, misalnya pada node ke-1 terdapat *itemset* 1(5), dimana 1 merupakan kandidat *1-itemset* (C_1) dan 5 adalah jumlah *support*. Telah dipaparkan pada bab tinjauan pustaka, bahwa untuk melakukan *insert* pada *hash tree* menggunakan *hash function* (2.4) dan pada perhitungan manual ini menggunakan *hash function* h(x)= x mod 7. Pada *hash tree* juga terdapat jumlah maksimum dari node atau *max node size* dan pada perhitungan manual ini *max node size* =3. Adapun kandidat *1-itemset* (C_1) apabila ditabelkan, maka hasilnya seperti yang ditunjukkan pada tabel 3.5.

Gambar 3. 20 Hash tree C1 dan jumlah support

Tabel 3. 5 kandidat 1-itemset (C1).

Itemset	Jumlah Support
1	5
2	5
4	4
5	2
6	1
9	1
12	2
13	4

Proses selanjutanya membentuk *hash table 2-itemset* (H_2) yaitu dengan cara membentuk *2-itemset* yang merupakan kombinasi dari data transaksi pada tabel 3.3. Adapun contohnya, misalkan untuk TID 105 yang terdiri dari kode item {1, 2, 4, 13} maka hasil dari pembentukan 2-*itemset* yaitu {(1,2); (1,4); (1,13); (2,4); (2,13); (4,13)}. Selanjutnya proses memperoleh alamat *bucket* yang digunakan untuk menyimpan data *2-itemset* pada *hash table* dengan menggunakan *hash function* (2.5) yaitu $hn(x1, \dots, xn) = \left(\sum_{i=1}^{n} xi \,.\, A^{k-(i-1)}\right) mod\ s$. Khusus pada perhitungan manual ini A=10 dan s=17, maka pada tahap ini menghasilkan *hash function* h(x,y)=(order of x*10 + order of y) mod 17.

Contoh perhitungan untuk proses memperoleh alamat *bucket* dari data *2-itemset* {(1,2); (1,4); (1,13); (2,4); (2,13); (4,13)} adalah sebagai berikut:

-*Itemset* (1,2). Maka H(1,2) = (1*10 + 2) mode 17 = 12 mod 17 =12.
-*Itemset* (1,4). Maka H(1,4) = (1*10 + 4) mode 14 = 14 mod 17 =14.
-*Itemset* (1,13). Maka H(1,13) = (1*10 + 13) mode 17= 23 mod 17 =6.
-*Itemset* (2,4). Maka H(2,4) = (2*10 + 4) mode 17 = 24 mod 17 =7.
-*Itemset*(2,13). Maka H(2,13)= (2*10 + 13) mode 17= 33 mod 17 =16.
-*Itemset*(4,13). Maka H(4,13) = (4*10 + 13) mode 17 = 53 mod 17 =2.

Setelah alamat *bucket* telah diketahui dari proses tersebut, maka *2-itemset* tersebut dimasukkan ke *hash table* H_2 yang hasilnya ditunjukkan pada tabel 3.6.

Seperti yang telah ditunjukkan pada tabel 3.6 terdapat data *2-itemset* di isi *bucket*, misalkan pada alamat *bucket* 1 terdapat *itemset* (9,13) dan (4,12). Pada H_2 juga terdapat jumlah dan *bit vector*, dimana jumlah adalah banyak data *2-itemset* di tiap alamat *bucket* dan *bit vector* merupakan penanda apakah jumlah data *2-itemset* pada H_2 memenuhi syarat *minimum support* atau tidak. Apabila nilai *bit vector* 1 maka memenuhi syarat *minimum support* dan apabila *bit vector* sama dengan 0 maka tidak memenuhi syarat *minimum support*.

Tabel 3. 6 2-itemset pada H2

Alamat Bucket	Jumlah	Bit Vector	Isi Bucket
0			
1	2	1	(9,13); (4,12)
2	4	1	(4,13); (1,9); (4,13); (1,13)
3			
4			
5	3	1	(5,6); (1,12); (1,12)
6	3	1	(1,13); (1,13); (1,13)
7	4	1	(2,4); (2,4); (2,4); (2,4)
8	2	1	(2,5); (2,5)
9	1	0	(2,6)
10			
11	2	1	(4,5); (4,5)
12	7	1	(1,2); (1,2); (2,9); (1,2); (4,6); (1,2); (1,2)
13			
14	7	1	(1,4); (1,4); (1,4); (1,4); (12,13); (12,13)
15	5	1	(1,5); (4,9); (1,5); (2,12); (2,12)
16	5	1	(2,13); (2,13); (1,6); (2,13); (2,13)

Proses akhir dari langkah 1 ini menemukan *large 1-itemset* (L_1), adapun L_1 merupakan hasil dari C_1 yang memenuhi dari syarat *minimum support*. Jadi kandidat *1-itemset* (C_1) yang memiliki jumlah *support* lebih dari sama dengan 2 akan menjadi *large 1-itemset* (L_1).

Tabel 3. 7 large 1-itemset (L1)

Itemset	Jumlah Support
1	5
2	5
4	4
5	2
12	2
13	4

Langkah 3: Melakukan proses partisi

Langkah 3 ini merupakan proses partisi, dimana hasil dari *large 1-itemset* (L_1) dipartisi sebanyak n-partisi yang diinginkan, adapun pada manual perhitungan ini menggunakan n-partisi = 3.

Tabel 3. 8 large 1-itemset (L1)

Itemset	Jumlah Support
1	5
2	5
4	4
5	2
12	2
13	4

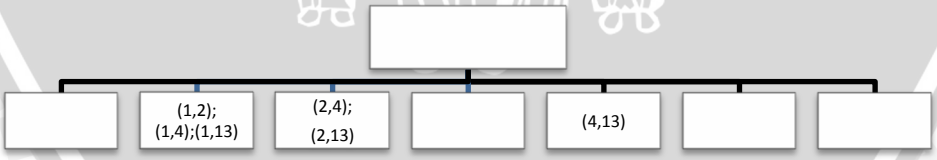
Pada tabel 3.8 tersebut dilakukan *shorting* dari jumlah *support* terkecil ke jumlah terbesar, maka hasil dari *shorting* dan sekaligus dengan hasil partisi yang ditunjukkan pada tabel 3.9:

Tabel 3. 9 large 1-itemset (L1) yang telah diurutkan secara ascending dan hasil partisi.

Itemset	Jumlah Support	Partisi
5	2	LP_1
12	2	LP_2
4	4	LP_3
13	4	
1	5	
2	5	

Langkah 4: Menemukan kandidat *k-itemset* (C_k), large *k-itemset* (L_k), data database transaksi yang tereduksi, dan membentuk *hash table k+1-itemset* (H_{k+1}) pada DHP bagian kedua ini.

Pada proses ini yaitu DHP bagian kedua akan terus berjalan selama jumlah dari *bucket* pada *hash table k-itemset* (H_k) atau ketika saat ini H_2 yang memenuhi dari syarat *minimum support* lebih besar sama dengan dari nilai *Large*. Untuk awal dari tahap ini memiliki nilai k = 2 dan *Large* = 2, karena jumlah *bucket* yang memenuhi syarat *mimimum support* berjumlah 11 lebih besar dari nilai *Large* yaitu 2 maka proses DHP bagian 2 dapat dikerjakan. Proses selanjutnya menemukan kandidat *k-itemset* menggunakan prosedur *gen candidate* yaitu proses join dari hasil proses partisi. Untuk tahap ini sesuai dengan metode M-DHP, maka yang diproses terlebih dahulu yaitu LP_3 {1, 2, 4, 12, 13} yang ditunjukkan pada tabel 3.9 dan hasil dari proses join pada prosedur *gen candidate* adalah {(1,2); (1,4); (1,12); (1,13); (2,4); (2,12); (2,13); (4,12); (4,13); (12,13)} namun karena proses join pada prosedur *gen candidate* yang diambil sebagai kandidat *k-itemset* (C_k) merupakan C_k yang telah memenuhi syarat *minimum support* hasil dari mengecek pada H_2 maka hasil dari C_k atau saat ini C_2 yaitu {4,13; 1,4; 2,4; 1,13; 2,13; 1,2;}, kemudian tahapan selanjutnya melakukan *insert* C_2 ke *hash tree* dengan menggunakan *hash function* h(x)= x mod 7 yang juga digunakan pada langkah 2.



Gambar 3. 21 Hash tree C2

Setelah memperoleh kandidat *2-itemset* (C_2) pada *hash tree*, maka proses selanjutnya mencari jumlah *support* dari C_2 dan data transaksi yang telah tereduksi pada prosedur *count support*.

Gambar 3. 22 Hash tree C2 dan jumlah support

Tabel 3. 10 Daftar k-subset dan Jumlah dari item transaksi

TID	Kode Item	k-Subset	Jumlah Item
105	1,2,4,13	(1,2); (1,4); (1,13); (2,4); (2,13); (4,13)	1-3, 2-3, 4-3, -13-3
106	1,2,4,9,13	(1,2); (1,4); (1,9); (1,13); (2,4); (2,9); (2,13); (4,9); (4,13)	1-3, 2-3, 4-3, 9-0, 13-3
107	1,2,4,5,6	(1,2); (1,4); (1,5); (1,6); (2,4); (2,5); (2,6); (4,5); (4,6); (5,6)	1-2, 2-2, 4-2, 5-0, 6-0
108	1,2,4,5,12,13	(1,2); (1,4); (1,5); (1,12); (1,13); (2,4); (2,5); (2,12); (2,13); (4,5); (4,12); (4,13); (5,12); (5,13); (12,13)	1-3, 2-3, 4-3, 5-0, 12-0, 13-2
109	1,2,12,13	(1,2); (1,12); (1,13); (2,12); (2,13); (12,13)	1-2, 2-2, 12-0,13-2

Proses pada *count support* merupakan proses reduksi database transaksi dan mencari jumlah *support* dari C_2. Tahapan pada *count support* yaitu pertama membentuk k-subset dari transaksi terlebih dahulu, proses ini seperti yang ditunjukkan pada tabel 3.10. Langkah-langkah prosesnya yaitu membentuk *k-subset* atau untuk saat ini *2-subset* dan tahapannya misalkan untuk TID 105 yang terdiri dari kode item {1, 2, 4, 13} maka hasil dari pembentukan k-*subset* atau *2-subset* yaitu dengan melakukan join antar item pada TID 105 dan hasilnya {(1,2); (1,4); (1,13); (2,4); (2,13); (4,13)}. Untuk tahapan selanjutnya adalah melakukan pencarian setiap data k-subset di *hash tree* dengan menggunakan *hash function* h(x)= x mod 7. Cara pencarian hampir sama dengan cara *insert* pada *hash tree* dan contoh prosesnya seperti yang telah dipaparkan pada bab tinjauan pustaka sub-bab 2.4 tentang Struktur data *Hash Tree*, namun terdapat sedikit perbedaan yaitu apabila *k-subset* ditemukan pada *hash tree* maka jumlah dari C_k tersebut bertambah satu dan hasilnya seperti yang ditunjukkan pada gambar 3.22.

Untuk contoh misal pada node ke-4 pada gambar 3.22 terdapat *itemset* (4,13)3 maka artinya (4,13) adalah kandidat *2-itemset* dan 3 adalah jumlah *support*. Masih proses pada prosedur *count support* yaitu mencari jumlah dari tiap item data transaksi berdasar dari *k-subset* yang terbentuk. Tahapan prosesnya yaitu ketika *2-subset* ada pada *hash tree* maka proses selanjutnya mengecek apakah item pada data transaksi sama dengan item-item pada *2-subset* tersebut, apabila sama maka jumlah item dari data transaksi tersebut bertambah satu, hal ini seperti yang ditunjukkan pada Tabel 3.10 kolom jumlah item pada TID 105 yaitu terdapat data 1-3, dimana 1 merupakan item dari data transaksi dan 3 adalah jumlah dari item transaksi tersebut berdasarkan dari *2-subset* yang terbentuk. Setelah seluruh k-subset telah diproses, maka selanjutnya melakukan reduksi database transaksi yaitu dengan cara melihat apakah jumlah item pada tabel 3.10 yang merupakan data dari tiap item transaksi ≥ k (k saat ini 2), apabila tidak memenuhi akan dilakukan reduksi. Hasil dari proses reduksi tersebut ditunjukkan pada tabel 3.11:

Tabel 3. 11 Data Transaksi tereduksi (t1) pada prosedur count support

TID	Kode Item
105	1, 2, 4, 13
106	1, 2, 4, 13
107	1, 2, 4
108	1, 2, 4, 13
109	1, 2, 13

Proses selanjutnya yaitu membentuk H_{k+1} pada prosedur *Make Hash*, namun dengan syarat jumlah dari t_1 lebih dari k. Karena nilai k sekarang sama dengan 2 dan jumlah t_1= 5 maka syarat tersebut terpenuhi, oleh karena itu proses membentuk H_{k+1} dilakukan.

Tabel 3. 12 Data Transaksi t1 dibentuk menjadi k+1-subset dan k-subset

TID	k+1-Subset	k-Subset
105	(1,2,4); (1,2,13); (1,4,13); (2,4,13)	{(1,2);(1,4);(2,4)}, {(1,2); (1,13); (2,13)}, {(1,4); (1,13); (4,13)}, {(2,4);(2,13);(4,13)}

106	(1, 2, 4); (1,2,13); (1,4, 13); (2, 4, 13)	{1,2;1,4;2,4;}, {1,2; 1,13; 2,13;}, {1,4; 1,13; 4,13}, {2,4;2,13;4,13;}
107	(1, 2, 4)	{1,2; 1,4; 2,4;}
108	(1, 2, 4); (1,2,13); (1,4, 13); (2, 4, 13)	{1,2;1,4;2,4;}, {1,2; 1,13; 2,13;}, {1,4; 1,13; 4,13}, {2,4;2,13;4,13;}
109	(1, 2, 13)	{1,2;1,13;2,13;}

Tabel 3.12 *merepresentasikan* proses dari prosedur *make hash* yaitu pertama membentuk *k+1-subset* dengan cara melakukan join, adapun contohnya misal pada TID 105 pada tabel 3.10 terdapat data transaksi yang berupa kode item {1, 2, 4, 13} dan kemudian melakukan join yang menghasilkan k+1-subset {(1,2,4); (1,2,13); (1,4,13); (2,4,13)} seperti yang ditunjukkan pada tabel 3.11, namun tidak semua data *k+1-subset* yang terbentuk menjadi data pada H_{k+1} (H_3) karena pada tahap ini juga melakukan pengecekan *k+1-subset* yang akan diletakkan pada H_{k+1} yaitu dengan cara memebentuk *k-subset* dari *k+1-subset* seperti yang ditunjukkan pada tabel 3.12. dan apabila seluruh *k-subset* tersebut memenuhi dari syarat dari *minimum support* pada H_k atau saat ini H_2 maka data k+1-subset dimasukkan kedalam H_{k+1} untuk membentuk H_3 dengan menggunakan *hash function* (2.5) yaitu $hn(x1, \dots, xn) = \left(\sum_{i=1}^{n} xi \,.\, A^{k-(i-1)}\right) mod\ s$. Pada perhitungan manual tahap ini A=10, s=17, dan membentuk *3-itemset* pada H_3, sehingga berdasarkan fungsi (2,6) menghasilkan *hash function* h(x,y,z)=(order of x*100 + order of y*10 + order of z) mod 17. Adapun contoh perhitungan untuk membentuk H_3, misalkan untuk *itemset* {(1,2,4);(1,2,13);(1,4,13); (2,4,13)} pada TID 105 adalah sebagai berikut:

-*Itemset* (1,2,4). Maka H(1,2,4) = (1*100 + 2*10+4) mode 17 = 124 mod 17 =5.
-*Itemset* (1,2,13). Maka H(1,2,13) = (1*100 + 2*10+13) mode 14 = 133 mod 17 =14.
-*Itemset* (1,4,13). Maka H(1,4,13) = (1*100 + 4*10+13) mode 17= 153 mod 17 =6.
-*Itemset* (2,4,13) tidak menjadi data *3-subset* pada H_3 karena *k-subset* atau *2-subset* yang terbentuk yaitu {(2,4);(2,13);(4,13)} ada yang tidak memenuhi syarat *minimum support* pada H_2 yaitu *itemset* (2,13).

Untuk hasil dari proses tersebut yaitu *hash table 3-itemset* (H_3) yang ditunjukkan pada tabel 3.13.

Tabel 3. 13 3-subset pada H3

Alamat Bucket	Jumlah	Bit Vector	Isi Bucket
0	3	1	1,4, 13; 1,4, 13; 1,4, 13;
1			
2			
3			
4			
5	4	1	1, 2, 4;1, 2, 4; 1, 2, 4; 1, 2, 4;
6			
7			
8			
9			
10			
11			
12			
13			
14	4	1	1,2,13; 1, 2, 13; 1,2,13; 1,2,13;
15	3	1	2, 4, 13; 2, 4, 13; 2, 4, 13;
16			

Hampir sama dengan proses pada *count support* yaitu pada tahap ini ini juga melakukan reduksi database, namun berbeda syarat saja karena pada *make hash* akan mereduksi data transaksi apabila jumlah item yang diperoleh dari mencocokkan 3-itemset yang terbentuk pada H_3 dengan item data transakai sama dengan nol, jadi jumlah minimal item pada data transaksi adalah 1 agar tidak terkena reduksi. Adapun hasilnya seperti yang ditunjukkan pada tabel 3.14:

Tabel 3. 14 Data Transaksi hasil reduksi (t2) make hash

TID	Kode Item
105	1, 2, 4, 13
106	1, 2, 4, 13
107	1, 2, 4
108	1, 2, 4, 13
109	1, 2, 13

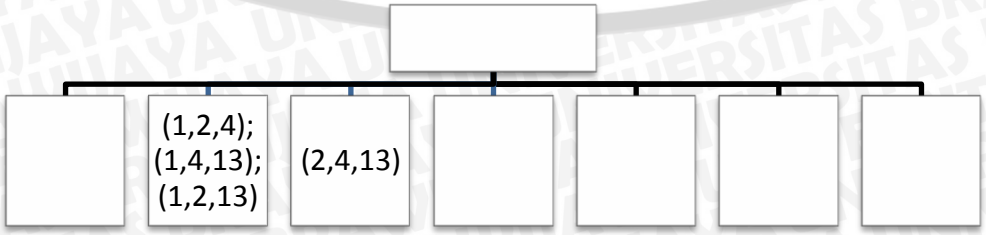
Setelah ketiga proses tersebut dilakukan, untuk tahap ini dimana k = 2 juga menghasilkan L_2 yaitu kandidat *2-itemset* (C_2) pada *hash tree* yang memenuhi syarat *minimum support* dan hasilnya seperti yang ditunjukkan pada tabel 3.15:

Tabel 3. 15 large 2-itemset (L2)

Itemset	Jumlah support
1,2	5
1,4	4
1,13	4
2,4	4
2,13	3
4,13	3

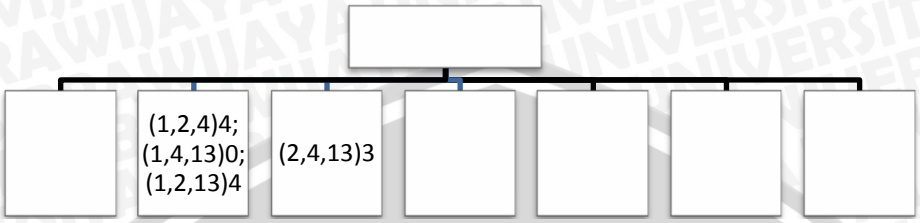
Proses ini masih merupakan Langkah 4, karena jumlah dari *bucket* yang memenuhi syarat dari *minimum support* pada H_3 berjumlah 4 dan masih memenuhi syarat lebih dari sama dengan nilai *Large*, dimana nilai *Large* sekarang sama dengan 3.

Sama dengan proses sebelumnya, untuk proses ini yaitu menemukan kandidat *k-itemset* menggunakan prosedur *gen candidate*, dimana hasil join dari L_2 dengan L_2 dan L_2 dengan p_2 atau partisi bagian ke-2 dan menghasilkan kandidat *3-itemset* (C_3) yaitu {(1,4,13); (2,4,13); (1,2,4); (1,2,13); (4,12,13); (1,4,12); (2,4,12); (1,12,13); (2,12,13); (1,2,12)}, namun karena proses pada prosedur *gen candidate* yang diambil sebagai kandidat *3-itemset* (C_3) merupakan C_3 yang telah memenuhi syarat *minimum support* yang dilihat pada H_3 maka hasil dari C_k atau saat ini C_3 yaitu {(1,4,13); (2,4,13); (1,2,4); (1,2,13)}. Tahapan berikutnya yaitu melakukan *insert* C_3 ke *hash tree* dengan menggunakan *hash function* h(x)= x mod 7 yang juga digunakan pada langkah 2 dan hasilnya seperti yang ditunjukkan pada gambar 3.23.



Gambar 3. 23 Hash tree C3

Tahap selanjutnya merupakan proses memperoleh jumlah *support* dari C_3 pada *hash tree* dan data transaksi yang telah tereduksi pada prosedur *count support*.



Gambar 3. 24 Hash tree C3

Tabel 3. 16 Daftar k-subset dan Jumlah dari item transaksi

TID	Kode Item	k-Subset	Jumlah Item
105	1, 2, 4, 13	(1,2,4); (1,2,13); (2,4,13)	1-2, 2-3, 4-2, 13-2
106	1, 2, 4, 13	(1,2,4); (1,2,13); (2,4,13)	1-2, 2-3, 4-2, 13-2
107	1, 2, 4	(1,2,4)	1-1,2-1,4-1
108	1, 2, 4, 13	(1,2,4); (1,2,13); (2,4,13)	1-2, 2-3, 4-2, 13-2
109	1, 2, 13	(1,2,13)	1-1,2-1,13-1

Sama seperti proses sebelumnya bahwa proses pada *count support* merupakan proses reduksi database transaksi dan mencari jumlah *support* dari C_3. Tahapan pada *count support* yaitu pertama membentuk k-subset dari transaksi terlebih dahulu, proses ini seperti yang ditunjukkan pada tabel 3.16. Langkah-langkah prosesnya yaitu membentuk *k-subset* atau untuk saat ini *3-subset* (sekarang nilai k=3) dan tahapannya misalkan untuk TID 105 yang terdiri dari kode item {1, 2, 4, 13} maka hasil dari pembentukan k-*subset* atau *3-subset* yaitu dengan melakukan join antar item pada TID 105 dan hasilnya (1,2,4); (1,2,13); (2,4,13)}. Untuk tahapan selanjutnya adalah melakukan pencarian setiap data k-subset di *hash tree* dengan menggunakan *hash function* h(x)= x mod 7. Cara pencarian hampir sama dengan cara *insert* pada *hash tree* dan contoh prosesnya seperti yang telah dipaparkan pada bab tinjauan pustaka sub-bab 2.4 tentang Struktur data *Hash Tree*, namun terdapat sedikit perbedaan yaitu apabila k-subset ditemukan pada *hash tree* maka jumlah dari k-subset tersebut bertambah satu dan hasilnya seperti yang ditunjukkan pada gambar 3.24. Untuk contoh misal pada node ke-2 pada gambar 3.24 terdapat *itemset* (2,4,13)3 maka artinya (2,4,13) adalah kandidat *3-itemset* dan 3 adalah jumlah *support*. Masih proses pada

prosedur *count support* yaitu mencari jumlah dari tiap item transaksi berdasar dari k-subset yang terbentuk . Tahapan prosesnya yaitu ketika *3-subset* ada pada *hash tree* maka proses selanjutnya mengecek apakah item pada data transaksi sama dengan item-item pada *3-subset* tersebut, apabila sama maka jumlah item dari data transaksi tersebut bertambah satu, hal ini seperti yang ditunjukkan pada Tabel 3.16 kolom jumlah item pada TID 105 yaitu terdapat data 1-2, dimana 1 merupakan item dari data transaksi dan 2 adalah jumlah dari item transaksi tersebut berdasarkan dari *3-subset* yang terbentuk. Setelah seluruh k-subset telah diproses, maka selanjutnya melakukan reduksi database transaksi yaitu dengan cara melihat apakah jumlah item pada tabel 3.16 yang merupakan data dari tiap item transaksi ≥ k (k saat ini 3), apabila tidak memenuhi akan dilakukan reduksi. Hasil dari proses reduksi tersebut ditunjukkan pada tabel 3.17:

Tabel 3. 17 Data Transaksi hasil reduksi (t1) count support

TID	Kode Item
105	2
106	2
107	kosong
108	2
109	kosong

Karena jumlah transaksi t_1 pada Tabel 3.17 dari hasil *count support* tidak memenuhi yaitu $|t_1|>k$ dimana k = 3 dan jumlah $t_1=3$ maka prosedur *make hash* tidak dapat dilaksanakan dan langsung menghasilkan L_k atau L_3 yaitu kandidat *3-itemset* (C_3) pada *hash tree* yang memenuhi syarat *minimum support*. Hasil dari L_3 seperti yang ditunjukkan pada tabel 3.18.

Tabel 3. 18 large 3-itemset (L_3)

Itemset	**Jumlah support**
2,4,13	3
1,2,4	4
1,2,13	4

Langkah 5: Pada tahap ini hampir sama dengan Langkah 4 yaitu menemukan kandidat *k-itemset* (C_k), large *k-itemset* (L_k), data database transaksi yang tereduksi, namun tidak ada proses pembentukan H_{k+1} pada tahap DHP bagian ketiga ini.

Pada DHP bagian ketiga ini akan terus berjalan membentuk *itemset* baik itu kandidat *k-itemset* ataupun *large k-itemset* selama masih ada kandidat *k-itemset* yang dihasilkan pada prosedur *gen candidate* DHP bagian ketiga, karena tidah terbentuk H_{k+1} atau H_4 maka DHP bagian ketiga tidak dapat dilakukan. Sehingga pada perhitungan manual ini telah selesai dan tinggal melakukan pembentukan *rule*.

Langkah 6: Melakukan pembentukan *Rule* menggunakan metode *Faster Algorithm*.

Pada metode ini mampu menghasilkan *consequent* lebih dari satu item atau *rule* dengan *consequent* lebih dari satu item. Pada metode ini yang diproses mulai L_2 sampai L_{k+1} sedangkan L_1 tidak diproses dengan ketentuan *minimum support* ≥40% dan n-partisi= 3 sedangkan untuk *minimum confidence* ≥ 50%, oleh karena itu hasilnya seperti yang ditunjukkan pada tabel 3.19.

Tabel 3. 19 Generated Rules

Rule	Support (%)	Confident (%)	Conviction	Hiper Lift Ratio
***1=>2**	100%	(5/5)*100=100%	(1-(5/5))/1-1 = ∞	1/0,99*1= 1,0101
***2=>1**	100%	(5/5)*100=100%	(1-(5/5))/1-1 = ∞	1/0,99*(5/5) = 1,0101
1=>4	80%	(4/5)*100=80%	(1-4/5)/10,8 = 0,2/0,2= 1	0,8/0,99*(4/5) =0,8/0,792= 1,0101
***4=>1**	80%	(4/4)*100=100%	(1-5/5)/1-1= ∞	1/0,99*(5/5) = 1,0101
1=>13	80%	(4/5)*100=80%	(1-4/5)/1-0,8 = 1	0,8/0,99*(4/5) = 1,0101
***13=>1**	80%	(4/4)*100=100%	(1-5/5)/1-1= ∞	1/0,99*(5/5) = 1,0101
2=>4	80%	(4/5)*100=80%	(1-4/5)/1-0,8 = 1	0,8/0,99*(4/5) = 1,0101
***4=>2**	80%	(4/4)*100=100%	(1-5/5)/1-1= ∞	1/0,99*(5/5) = 1,0101

2=>13	60%	(3/5)*100=60%	(1-4/5)/(1-0.6) = 0,5	0.6/0,99*(4/5) =0,6/0,792= 0,7575
13=>2	60%	(3/4)*100=75%	(1-5/5)/1-0,75 = 0	0,75/0,99*(5/5) = 0,7575
4=>13	60%	(3/4)*100=75%	(1-4/5)/1-0,75 =0,2-0,25= 0,8	0,75/0,99*(4/5)=0, 75/0,792= 0,946946
13=>4	60%	(3/4)*100=75%	(1-4/4)/1-0,75 = 0	0,75/0,99*(4/5)=0, 75/0,792= 0,946946
2=>4,13	60%	(3/5)*100=60%	(1-3/5)/1-0,6 = 1	0,6/0,99*(3/5) = 1,0101
***4=>2,13**	60%	(3/4)*100=75%	(1-3/5)/1-0,75 =0,4/0,25= 1,6	0,75/0,99*(3/5)=0, 75/0,594= 1,2626
13=>2,4	60%	(3/4)*100=75%	(1-4/5)/1-0,75 =0,2/0,25= 0,8	0,75/0,99*(4/5)=0, 75/0,792= 0,946946
1=>2,4	80%	(4/5)*100=80%	(1-4/5)/1-0,8 =1	0,8/0,99*(4/5)=0,8/ 0,792= 1,0101
2=>1,4	80%	(4/5)*100=80%	(1-4/5)/1-0,8 = 1	0,8/0,99*(4/5)= 0,8/0,792= 1,0101
***4=>1,2**	80%	(4/4)*100=100%	(1-5/5)/1-1= ∞	1/0,99*(5/5) = 1,0101
***1=>2,13**	80%	(4/5)*100=80%	(1-3/5)/1-0,8 =0,4/0,2= 2	0,8/0,99*(3/5)=0,8/ 0,594= 1,346801
2=>1,13	80%	(4/5)*100=80%	(1-4/5)/1-0,8= 1	0,8/0,99*(4/5) = 1,0101
***13=>1,2**	80%	(4/4)*100=100%	(1-5/5)1-1= ∞	1/0,99*(5/5) = 1,0101

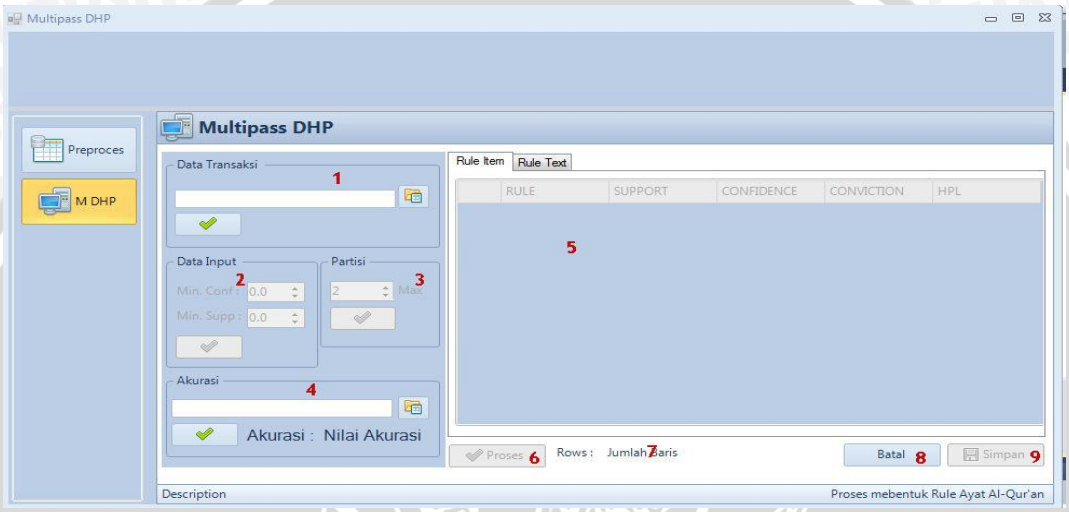
Untuk seluruh rule yang telah dipaparkan pada tabel 3-19 memenuhi *minimum confident*≥ 50%, namun setelah dilakukan evaluasi rule menggunakan *conviction* dan *hiper lift ratio* dengan ketentuan untuk seluruh rule harus memenuhi syarat dari *conviction* yaitu memiliki range nilai antara 0.5,...,1,...∞ dan untuk *hiper lift ratio* > 1,terdapat rule-rule yang tidak memenuhi. Adapun rule – rule yang bertanda bintang merupakan rule-rule yang memenuhi evaluasi dari *conviction* dan *hiper lift ratio*. Tahapan selanjutnya adalah mencari akurasi dari *rule* tersebut dengan menerapkan persamaan 2.8.

Tabel 3. 20 Data Uji

TID	KODE ITEM
110	1,2,4
111	1,2,4,9

Setelah dilakukan pengecekan seluruh *rule* yang bertanda bintang ke data uji, terdapat hasil akurasi sebesar 55,56 % sebesar. Hasil ini diperoleh dari jumlah rule yang tidak terdapat pada data uji berjumlah 4 maka hasilnya (4/9)*100= 44.44%, oleh karena itu *rule-rule* tersebut memiliki akurasi 100% - 44,44% = 55,56%.

3.5 Perancangan Antarmuka



Gambar 3. 25 Perancangan Antarmuka

Keterangan gambar 3.21:

1. *Textbox* yang digunakan untuk menginputkan transaksi.
2. Tombol *browse* yang digunakan untuk mencari data yang akan diinputkan ke dalam sistem.
3. Tombol ok untuk memastikan bahwa proses ini telah dilakukan.
4. *Textboxnumeric* untuk input *minimum confidence*.
5. *Textboxnumeric* untuk input *minimum support*.
6. Tombol ok untuk memastikan bahwa proses ini telah dilakukan.
7. *Textboxnumeric* untuk input *k-partisi*.
8. Tombol ok untuk memastikan bahwa proses ini telah dilakukan.
9. *Textbox* yang digunakan untuk input data uji.
10. Tombol *browse* yang digunakan untuk mencari data yang akan diinputkan ke dalam sistem.

11. Tombol simpan yang berfungsi untuk menyimpan hasil pada tahapan ini.
12. Tampilan untuk melihat hasil akurasi daru *rule*.
13. Tombol proses untuk menjalankan proses pada tahapan ini.
14. Tampilan *datagridview*, yang menampilkan *rule* beserta dengan nilai tingkat kekuatan dari *rule-rule* tersebut.
15. Tampilan jumlah baris pada *datagridview*.
16. Tombol bata untuk membatalkan proses.
17. Tombol simpan yang berfungsi untuk menyimpan hasil dari proses pada tahapan M-DHP.

3.6 Rancangan Evaluasi Rule

Pada sub bab ini akan dijelaskan mengenai perancangan uji coba rule yang dihasilkan sistem ini melalui pengujian jumlah partisi dan akurasi berdasarkan nilai *conviction* dan *hiper lift*.

Tabel 3. 21 Pengujian Rata-Rata Conviction dan HLR serta Jumlah Rule

Min. Confidence	**Min. Support**	**k-partisi**	**Jml. *Rule***	**Rata-rata *Conviction***	**Rata-rata *HLR***

Tabel 3. 22 Pengujian Conviction, HLR dan Confidence Terhadap Akurasi Rule

Min. *confidence*	**Min. *Support***	**k-partisi**	**Akurasi (%)**

Keterangan:

Minimum Support : Nilai *minimum support* yang dinginkan untuk evaluasi rule pada sistem

Minimum Confident : Nilai *minimum confident* yang dinginkan untuk evaluasi rule pada sistem

k-partisi : Partisi frequent 1-itemset yang dinginkan.

Jumlah Rule : Merupakan jumlah dari *rule-rule* yang dihasilkan dari sistem ini.

Rata-rata Conviction : Rata-rata nilai *conviction* yang dihasilkan oleh rule.

Rata-rata Hiper Lift : Rata-rata nilai *hiper lift* yang dihasilkan oleh rule.

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Bab ini meliputi implementasi dari perancangan sistem yang telah dibahas pada bab-bab sebelumnya beserta penjelasan dan pembahasan mengenai hasil uji coba, evaluasi dan analisanya.

4.1. Lingkungan Implementasi

Implementasi perangkat lunak ini berupa sistem aplikasi yang menerapkan metode *Multipass Direct Hashing and Prunning* (M-DHP) untuk menemukan asosiasi atau keterkaitan topik dalam ayat Al-Qur'an. Adapun untuk lingkungan implementasi yang akan disebutkan disini meliputi lingkungan perangkat keras dan lingkungan perangkat lunak.

4.1.1. Lingkungan implementasi perangkat keras

Perangkat keras (*Hardware*) yang digunakan dalam pengembangan sistem pencarian *asosiasi* topik dalam ayat al-Qur'an menggunakan metode *Multipass Direct Hashing and Prunning* M-DHP ini adalah:

1. *Prosesor Intel Pentium Dual-Core @2.3 GHz*
2. *RAM 2 GB*
3. *Hardisk 250 GB*
4. *Monitor*
5. *Keyboard*

4.1.2. Lingkungan implementasi perangkat lunak

Perangkat lunak (*Software*) yang digunakan dalam pengembangan sistem pencarian *asosiasi* topik *dalam ayat al-Qur'an* ini adalah:

1. Sistem Operasi Microsoft Windows 7
2. Microsoft Visual C# 2010 *Express Edition* sebagai *software development* dalam implementasi perancangan sistem.

3. Notepad dan *Microsoft office excel* 2007 sebagai media penyimpanan data.

4.2. Implementasi Program

Berdasarkan perancangan dan analisa sistem pada BAB III, proses-proses pada sistem pencarian keterkaitan atau *asosiasi* topik dalam *ayat al-Qur'an* ini terbagi menjadi dua bagian utama yaitu *text preprocessing* dan *Multipass Direct Hashing and Prunning* (M-DHP). Adapun pada sub bab dibawah ini akan dijelaskan implementasi tahapan-tahapan tersebut.

4.2.1. Implementasi *Preprocessing*

Tahapan *Preprocessing* diterapakan di kelas *Preprocessing* pada program, dimana kelas ini nantinya akan menghasilkan data transaksi yang siap diolah pada proses-proses lanjutan dalam sistem ini.

Pada kelas *Preprocessing* terdapat beberapa tahapan. Adapun tahapannya diawali dengan input data-data topik, dimana inputan berupa Array string yang berasal dari data topik pada file notepad. Untuk tahapan selanjutnya yaitu pencocokan input data topik dengan daftar topik pada sistem, melakukan proses pemilahan kata menggunakan *method Tokenizing* seperti yang ditunjukkan pada baris 30, dan penyusunan data transaksi dengan hasil akhir berupa *return value* List Transaksi. Proses *preprocessing* pada kelas *Preprocess* akan direpresen-tasikan pada *sourcecode* 4.1.

```
1.  public class Preprocessing
2.  {
3.    List<Transaksi> dataTransaksi = new List<Transaksi>();
4.    public List<Transaksi> ReadTextFile(string[] inputData)
5.    {
6.      List<string> kumpulan = new List<string>();
7.      Transaksi transaksi = new Transaksi();
8.      dataTransaksi = new List<Transaksi>();
9.      for (int i = 1; i <= 114; i++){
10.       transaksi = new Transaksi();
11.       transaksi.TID = i.ToString();
12.       dataTransaksi.Add(transaksi);
13.     }
14.       foreach (string item in inputData)
15.       {
16.         string filePath = @item;
17.         string line;
18.         if (File.Exists(filePath))
19.         {
20.          StreamReader file = null;
21.          try{
```

```
22.              kumpulan = new List<string>();
23.              file = new StreamReader(filePath);
24.              string nama =
25.                              Path.GetFileNameWithoutExtension(filePat
26.                              h);
27.              int noTopik =DatabaseTopik.database(nama.ToLower());
28.              if (noTopik>0){
29.              while ((line = file.ReadLine()) != null){
30.               List<string> input = Tokenizing.Tokenize(line);
31.               if (input.Count != 0){
32.                   int count = 0;
33.                   do{
34.                     if (kumpulan.Count == 0 ||
35.                                           !kumpulan.Contains(input[
36.                                           count])){
37.                kumpulan.Add(input[count]);
38.              }
39.                 count++;
40.            } while (count < input.Count);
41.          }
42.       }
43.       string con = "";
44.       for (int i = 0; i < kumpulan.Count; i++){
45.         transaksi = new Transaksi();
46.         if (kumpulan[i] != ""){
47.           int index = Convert.ToInt32(kumpulan[i]);
48.           con = dataTransaksi[index - 1].ITEMSET;
49.           con = String.Concat(con, noTopik.ToString());
50.           con = String.Concat(con, " ");
51.           dataTransaksi[index - 1].ITEMSET = con;
52.         }
53.        }
54.     }
55.   }
56.    finally{
57.       if (file != null)
58.          file.Close();
59.      }
60.     }
61.    }
62.   return dataTransaksi;
63.   }
64.  }
```

Sourcecode 4. 1 Kelas *Preprocessing*

4.2.1.1 Tokenizing

Pada *method Tokenize* ini melakukan suatu proses untuk menguraikan atau pemilahan kata dokumen input topik(*.txt) yang belum tertata menjadi sekumpulan kata tunggal atau daftar nomor surat, dimana hasil dari proses ini berfungsi untuk membentuk data transaksi. Proses pada *method Tokenize* ini diawali dengan memanggil fungsi *Regex.Replace()*, yang berfungsi untuk mereplace seluruh karakter selain angka 0-9 yang bertanda khusus yaitu titik dua (":") atau [0-9:] dengan spasi. Lalu menggunakan fungsi *Split()* yang berarti jika

ditemukan spasi, maka akan langsung dilakukan pemotongan terhadap string tersebut dan kemudian dibentuk menjadi sebuah array.

Hasil tahapan tersebut masih belum selesai, adapun proses selanjutnya yaitu mengambil angka 0-9 didepan tanda titik dua (" : ") dengan memanggil fungsi Substring. Hasil akhir dari proses tokenizing yaitu berupa *return value* List String yang berisi daftar nomor surat Al-Qur'an pada suatu topik, dimana nantinya dibentuk menjadi suatu data transaksi. Fungsi *Tokenize()* ditunjukkan pada Sourcecode 4.2.

```
1.  public static List<string> Tokenize(string dokumen)
2.  {
3.     List<string> daftarNoSurat = new List<string>();
4.     var clearDokumen = Regex.Replace(dokumen, "[^0-9:]", " ");
5.       string[] regexResult = clearDokumen.Split(new[] { " " },
6.       StringSplitOptions.RemoveEmptyEntries).ToArray();
7.       if (regexResult.Length != 0)
8.       {
9.         int count = 1;
10.        do
11.        {
12.         string noSurat = regexResult[count];
13.         int TitikDua = noSurat.IndexOf(":");
14.         if (TitikDua != -1)
15.         {
16.             noSurat = noSurat.Substring(0, TitikDua);
17.         }
18.          if (daftarNoSurat.Count == 0 ||
19.          !daftarNoSurat.Contains(noSurat))
20.           {
21.              daftarNoSurat.Add(noSurat);
22.           }
23.              count++;
24.         } while (count < regexResult.Length);
25.       }
26.    return daftarNoSurat;
27. }
```

Sourcecode 4. 2 Kelas *Itemset*

4.2.2. Implementasi Multipass Direct Hashing and Prunning *(M-DHP)*

Pada proses implementasi sistem untuk menemukan *asosiasi* topik dalam ayat Al-Qur'an menggunakan metode M-DHP ini terdapat beberapa tahapan. Diawali dengan tahapan DHP bagian 1, Partisi, DHP bagian 2, DHP bagian 3 dan proses *Generate Rule*.

4.2.2.1 DHP bagian Pertama

DHP bagian pertama ini akan menghasilkan kandidat *frequent k itemset* dimana saat ini nilai k=1 dan kandidat *k+1-itemset* yang disimpan di dalam *Hash Table* (H_2). Prosesnya diawali dengan mencari kandidat *k-itemset* dengan inputan berupa List data transaksi dan *minimum support* seperti yang ditunjukkan pada baris 3-8, kemudian menemukan jumlah *support* dari tiap kandidat *k-itemset* yang telah ditemukan tersebut *(Counting Support)* dengan menggunakan struktur data bentukan yaitu *hash tree* dan membentuk *k+1-itemset* yang disimpan di dalam *Hash Table* (H_2), adapun proses ini ditunjukkan pada baris 9-50. Hasil akhir pada tahapan ini adalah H_2 dan *frequent 1 itemset*, dimana diperoleh dari *k-itemset* yang memenuhi syarat minimum dari *minimum support* dan hasilnya disimpan pada variable *frequent_1_itemset*. Proses DHP bagian Pertama direpresentasikan pada *sourcecode* 4.3.

```
1.  public static void DHP1(List<Transaksi> param_data_transaksi
2.        double minimumSupport)
3.  {
4.    Itemset itemPindah;
5.    int jumlahTransaksi = awal_data_transaksi.Count;
6.
7.            counting_support_candidate_itemset_hashTree(Generate_Ko
8.            mbinasi_Candidate_Itemset(null, 1,
9.            param_data_transaksi), 1);
10.
11.   List<Itemset> candidate_1_itemset = new List<Itemset>();
12.   candidate_1_itemset = list_candidate_itemset;
14.
15.   frequent_1_itemset = frequent_Itemset(candidate_1_itemset,
16.                        getMin_Support(minSupport,
17.                        jumlahTransaksi));
18.
19.     for (int i = 0; i < frequent_1_itemset.Count; i++)
20.     {
21.       itemPindah = new Itemset();
22.       itemPindah.ITEMSET = frequent_1_itemset[i].ITEMSET;
23.       itemPindah.JUMLAH_SUPPORT =
24.       frequent_1_itemset[i].JUMLAH_SUPPORT;
25.       all_frequent_k_itemset.Add(itemPindah);
26.     }
27.
28.     Generate_Itemset_HashTable(1 + 1, param_data_transaksi,
29.                                null);
30.     List<Itemset> item_2Itemset = hasil_itemset_HashTable;
31.     Itemset items;
32.     string[] itemPecah;
33.     for (int i = 0; i < item_2Itemset.Count; i++)
34.     {
35.       items = new Itemset();
36.       itemPecah = item_2Itemset[i].ITEMSET.Split(';');
37.       for (int x = 0; x < itemPecah.Length - 1; x++)
38.       {
39.         string[] itemPecahLagi;
```

```
40.          itemPecahLagi =
41.                          System.Text.RegularExpressions.Regex.Split
42.                          (itemPecah[x], @"\W");
43.          int HashFunction = getHashFunctiont(1, 10, itemPecahLagi,
44.                                              17);
45.          List<string> data = new List<string>();
46.          if (H2.ContainsKey(HashFunction) == true)
47.          {
48.            List<String> tempHash = (List<string>)H2[HashFunction];
49.            items.ITEMSET = itemPecah[x];
50.            tempHash.Add(items.ITEMSET);
51.            H2[HashFunction] = tempHash;
52.           }
53.            else{
54.             data = new List<string>();
55.             items.ITEMSET = itemPecah[x];
56.             data.Add(items.ITEMSET);
57.             H2[HashFunction] = data;
58.           }
59.         }
60.      }
61.  }
```

Sourcecode 4. 3 DHP bagian 1

4.2.2.2 Partisi

Pada proses partisi, inputan data diperoleh dari proses DHP bagian Pertama yang berupa *frequent 1 itemset* (L_1). Tahapan selanjutnya adalah men-*shorting frequent 1 itemset* secara *ascending*, kemudian melakukan partisi sesuai dengan jumlah partisi yang diinginkan seperti yang ditunjukkan pada baris 4-18 dan menghasilkan *return value* List *Itemset frequent 1 itemset* yang telah ter-partisi (LP). Proses Partisi direpresentasikan pada *sourcecode* 4.4.

```
1.   public static List<List<Itemset>> partisi(List<Itemset>
2.          input_freq1_itemset, int n_partisi)
3.   {
4.     input_freq1_itemset = input_freq1_itemset.OrderBy(x =>
5.                           x.JUMLAH_SUPPORT).ToList();
6.     List<List<Itemset>> partisi_freq1_itemset = new
7.                                                 List<List<Itemset>>
8.                                                 ();
9.    for (int i = 0; i < n_partisi; i++)
10.   {
11.     if (i == n_partisi - 1)
12.     {
13.       var temp = input_freq1_itemset.GetRange(i,
14.                  input_freq1_itemset.Count - (n_partisi - 1));
15.       partisi_freq1_itemset.Add(temp);
16.     }
17.     else{
18.      var temp = input_freq1_itemset.GetRange(i, 1);
19.      partisi_freq1_itemset.Add(temp);
20.     }}
21.   return partisi_freq1_itemset;
22.  }
```

Sourcecode 4. 4 Partisi

4.2.2.3 DHP bagian Kedua

DHP bagian kedua ini memerlukan inputan data H_k yang berupa *Hash Table* dari proses DHP 1 dan *frequent 1 itemset* yang telah ter-partisi dari proses partisi. Terdapat beberapa tahapan pada proses ini, diawali dengan menemukan kandidat *itemset* menggunakan *method Gen_Candidate* yang ditunjukkan pada baris 26, melakukan *counting support* atau pencarian jumlah *support* tiap *itemset* sekaligus melakukan reduksi data transaksi yang ditunjukkan pada baris 28-31, membentuk k+1-*itemset* yang disimpan pada *hash table* (H_{k+1}) dan reduksi data transaksi kedua kalinya, dimana proses ini ditunjukkan pada baris 32-42 . Output dari proses ini adalah *frequent k itemset* (L_k) yang berupa List *Itemset*, adapun untuk proses DHP bagian kedua direpresentasikan pada *sourcecode* 4.5.

```
1.  public static void DHP2(Hashtable param_H2, List<List<Itemset>>
2.        param_hasil_partisi, List<Transaksi> param_data_transaksi)
3.  {
4.    Itemset itemPindah;
5.    k = 2;
6.    p = (param_hasil_partisi.Count) - 1;
7.    int Large = 0;
8.    int jumlahTransaksi = awal_data_transaksi.Count;
9.    if (k == 2){
10.    hashTable_k_itemset = param_H2;
11.   }
12.   while (Get_HashTable_minSupport(hashTable_k_itemset,
14.   jumlahTransaksi, minSupport) >= Large)
15.    {
16.      List<Itemset> tempPartisiByIndex;
17.      if (p >= 0){
18.       tempPartisiByIndex = new List<Itemset>();
19.       tempPartisiByIndex = param_hasil_partisi[p];
20.      }
21.      else{
22.        tempPartisiByIndex = new List<Itemset>();
23.        tempPartisiByIndex = null;
24.      }
25.
26.   Gen_Candidate(tempPartisiByIndex, hashTable_k_itemset,
27.           frequent_k_itemset, k, jumlahTransaksi);
28.   Count_Support(param_data_transaksi, k, list_candidate_hashTree,
29.           hasil_itemset_HashTable);
30.  data_transaksi = new List<Transaksi>();
31.  data_transaksi = data_transaksi_count_support;
32.
33.  if (data_transaksi_count_support.Count > k){
34.   Make_Hash(hashTable_k_itemset, k,
35.             data_transaksi_count_support,
36.             temp_hasil_itemset_HashTable);
37.
38.  if (data_transaksi_make_hash.Count > k){
39.   data_transaksi = new List<Transaksi>();
40.   data_transaksi = data_transaksi_make_hash;
41.  }
42. }
43. else{
```

```
44.     hashTable_k_itemset = null;
45.   }
46.     frequent_k_itemset = new List<Itemset>();
47.     frequent_k_itemset = frequent_Itemset(list_candidate_hashTree,
48.                          getMin_Support(minSupport,
49.                          jumlahTransaksi));
50.     for (int i = 0; i < frequent_k_itemset.Count; i++){
51.      itemPindah = new Itemset();
52.      itemPindah.ITEMSET = frequent_k_itemset[i].ITEMSET;
53.      itemPindah.JUMLAH_SUPPORT =
54.                                  frequent_k_itemset[i].JUMLAH_SUPPORT;
55.      all_frequent_k_itemset.Add(itemPindah);
56.     }
57.     if (p > -1){
58.       p--;
59.     }
60.    k++;
61.    Large = k;
62.    }
63. }
```

Sourcecode 4. 5 DHP bagian Kedua

4.2.2.3.1 Gen Candidate

Proses *Gen Candidate* ini berfungsi untuk menemukan kandidat *k-itemset* dan diimplementasikan pada *method* `Gen_Candidate()`. Adapun data input yang diperlukan yaitu *frequent k itemset* (L_k) yang berupa List *Itemset*, H_k atau *Hash Table* dan *frequent 1 itemset* yang telah ter-partisi (LP). Tahapannya diawali dengan melakukan *join* antara L_k dengan LP untuk membentuk *itemset*, adapun prosesnya seperti yang ditunjukkan pada baris 8-33. Proses selanjutnya melakukan pengecekan kandidat *itemset* yang telah terbentuk tersebut pada *hash table* (H_k), dimana dengan syarat jika *item* yang terbentuk memiliki jumlah pada H_k lebih dari sama dengan nilai *minimum support*, maka *itemset* tersebut menjadi kandidat *itemset*. Untuk hasil akhir pada tahapan ini adalah kandidat *k-itemset*, dimana *k-itemset* tersebut disimpan pada struktur data *hash tree* seperti yang ditunjukkan pada baris 48-52. *Gen Candidate* direpresentasikan pada *sorcecode* 4.6.

```
1.  private static void Gen_Candidate(List<Itemset> Lp, Hashtable Hk,
2.          List<Itemset> Lk_1, int k, int jumlahTransaksi)
3.  {
4.    List<Itemset> tempCk = new List<Itemset>();
5.    list_candidate_hashTree = new List<Itemset>();
6.    Itemset items;
7.    hashTree = new HashTree<Itemset>();
8.    if (k == 2){
9.     tempCk = new List<Itemset>();
10.    tempCk = Generate_Kombinasi_Candidate_Itemset(Lp, k, null);
11.   }
```

```
12.    else if (k > 2){
14.      if (Hk != null && Lk_1 != null){
15.         if (Lp != null){
16.         for (int i = 0; i < Lk_1.Count; i++){
17.           items = new Itemset();
18.           items.ITEMSET = Lk_1[i].ITEMSET;
19.           items.JUMLAH_SUPPORT = Lk_1[i].JUMLAH_SUPPORT;
20.           Lp.Add(items);
21.          }
22.       }
23.       else{
24.        Lp = new List<Itemset>();
25.        for (int i = 0; i < Lk_1.Count; i++){
26.          items = new Itemset();
27.          items.ITEMSET = Lk_1[i].ITEMSET;
28.          items.JUMLAH_SUPPORT = Lk_1[i].JUMLAH_SUPPORT;
29.          Lp.Add(items);
30.        }
31.       }
32.       tempCk = new List<Itemset>();
33.       tempCk = Generate_Kombinasi_Candidate_Itemset(Lp, k, null);
34.      }
35.   }
36.    if (tempCk.Count > 0){
37.    for (int j = 0; j < tempCk.Count; j++){
38.      string[] itemPecahLagi;
39.      itemPecahLagi =
40. System.Text.RegularExpressions.Regex.Split(tempCk[j].ITEMSET,
41. @"\W");
42.      int HashFunction = getHashFunctiont(k - 1, 10, itemPecahLagi,
43. 17);
44.       List<string> tempHk = (List<string>)Hk[HashFunction];
45.       if (tempHk != null){
46.         int jumlahHashTabel = tempHk.Count;
47.         if (jumlahHashTabel >= getMin_Support(minSupport,
48.             jumlahTransaksi)){
49.            items = new Itemset();
50.            items.ITEMSET = tempCk[j].ITEMSET;
51.            hashTree.Add(items, k);
52.
53.            }
54.          }
55.        }
56.      list_candidate_hashTree = hashTree.getLeavesNodes(hashTree);
57.     }
58.     else{
59.         list_candidate_hashTree = null;
60. }
```

Sourcecode 4. 6 Gen Candidate

4.2.2.3.2 Counting Support

Tahapan *Counting Support* diimplementasikan pada *method* `Count_Support()`. Pada penerapnnya tahapan ini memiliki dua proses penting. Pertama, menemukan jumlah *support* dengan menggunakan *hash tree* yang pada *method* `counting_support_candidate_itemset_hashTree()`, dimana proses ini ditunjukkan pada baris 32-39. Proses kedua adalah melakukan reduksi data

transaksi yang tidak diperlukan pada tahapan selanjutnya, seperti yang ditunjukkan pada baris 42-55. Untuk *method Counting Support* direpresentasikan pada *sourcecode* 4.7.

```
1.  private static void Count_Support(List<Transaksi>
2.          param_data_transaksi, int k_itemset, List<Itemset>
3.          param_list_candidate_hashTree, List<Itemset>
4.          param_hasil_itemset_HashTable)
5.  {
6.    Itemset itemCount_support;
7.    Transaksi transaksiItem;
8.    string[] hasil_itemset_HashTablePecah;
9.    temp_hasil_itemset_HashTable = new List<Itemset>();
10.   data_transaksi_count_support = new List<Transaksi>();
11.   List<Itemset> list_temporary;
12.   List<Transaksi> list_temp_trnsaksi;
13.   string[] tempGabungItemset = new string[1];
14.   string[] tempGabungTransakai = new string[1];
15.
16.   for (int i = 0; i < param_hasil_itemset_HashTable.Count; i++){
17.     itemCount_support = new Itemset();
18.     list_temporary = new List<Itemset>();
19.     list_temp_trnsaksi = new List<Transaksi>();
20.     hasil_itemset_HashTablePecah=
21.                                   param_hasil_itemset_HashTable
22.                                   [i].ITEMSET.Split(';');
23.       List<string> hasil = new List<string>();
24.       List<int> ihasil = new List<int>();
25.       For (int j=0; j <hasil_itemset_HashTablePecah.Length-1; j++){
26.           List<Itemset> temp_counting_support;
27.           string cari = hasil_itemset_HashTablePecah[j];
28.           if (param_list_candidate_hashTree != null){
29.               bool kondisi = false;
30.               foreach (var temp_list_candidate_hashTree in
31.                           param_list_candidate_hashTree)
32.                   if (temp_list_candidate_hashTree.
33.                       ITEMSET.Equals(cari)){
34.
35.                       //1. melakukan count suport
36.                       itemCount_support = new Itemset();
37.                       itemCount_support.ITEMSET = cari;
38.                       temp_counting_support = new List<Itemset>();
39.                       temp_counting_support.Add(itemCount_support);
40.                       counting_support_candidate_itemset_hashTree(te
41.                       mp_counting_support, k_itemset
42.
43.                       //2. reduksi database
44.                       string[] kataPecahItemset;
45.                       kataPecahItemset =
46.                                           System.Text.RegularExpress
47.                                           ions.Regex.Split(cari,
48.                                           @"\W");
49.           for (int y = 0; y < kataPecahItemset.Length; y++){
50.               string cariReduksi = kataPecahItemset[y];
51.               if (hasil.Count == 0 ||
52.                   !hasil.Contains(cariReduksi)){
53.       hasil.Add(cariReduksi);
54.       ihasil.Add(1);
55.  }else if (hasil.Contains(cariReduksi)){
56.    ihasil[hasil.IndexOf(cariReduksi)] += 1;
57.    }
58.   }
```

```
59.     list_temporary.Add(itemCount_support);
60.     kondisi = true;
61.     if (kondisi == true)
62.        break;
63.         }
64.        }
65.     }
66.
67.    for (int h = 0; h < hasil.Count; h++){
68.     if (ihasil[h] >= k){
69.       transaksiItem = new Transaksi();
70.       transaksiItem.ITEMSET = hasil[h];
71.       list_temp_trnsaksi.Add(transaksiItem);
72.       }
73.      }
74.       tempGabungTransakai[0] = "";
75.       for (int x = 0; x < list_temp_trnsaksi.Count; x++){
76.       tempGabungTransakai[0] += list_temp_trnsaksi[x].ITEMSET + "
77.                                 ";
78.       }
79.        if (tempGabungTransakai[0] != ""){
80.         transaksiItem = new Transaksi();
81.         transaksiItem.ITEMSET = tempGabungTransakai[0].Trim();
82.         transaksiItem.TID = param_hasil_itemset_HashTable[i].TID;
83.         data_transaksi_count_support.Add(transaksiItem);
84.         }
85.        tempGabungItemset[0] = "";
86.        for (int x = 0; x < list_temporary.Count; x++){
87.         tempGabungItemset[0] += list_temporary[x].ITEMSET + ";";
88.         }
89.          itemCount_support = new Itemset();
90.          itemCount_support.ITEMSET = tempGabungItemset[0].Trim();
91.          itemCount_support.TID =
92.                                  param_hasil_itemset_HashTable[i].TI
93.                                  D;
94.         temp_hasil_itemset_HashTable.Add(itemCount_support);
95.         }
96.      list_candidate_hashTree = new List<Itemset>();
97.      list_candidate_hashTree = hashTree.getLeavesNodes(hashTree);
98. }
```

Sourcecode 4. 7 *Count Support*

4.2.2.3.3 Make Hash

Tahapan *Make Hash* diimplementasikan pada *method* `Make_Hash()`. Pada tahapan ini membutuhkan memiliki dua proses penting. Pertama, membentuk *k+1-itemset* yang hasilnya akan disimpan pada *hash table* (H_{k+1}) yang berfungsi untuk mengecek apakah nantinya *k+1-itemset* yang terbentuk layak menjadi kandidat *itemset* atau tidak, adapun seperti yang ditunjukkan pada baris 16-45. Proses kedua adalah melakukan reduksi data transaksi yang tidak diperlukan pada tahapan selanjutnya dan ditunjukkan pada baris 50-70. Hasil akhir pada tahapan ini adalah *k+1 itemset* yang disimpan pada *Hash Table* hashTable_k_itemset dan

data transaksi yang telah tereduksi. Untuk *method Make Hash* direpresentasikan pada *sourcecode* 4.8.

```
private static void Make_Hash(Hashtable param_Hk, int param_k,
        List<Transaksi> param_data_trnsaksi, List<Itemset>
        param_temp_hasil_itemset_HashTable)
{
   Itemset items;
   Transaksi transaksiItem;
   hashTable_k_itemset = new Hashtable();
   data_transaksi_make_hash = new List<Transaksi>();
   List<Transaksi> list_temp_trnsaksi;
   string[] tempGabungTransakai = new string[1];
   int k_hashTable = param_k + 1;
   int jumlahTransaksi = awal_data_transaksi.Count;
   Generate_Itemset_HashTable(k_hashTable, null,
  param_temp_hasil_itemset_HashTable);
for (int i = 0; i < hasil_itemset_HashTable.Count; i++){
    string[] pecah;
    pecah = hasil_itemset_HashTable[i].ITEMSET.Split(';');
    List<string> hasil = new List<string>();
    List<int> ihasil = new List<int>();
    list_temp_trnsaksi = new List<Transaksi>();
    for (int j = 0; j < pecah.Length - 1; j++){
      List<Itemset> temp_k_minus1_subset =
                                        Generate_Kminus1_itemset(pecah
                                        [j]);
      items = new Itemset();
      if (Cek_Kminus1_support(temp_k_minus1_subset, param_Hk,
          jumlahTransaksi) == true){
            string[] itemPecahLagi;
            itemPecahLagi =
System.Text.RegularExpressions.Regex.Split(pecah[j], @"\W");
            int HashFunction = getHashFunctiont(param_k, 10,
itemPecahLagi, 17);
            List<string> data = new List<string>();
     if (hashTable_k_itemset.ContainsKey(HashFunction) == true) {
       List<String> tempHash =
                                (List<string>)hashTable_k_itemset[HashFun
                                ction];
        items.ITEMSET = pecah[j];
        tempHash.Add(items.ITEMSET);
        hashTable_k_itemset[HashFunction] = tempHash;
       }
        else{
          data = new List<string>();
          items.ITEMSET = pecah[j];
          data.Add(items.ITEMSET);
          hashTable_k_itemset[HashFunction] = data;
        }
       string[] kataPecahItemset;
       kataPecahItemset =
System.Text.RegularExpressions.Regex.Split(pecah[j], @"\W");
       for (int y = 0; y < kataPecahItemset.Length; y++){
         string cariReduksi = kataPecahItemset[y];
         if (hasil.Count == 0 || !hasil.Contains(cariReduksi)) {
            hasil.Add(cariReduksi);
            ihasil.Add(1);
        }
                          else if (hasil.Contains(cariReduksi))
                          {
```

```
                            ihasil[hasil.IndexOf(cariReduksi)] += 1;
                        }
                    }
                }
            }
            for (int h = 0; h < hasil.Count; h++)
            {
                if (ihasil[h] > 0)
                {
                    transaksiItem = new Transaksi();
                    transaksiItem.ITEMSET = hasil[h];
                    list_temp_trnsaksi.Add(transaksiItem);
                }
            }
            tempGabungTransakai[0] = "";
            for (int x = 0; x < list_temp_trnsaksi.Count; x++)
            {
             tempGabungTransakai[0] +=
              list_temp_trnsaksi[x].ITEMSET + " ";
            }
            transaksiItem = new Transaksi();

            if (tempGabungTransakai[0].Trim() != "")
            {
              transaksiItem.ITEMSET = tempGabungTransakai[0].Trim();
                transaksiItem.TID = hasil_itemset_HashTable[i].TID;
                 data_transaksi_make_hash.Add(transaksiItem);
            }
        }
    }
```

Sourcecode 4. 8 Make Hash

4.2.2.4 DHP bagian Ketiga

DHP bagian ketiga ini diiimplementasikan pada *method* `DHP3()`. Untuk input pada proses ini antara lain *Hash Table k-itemset*, *List Itemset*, *List frequent k-itemset*, dan data transaksi. Terdapat beberapa tahapan pada proses ini, diawali dengan menemukan kandidat *itemset* menggunakan *method* `Gen_Candidate()` namun hanya digunakan sekali saja, melakukan *counting support* atau pencarian jumlah *support* tiap *itemset* sekaligus melakukan reduksi data transaksi pada *method* `Count_Support()`, dimana proses untuk kedua *method* tersebut telah dijelaskan pada DHP bagian Kedua, dan untuk proses membentuk kandidat *k+1-itemset* selanjutnya akan menggunakan *method* `Apriori_Gen()`, hal inilah yang membedakan dengan DHP bagian ketiga dengan DHP bagian kedua, adapun untuk proses DHP bagian ketiga direpresentasikan pada *sourcecode* 4.9.

```
public static void DHP3(Hashtable param_hashTable_k_itemset,
        List<List<Itemset>> param_hasil_partisi, List<Transaksi>
        param_data_transaksi, List<Itemset>
        input_frequent_k_itemset)
```

```
5.    {
6.      Itemset itemPindah;
7.              List<Itemset> tempPartisiByIndex;
8.              if (p >= 0)
9.              {
10.                 tempPartisiByIndex = new List<Itemset>();
11.                 tempPartisiByIndex = param_hasil_partisi[p];
12.             }
13.             else
14.             {
15.                 tempPartisiByIndex = new List<Itemset>();
16.                 tempPartisiByIndex = null;
17.             }
18.             int jumlahTransaksi = awal_data_transaksi.Count;
19.             Gen_Candidate(tempPartisiByIndex,
20.             param_hashTable_k_itemset, input_frequent_k_itemset,
21.             k, jumlahTransaksi);
22.             while (list_candidate_hashTree != null)
23.             {
24.                 if (data_transaksi.Count > 0)
25.                 {
26.                    Count_Support(param_data_transaksi, k,
27.                    list_candidate_hashTree, hasil_itemset_HashTable);
28.                         if (data_transaksi_count_support.Count > k)
29.                         {
30.                           data_transaksi = new List<Transaksi>();
31.                          data_transaksi =
32.                           data_transaksi_count_support;
33.                         }}
34.                     frequent_k_itemset = new List<Itemset>();
35.                      frequent_k_itemset =
36.                                         frequent_Itemset(list_candidat
37.                                         e_hashTree,
38.                                         getMin_Support(minSupport,
39.                                         jumlahTransaksi));
40.
41.              for (int i = 0; i < frequent_k_itemset.Count; i++){
42.                    itemPindah = new Itemset();
43.                    itemPindah.ITEMSET =
44.                                         frequent_k_itemset[i].ITEMSET;
45.                        itemPindah.JUMLAH_SUPPORT =
46.                                                 frequent_k_itemset[i]
47.                                                 .JUMLAH_SUPPORT;
48.                        all_frequent_k_itemset.Add(itemPindah);
49.                    }
50.                    if (p > -1)
51.                    {
52.                        p--;
53.                    }
54.             List<Itemset> tempPartisiByIndexDalam;
55.             if (p >= 0) {
56.                    tempPartisiByIndexDalam = new List<Itemset>();
57.                    tempPartisiByIndexDalam = param_hasil_partisi[p];
58.                    }
59.                    else{
60.                        tempPartisiByIndexDalam = new
61.                                                 List<Itemset>();
62.                        tempPartisiByIndexDalam = null;
63.                    }
64.                    Apriori_Gen(tempPartisiByIndexDalam,
65.  frequent_k_itemset, k + 1);
66.                    k++;
67.             }}
```

Sourcecode 4. 9 DHP bagian ketiga

4.2.2.4.1 Apriori Gen

Pada method `Apriori_Gen()` ini digunakan untuk menemukan kandidat *itemset* pada tahapan DHP bagian ketiga dan ini yang memebedakan dengan DHP bagian kedua. Terdapat beberapa proses pada tahapan ini, diawali dengan inputan *frequent k itemset* (L_k) dan *frequent k itemset* yang ter-partisi (LP), kemudian melakukan *join* seperti yang ditunjukkan pada baris 10-43, sampai akhirnya menghasilkan kandidat *k+1-itemset* dimana hasil akhir pada tahapan ini disimpan pada List candidate *hash tree*. *Apriori Gen* direpresentasikan pada *sourcecode* 4.10.

```
1.  private static void Apriori_Gen(List<Itemset> Lp, List<Itemset>
2.          Lk_1, int k)
3.          {
4.              Itemset items;
5.              List<Itemset> tempCk = new List<Itemset>();
6.              Itemset itemAg;
7.              list_candidate_hashTree = new List<Itemset>();
8.
9.              if (k >= 2)
10.             {
11.                 if (Lk_1 != null)
12.                 {
13.                     if (Lp != null)
14.                     {
15.                         for (int i = 0; i < Lk_1.Count; i++)
16.                         {
17.                             items = new Itemset();
18.                             items.ITEMSET = Lk_1[i].ITEMSET;
19.                             items.JUMLAH_SUPPORT =
20.                                                 Lk_1[i].JUMLAH_SU
21.                                                 PPORT;
22.                             Lp.Add(items);
23.                         }
24.                     }
25.                     else
26.                     {
27.                         Lp = new List<Itemset>();
28.                         for (int i = 0; i < Lk_1.Count; i++)
29.                         {
30.                             items = new Itemset();
31.                             items.ITEMSET = Lk_1[i].ITEMSET;
32.                             items.JUMLAH_SUPPORT =
33.                                                 Lk_1[i].JUMLAH_SU
34.                                                 PPORT;
35.                             Lp.Add(items);
36.                         }
37.                     }
38.
39.                     tempCk = new List<Itemset>();
40.                     tempCk =
41.                             Generate_Kombinasi_Candidate_Itemset(L
42.                             p, k, null);
43.                 }
44.                 if (tempCk.Count > 0)
45.                 {
46.                     for (int j = 0; j < tempCk.Count; j++)
```

```
47.                     {
48.         List<Itemset> temp_k_minus1_subset =
49.                                             Generate_Kminus1_itemset(
50.                                             tempCk[j].ITEMSET);
51.         if (Cek_Kminus1_support_Apriori_Gen(temp_k_minus1_subset,
52.            Lk_1) == true) {
53.                                 itemAg = new Itemset();
54.                                 itemAg.ITEMSET = tempCk[j].ITEMSET;
55.                                 list_candidate_hashTree.Add(itemAg);
56.
57.                             }
58.                         }
59.                     }
60.                     else
61.                     {
62.                         list_candidate_hashTree = null;
63.                     }
64.                 }
65.             }
```

Sourcecode 4. 10 Apriori Gen

4.2.2.5 Generate Rules

Proses *Generate Rules* pada tahapan ini diimplementasikana pada *method* `Generate_rules()`. Metode yang digunakan untuk *generate rules* ini menggunakan *faster algorithm*, karena pada sistem ini membutuhkan *consequent* lebih dari satu item. Untuk input pada proses ini adalah *List itemset frequent k-itemset* dan data uji yang digunakan untuk mencari akurasi dari *rule* dengan proses yaitu melakukan pengecekan, untuk yang dibentuk menjadi *rule* merupakan *itemset* lebih dari 1-itemset, tahapan ini seperti yang ditunjukkan pada baris 14, apabila memenuhi syarat tersebut maka dilakukan ke proses selanjutnya pada *method Ap-Genrules*. Tahap akhit dari proses ini adalah menemukan akurasi daru seluruh *rule*. Proses *Generate Rules* direpresentasikan pada *sourcecode* 4.11.

```
1.  public static void Generate_rules(List<Itemset>
2.          param_frequent_k_itemset, List<Transaksi> dataUji)
3.  {
4.    List<Itemset> H_frequent_k_itemset = new List<Itemset>();
5.    List<Itemset> temp;
6.    Itemset itemRule;
7.    for (int i = 0; i < param_frequent_k_itemset.Count; i++){
8.          string[] pecah;
9.          pecah =
10.                 System.Text.RegularExpressions.Regex.Split(param_fre
11.                 quent_k_itemset[i].ITEMSET, @"\W");
12.         if (pecah.Length >= 2)
13.         {
14.                 H_frequent_k_itemset = new List<Itemset>();
15.                 for (int j = 0; j < pecah.Length; j++)
16.                 {
17.                         itemRule = new Itemset();
18.                             itemRule.ITEMSET = pecah[j];
```

```
19.                         H_frequent_k_itemset.Add(itemRule);
20.                     }
21.                     temp = new List<Itemset>();
22.                     itemRule = new Itemset();
23.                     itemRule.ITEMSET =
24.                                     param_frequent_k_itemset[i].I
25.                                     TEMSET;
26.                     itemRule.JUMLAH_SUPPORT =
27.                                         param_frequent_k_itemse
28.                                         t[i].JUMLAH_SUPPORT;
29.                     temp.Add(itemRule);
30.                     Ap_Genrules(temp, H_frequent_k_itemset, 0);
31.                 }
32.             }
33.         }
34.     akurasiTemp = akurasi(rule_uji, dataUji);
35.     }
```

Sourcecode 4. 11 Generate Rules

4.2.2.5.1 Ap-Genrules

Tahapan pada proses ini diimplementasikan pada *method* `Ap_Genrules()`. Pada method ini membutuhkan input berupa *frequent k itemset*. Secara umum proses pada tahapan ini adalah menemukan nilai *confidence* dari *rule* yang dihasilkan *frequent k-itemset* seperti yang ditunjukkan pada baris 30-59. Proses selanjutnya yaitu apabila *rule-rule* tersebut telah memenuhi syarat *minimum sconfidence* yang telah ditetapkan, maka kemudian menemukan *rule-rule* dengan tingkat kekuatan yang baik dimana dengan cara menemukan nilai *conviction* dan *hiper lift ratio* dari setiap *rule* tersebut, adapun proses ini ditunjukkan pada baris 61-100. Proses *Ap-Genrules* direpresentasikan pada *sourcecode* 4.12.

```
1.  private static void Ap_Genrules(List<Itemset>
2.         frequent_k_itemset_k_i, List<Itemset>
3.         param_H_frequent_k_itemset, int m, double minConf)
4.  {
5.              int kRule = 0;
6.              Itemset itemRules;
7.              List<Itemset> hasil_H = new List<Itemset>();
8.              int jumlahTrnsaksi = awal_data_transaksi.Count;
9.              string[] L_itemset;
10.             L_itemset =
11.                     System.Text.RegularExpressions.Regex.Split(f
12.                     requent_k_itemset_k_i[0].ITEMSET, @"\W");
13.             kRule = L_itemset.Length;
14.             if (kRule == 2)
15.             {
16.                 m = 0;
17.             }
18.             else
19.             {
20.                 m = 1;
21.             }
22.             while (k > m + 1)
23.             {
```

```
24.                 hasil_H = new List<Itemset>();
25.                hasil_H = Apriori_Gen_generate_Rules(null,
26.                param_H_frequent_k_itemset, kRule);
27.
28.                 for (int i = 0; i < hasil_H.Count; i++)
29.                 {
30.                    double support_1 =
31.                                        frequent_k_itemset_k_i[0].JUML
32.                                        AH_SUPPORT;
33.                     double support_2 = 0;
34.                     double support2_ConviHiper = 0;
35.                     string[] L_itemset_pengurang;
36.                    L_itemset_pengurang =
37.                                           System.Text.RegularExpress
38.                                           ions.Regex.Split(hasil_H[i
39.                                           ].ITEMSET, @"\W");
40.                    var hasil_kurang =
41.                                            L_itemset.Except(L_itemset_pe
42.                                           ngurang);
43.                    string hasilKurang = String.Join(" ",
44.                    hasil_kurang.ToArray());
45.                     bool kondisi1 = false;
46.                     foreach (var item in all_frequent_k_itemset)
47.                         if (item.ITEMSET.Equals(hasilKurang))
48.                         {
49.                             support_2 = item.JUMLAH_SUPPORT;
50.                             kondisi1 = true;
51.
52.                             if (kondisi1 == true)
53.                                 break;
54.                         }
55.
56.                     double confidence = 0;
57.                     confidence = (support_1 / support_2) * 100;
58.
59.                    if (confidence >=
60.                                        Get_minConfidence(minConfidenc
61.                                        e))
62.                     {
63.                         itemRules = new Itemset();
64.                         string a = Konversi(hasilKurang);
65.                         string b = Konversi(hasil_H[i].ITEMSET);
66.                         itemRules.RULE = a + "=>" + b;
67.                         itemRules.NILAI_CONFIDENT = confidence;
68.                          itemRules.NILAI_SUPPORT = (support_1 /
69.                        jumlahTrnsaksi) * 100;
70.                         bool kondisi2 = false;
71.                      foreach (var item in all_frequent_k_itemset)
72.               if (item.ITEMSET.Equals(hasil_H[i].ITEMSET)){
73.                 support2_ConviHiper = item.JUMLAH_SUPPORT;
74.                  kondisi2 = true;
75.                if (kondisi2 == true)
76.                   break;
77.                }
78.             double tempConviction = (1 - (confidence / 100));
79.            if (tempConviction == 0) {
80.              itemRules.NILAI_CONVICTION = Double.NaN;
81.             }
82.               else{
83.               itemRules.NILAI_CONVICTION = (1 - (support2_ConviHiper
84.               / jumlahTrnsaksi)) / (tempConviction);
85.               }
86.                itemRules.NILAI_HPL = (confidence / 100) / ((0.99) *
87.   (support2_ConviHiper / jumlahTrnsaksi));
```

```
88.                         if (((itemRules.NILAI_CONVICTION >= 0.5
89.                             && itemRules.NILAI_CONVICTION < 1) ||
90.                             (itemRules.NILAI_CONVICTION > 1) ||
91.                             itemRules.NILAI_CONVICTION.Equals(Dou
92.                             ble.NaN)) && (itemRules.NILAI_HPL >
93.                             1))
94.         {
95.           all_rule.Add(itemRules);
96.         }
97.         }
98.       }
99.      m++;
100.   param_H_frequent_k_itemset = hasil_H;
101.   }
102. }
```

Sourcecode 4. 12 Ap-Genrules

4.3. Implementasi Antarmuka

Impelementasi antarmuka dari sistem ini terdiri dari dua form bagian utama, yaitu:

1. Antarmuka *Preprocessing*

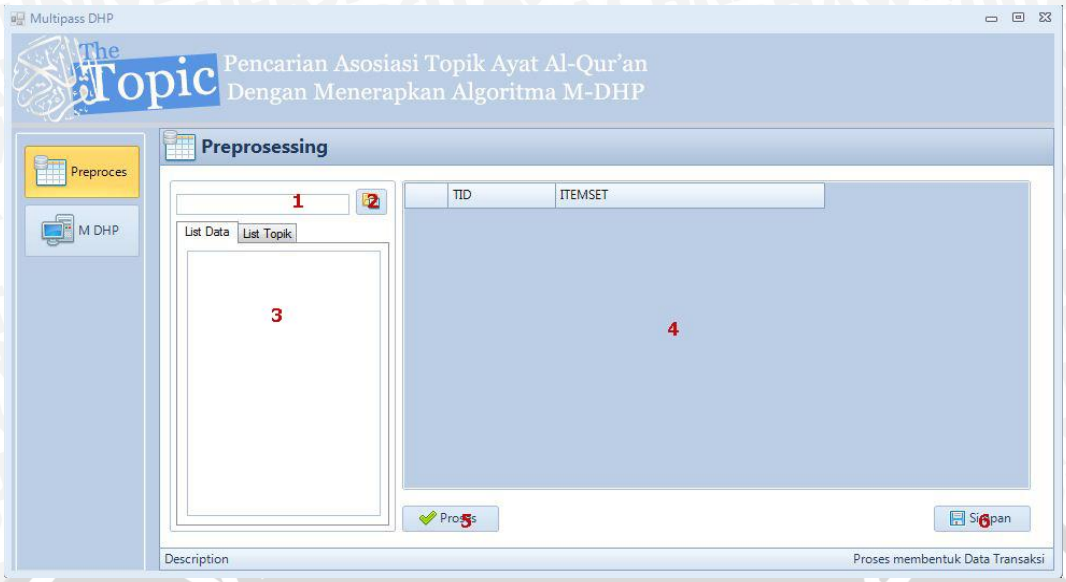
   Pada antarmuka *preprocessing* digunakan sebagai antarmuka untuk melakukan penyusunan atau pembentukan data transaksi yang akan digunakan pada sistem ini.

2. Antarmuka M-DHP

   Pada antarmuka M-DHP digunakan sebagai antarmuka untuk melakukan proses dari metode M-DHP dalam menemukan seluruh *rule*. Dimana pada form ini juga terdapat informasi tingkat akurasi *rule* pada sistem.

4.3.1 Antarmuka Preprocessing

Antarmuka untuk proses *preprocessing* ditunjukkan pada gambar 4.1, dimana tahap pertama user melakukan input dataset surat dan ayat al-qur'an pada topik. Setelah user memasukkan data tersebut, maka langkah selanjutnya menekan tombol *preprocessing* untuk menjalankan proses awal dari sistem. Apabila proses telah selesai maka hasil proses ini ditampilkan pada *datagridview* dan dapat disimpan dengan menekan tombol simpan. Hasil dari proses ini adalah data transaksi yang digunakan dalam tahapan berikutnya pada untuk menemukan *rule* menggunakan metode M-DHP.



Gambar 4.1 Antarmuka Tampilan untuk Proses Preprocessing

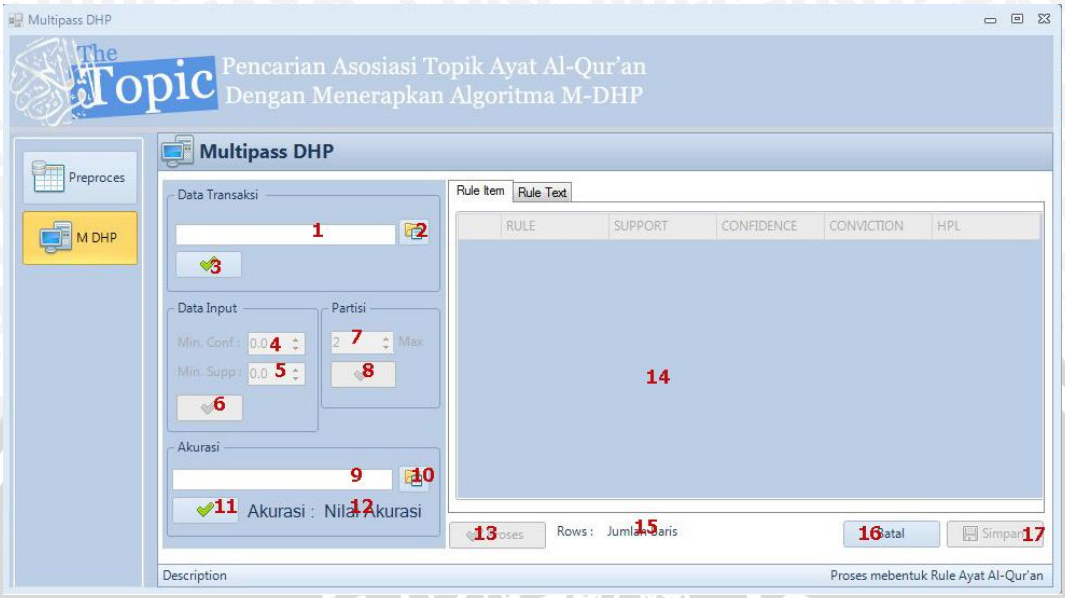
Keterangan gambar 4.1:

1. *Textbox* yang digunakan untuk menginputkan dataset surat dan ayat al-qur'an pada topik.
2. Tombol *browse* yang digunakan untuk mencari data yang akan diinputkan ke dalam system
3. Tab list data yang berisi path seluruh data dan list topik yanag berisi list selutuh topik.
4. Tampilan datagridview akan menampilkan hasil dari tahap *preprocessing* yang berbentuk data transaksi.
5. Tombol proses yang berfungsi untuk menjalankan proses pada tahapan *preprocessing* ini.
6. Tombol simpan yang berfungsi untuk menyimpan hasil pada tahapan ini.

4.3.2 Antamuka Proses M-DHP

Antarmuka untuk proses M-DHP ditunjukkan pada gambar 4.2, dimana tahap pertama user melakukan input dataset transaksi. Setelah user memasukkan data tersebut, maka langkah selanjutnya secara berturut-turut melakukan input *minimum confidence*, *minimum support*, *k-partisi*, dan data uji. Apabila proses telah selesai maka hasil proses ini ditampilkan pada *datagridview* dan dapat

disimpan dengan menekan tombol simpan. Hasil dari proses ini adalah *rule-rule* dengan nilai tingkat kekuatan dari *rule* tersebut beserta akurasinya.



Gambar 4.2 Antarmuka Tampilan untuk Proses M-DHP

Keterangan gambar 4.2:

1. *Textbox* yang digunakan untuk menginputkan transaksi.
2. Tombol *browse* yang digunakan untuk mencari data yang akan diinputkan ke dalam sistem.
3. Tombol ok untuk memastikan bahwa proses ini telah dilakukan.
4. *Textboxnumeric* untuk input *minimum confidence*.
5. *Textboxnumeric* untuk input *minimum support*.
6. Tombol ok untuk memastikan bahwa proses ini telah dilakukan.
7. *Textboxnumeric* untuk input *k-partisi*.
8. Tombol ok untuk memastikan bahwa proses ini telah dilakukan.
9. *Textbox* yang digunakan untuk input data uji.
10. Tombol *browse* yang digunakan untuk mencari data yang akan diinputkan ke dalam sistem.
11. Tombol simpan yang berfungsi untuk menyimpan hasil pada tahapan ini.
12. Tampilan untuk melihat hasil akurasi daru *rule*.
13. Tombol proses untuk menjalankan proses pada tahapan ini.

14. Tampilan *datagridview*, yang menampilkan *rule* beserta dengan nilai tingkat kekuatan dari *rule-rule* tersebut.
15. Tampilan jumlah baris pada *datagridview.*
16. Tombol bata untuk membatalkan proses.
17. Tombol simpan yang berfungsi untuk menyimpan hasil dari proses pada tahapan ini.

4.4. Pengujian Sistem

4.4.1. Pengujian

Pada sistem ini, terdapat dua macam pengujian yang dilakukan. Pengujian yang pertama dilakukan untuk mengetahui pengaruh nilai k-partisi *frequent 1-itemset* terhadap tingkat kekuatan *rule* yang telah dihasilkan menggunakan *Conviction* dan *Hiper Lift* serta pengaruh k-partisi terhadap jumlah *rule* yang dihasilkan dengan menerapkan metode *Multipass Direct Hashing and Prunning (M-DHP)*. Pengujian kedua dilakukan untuk mengetahui pengaruh *Confidence, Conviction* dan *Hiper Lift Ratio* terhadap tingkat akurasi aturan asosiasi pada sistem ini.

4.4.1.1. Pengujian Pengaruh k-Partisi Terhadap Tingkat Kekuatan *Rule* dan Jumlah *Rule.*

Pengujian yang pertama ini dilakukan pada data transaksi yang telah siap diolah dari hasil tahapan preprocessing. Data input transaksi tersebut akan diuji pada minimum *support* dan minimum *confidence* yang telah ditentukan. Adapun nilai minimum *confidence* hanya diambil pada 70% dengan tujuan agar *rule* yang dihasilkan memiliki nilai *confidence* minimal 70% dengan variasi nilai minimum *support* 6%, 7%, dan 8% agar *items* yang memiliki *support* rendah dapat memenuhi syarat *minimum support*, serta dilakukan pengujian tiap k-partisi dimulai dari 2-partisi hingga n-partisi yang tergantung dari *frequent 1-itemset* yang dihasilkan berdasar pada input batasan nilai *minimum support*. Tahapan selanjutnya menbandingkan k-partisi terhadap tingkat kekuatan *rule* yang telah dihasilkan menggunakan *conviction* dan *hiper lift ratio* dengan tujuan untuk manganalisa bagaimana pengaruh perubahan k-partisi terhadap tingkat kekuatan *rule*, serta

membandingkan k-partisi dengan jumlah *rule* yang dihasilkan dengan tujuan untuk menganalisa pengaruk k-partisi terhadap jumlah kombinasi *rule* yang dihasilkan menggunakan variasi minimum *support* dan minimum *confidence* yang telah ditentukan. Hasil pengujian yang terdiri dari rata-rata *conviction*, rata-rata *hiper lift ratio*, dan jumlah *rule* ditunjukkan pada Tabel 4.1. Data pada Tabel 4.1 dihasilkan pada minimum support 6%, 7% dan 8%.

Tabel 4. 1 Hasil Pengujian Rata-Rata Conviction dan HLR serta Jumlah Rule

Min. Confidence	**Min. Support**	**k-partisi**	**Jml. *Rule***	**Rata-rata *Conviction***	**Rata-rata *HLR***
70%	6%	2	142	5.294	4.807
		3	105	4.508	4.165
		4	103	4.540	4.207
		5	103	4.540	4.207
		6	102	4.549	4.234
		7	60	4.241	3.718
		8	28	3.848	2.976
		9	26	3.844	3.044
		10	25	3.795	3.107
		11	24	3.896	3.188
		12	15	3.317	2.611
		13	10	2.977	2.466
		14	5	2.377	2.038
		15	5	2.377	2.038
		16	2	2.575	1.344
		17	1	3.264	1.409
70%	7%	2	62	4.176	3.578
		3	61	4.197	3.614
		4	61	4.197	3.614
		5	60	4.206	3.651
		6	43	3.962	3.412
		7	25	3.851	2.930
		8	23	3.846	3.002
		9	22	3.790	3.072
		10	21	3.906	3.163
		11	12	3.189	2.423
		12	9	2.701	2.302
		13	5	2.377	2.038
		14	5	2.377	2.038
		15	2	2.575	1.344
		16	1	3.264	1.409
70%	8%	2	47	4.265	3.336

		3	47	4.265	3.336
		4	46	4.279	3.377
		5	29	3.968	2.863
		6	19	3.786	2.642
		7	17	3.772	2.706
		8	16	3.690	2.784
		9	15	3.845	2.892
		10	8	3.390	2.030
		11	5	2.632	1.577
		12	3	2.240	1.303
		13	3	2.240	1.303
		14	2	2.575	1.344
		15	1	3.264	1.409

Berdasarkan data pada tabel 4.1 dapat diamati bahwa jumlah *rule* tertinggi untuk minimum *support* 6% dan minimum *confidence* 70% adalah ketika k=2 dengan jumlah 142 *rule*, sedangkan untuk jumlah terendah pada k=17 dengan jumlah 1 *rule*. Pada *minimum support* 7% dan *minimum confident* 70% jumlah tertinggi pada k=2 berjumlah 62 *rule* dan terendah pada k=16 dengan jumlah 1 *rule*. Untuk *minimum support* 8% dan *minimum confident* 70% jumlah tertinggi pada k=2 berjumlah 47 *rule* dan terendah pada k=15 dengan jumlah 1 *rule*.

Nilai rata-rata *conviction* terendah berada pada nilai k=15 dengan nilai rata-rata 2.377142857, sedangkan untuk nilai tertinggi pada k=2 dengan nilai 5.293828355 untuk *minimum support* 6% dan *minimum confident* 70%. Pada *minimum support* 7% dan *minimum confident* 70% nilai terendah rata-rata *conviction* pada k=14 dengan nilai 2.377142857 dan nilai tertinggi rata-rata *conviction* pada k=2 dengan nilai 4.175655796. Untuk *minimum support* 8% dan *minimum confident* 70% nilai terendah rata-rata *conviction* pada k=13 dengan nilai 2.24029304 dan nilai tertinggi rata-rata *conviction* pada k=2 dengan nilai 4.265392409.

Pada tabel 4.1 dapat diamati juga bahwa nilai rata-rata *hiper lift ratio* terendah berada pada nilai k=16 dengan nilai 1.344382523, sedangkan untuk nilai tertinggi pada k=2 dengan nilai rata-rata *hiper lift ratio* 4.806898763 untuk *minimum support* 6% dan *minimum confident* 70%. Pada *minimum support* 7% dan *minimum confident* 70% nilai terendah rata-rata *hiper lift ratio* pada k=15 dengan nilai 1.344382523 dan nilai terbesar rata-rata *hiper lift ratio* pada k=5

dengan nilai 3.650580745. Untuk *minimum support* 8% dan *minimum confident* 70% nilai terendah rata-rata *hiper lift ratio* pada k=13 dengan nilai 1.30261749 dan nilai terbesar rata-rata *hiper lift ratio* pada k=4 dengan nilai 3.377434279. Adapun untuk mengetahui pengaruh k-partisi terhadap jumlah *rule* dan tingkat kekuatan *rule* yang telah dihasilkan menggunakan *conviction* dan *hiper lift ratio* pada sistem ini akan dijelaskan lebih lengkap pada sub-bab 4.5 Analisa Hasil.

4.4.1.2. Pengujian Pengaruh *Confidence, Conviction* dan *Hiper Lift Ratio* Terhadap Tingkat Akurasi *Rule*.

Pada pengujian yang kedua ini merupakan hasil dari pencarian pengaruh nilai *Conviction, Hiper Lift Ratio, Confidence* dari *rule-rule* yang dihasilkan terhadap akurasi yang bertujuan untuk mengetahui keakuratan *rule* yang dihasilkan, berdasar pengujian dengan data latih. Untuk pengujian ini menggunakan data latih sebanyak 90% dan data uji 10% dari data transaksi Al-Qur'an yang terbentuk pada proses *preprocessing*. Untuk pengujian ini menggunakan nilai partisi 4 agar kombinasi rule yang dihasilkan lebih banyak, untuk rule yang dipakai dalam pengujian merupakan *rule* yang memiliki nilai *Conviction* dan *Hiper Lift Ratio* lebih dari 1. Untuk variasi nilai minimum *confidence* ditetapkan pada 70%, 80% dan 90%, sedangkan untuk minimum *support* ditetapkan pada nilai 6%, 7% dan 8% karena berdasarkan pada berbagai percobaan pada batasan tersebut manghasilkan nilai akurasi terbaik.

Tabel 4. 2 Hasil Pengujian Conviction, HLR dan Confidence Terhadap Akurasi Rule

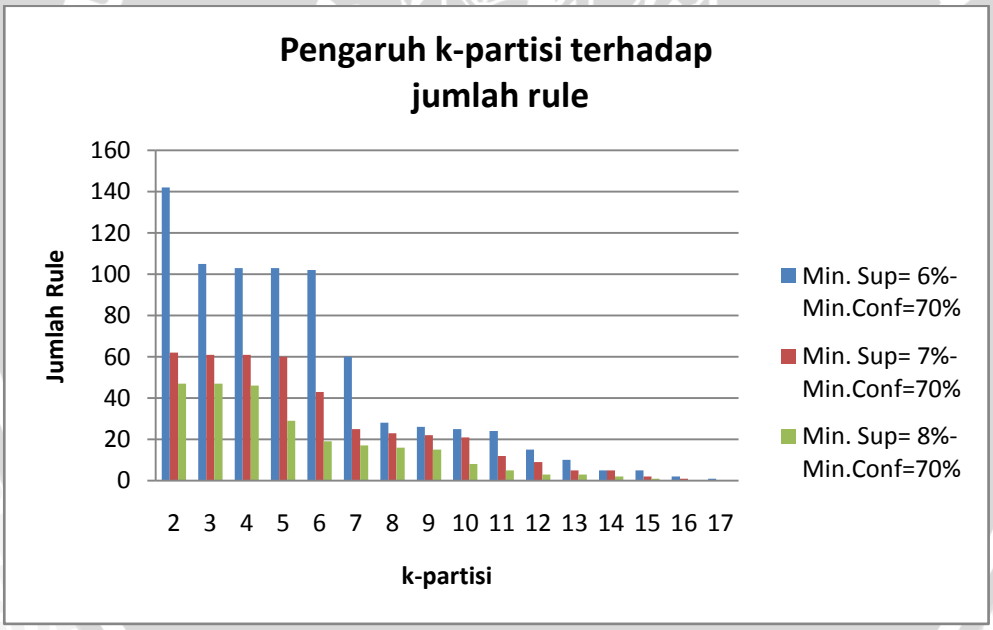
Min. *confidence*	Min. *Support*	k-partisi	Akurasi (%)
70%	6%	4	36.89
	7%	4	55.74
	8%	4	67.4
80%	6%	4	46
	7%	4	67.86
	8%	4	75
90%	6%	4	45.5
	7%	4	76.9
	8%	4	81.8

Berdasarkan pada tabel 4.2, nilai akurasi terendah sebesar 36,89% pada minimum *confidence* 70%, minimum *support* 6%, sedangkan untuk nilai akurasi terbesar pada minimum *confidence* 90%, minimum *support* 8% dengan nilai sebesar 81,8%.

4.5. Analisa Hasil

4.5.1. Analisa Hasil Pengujian Pengaruh k-Partisi Terhadap Tingkat Kekuatan *Rule* dan Jumlah *Rule*.

Pada tahapan ini, hasil pengujian pengaruh k-partisi terhadap jumlah *rule* yang dihasilkan dengan range-partisi 2-partisi hingga n-partisi (tergantung dari batas minimum *support*), variasi minimum *support* 6%, 7% dan 8%, serta minimum *confidence* 70% dapat dilihat pada gambar 4.3.

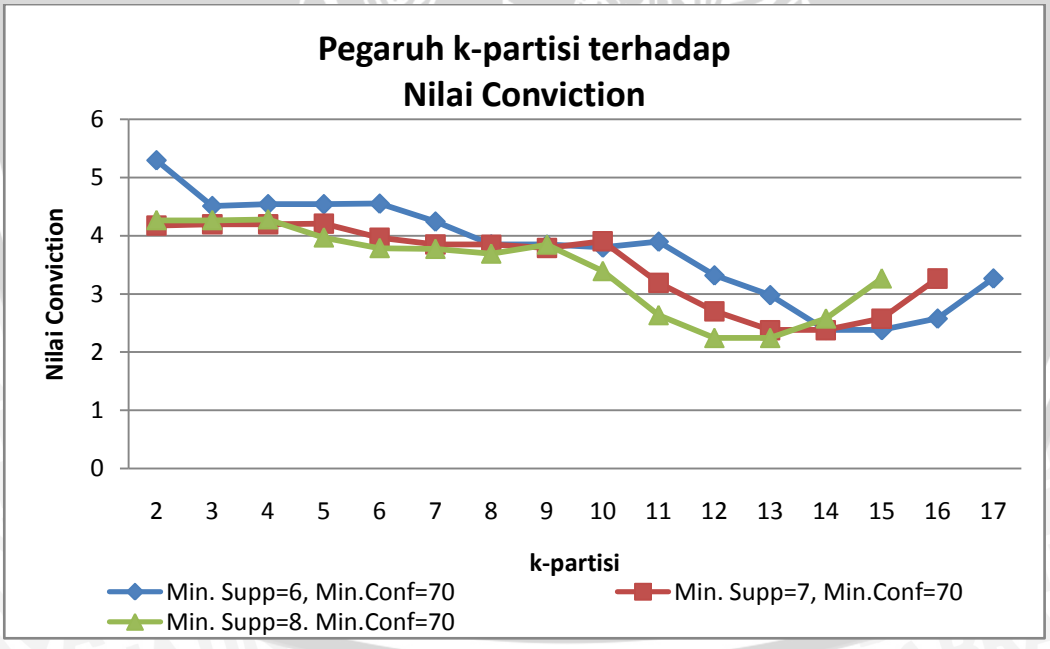


Gambar 4.3 Pengaruh k-partisi terhadap jumlah rule

Sebagaimana telah ditunjukkan pada gambar 4.3, jumlah *rule* dengan k-partisi=2, minimum *support*=6% dan minimum *confidence*=70% memiliki jumlah *rule* tertinggi, yaitu berjumlah 142 *rule*. Pada grafik minimum *support* 6% dan minimum *confidence* 70% terjadi penurunan jumlah *rule* seiring bertambahnya jumlah k-partisi. Hal ini pula yang terjadi pada minimum *support* 7% dan 8%,

yaitu semakin besar k-partisi maka mengakibatkan jumlah *rule* yang dihasilkan juga semakin berkurang.

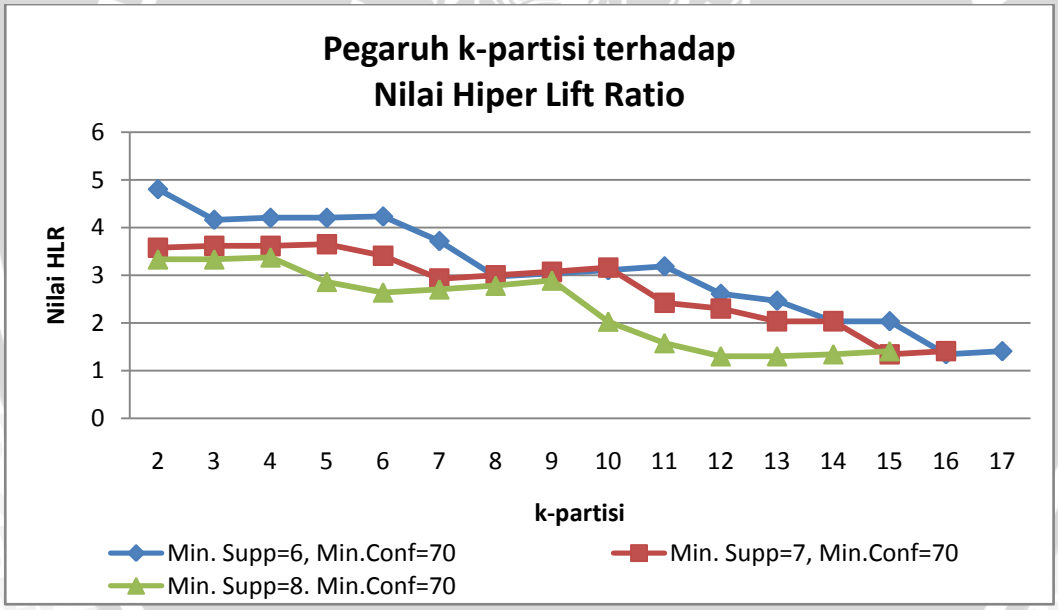
Seperti yang ditunjukkan pada gambar 4.3, bahwa k-partisi juga berperan dalam hal jumlah *rule* yang dihasilkan pada sistem ini, yaitu semakin tinggi nilai k-partisi maka semakin rendah pula jumlah *rule* yang dihasilkan begitupun sebaliknya, ketika k-partisi rendah maka jumlah *rule* yang dihasilkan juga cenderung meningkat. Hal ini disebabkan pada metode M-DHP terdapat proses reduksi *item* database transaksi dan proses reduksi berkaitan dengan proses partisi karena pada tahapan partisi memilliki peranan dalam pembentukan *candidate k+1-itemset*. Adapun dalam mereduksi suatu *item* pada database transaksi seperti yang telah dipaparkan pada Bab Metodologi dan Perancangan tergantung dari jumlah dari *item* tersebut, dimana untuk menemukan jumlah tiap *item* database transaksi berdasar pada kombinasi *candidate itemset* yang terbentuk dari *frequent 1-itemset* yang telah terpartisi.



Gambar 4.4 Pengaruh k-partisi terhadap Nilai Conviction

Pada Gambar 4.4 menunjukkan pengaruh k-partisi terhadap tingkat kekuatan *rule* yang dihasilkan menggunakan *Conviction*, adapun dapat diamati bahwa nilai *conviction* yang dihasilkan untuk mengukur kekuatan suatu *rule*

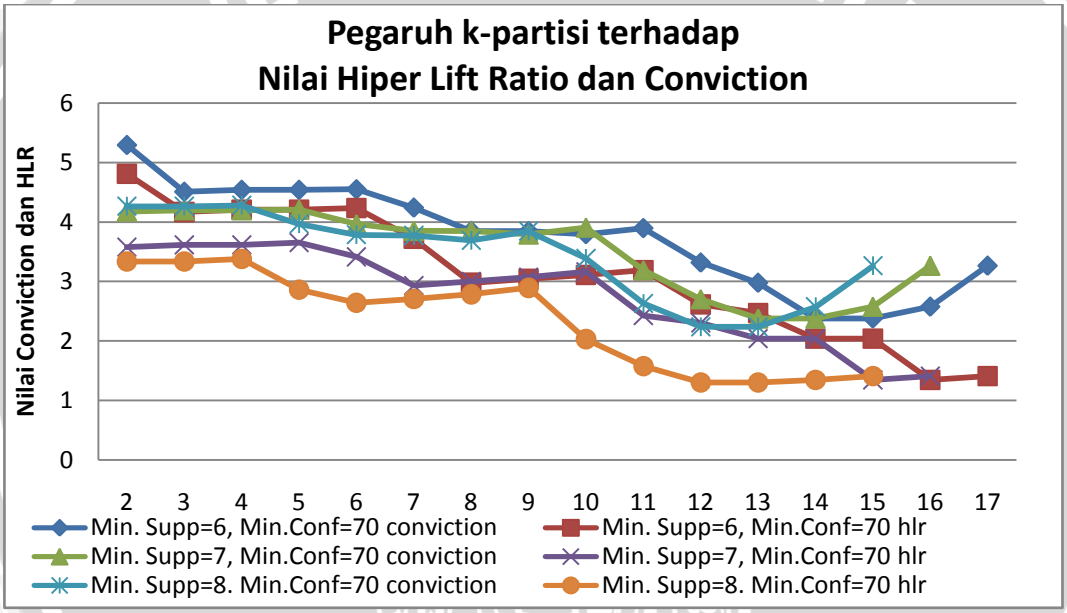
tergolong pada *rule* yang baik atau kuat walaupun terlihat pada gambar 4.4 terdapat pola-pola grafik yang menurun. Misalkan pada minimum *support* 6% dan minimum *confidence* 70%, untuk k=2 nilai *conviction* berada pada nilai 5, 3 dan selanjutnya seiring dengan bertambahnya k-partisi maka grafik juga cenderung menurun namun pada akhirnya meningkat. Berdasar penjelasan tersebut terlihat bahwa terdapat perubahan akibat dari k-partisi, namun tetap tingkat kekuatan *rule* yang telah dihasilkan menggunakan *conviction* pada taraf tingkat kekuatan *rule* yang baik, dimana sesuai dengan yang telah dipaparkan pada Bab Tinjauan Pustaka bahwa nilai range pada *conviction* berada pada nilai 0.5,...,1,...∞ dan jika *conviction* menghasilkan nilai rule yang menjauh dari 1 maka akan dianggap akurat atau *rule* tersebut memiliki tingkat kekuatan yang baik (Azevedo dkk, 2006).



Gambar 4.5 Pengaruh k-partisi terhadap Nilai HLR

Sebagaimana yang telah ditunjukkan pada Gambar 4.5 bahwa pengaruh k-partisi terhadap tingkat kekuatan *rule* yang dihasilkan menggunakan *Hiper Lift Ratio*, adapun dapat diamati bahwa nilai *Hiper Lift Ratio* yang dihasilkan untuk mengukur kekuatan suatu *rule* tergolong pada *rule* yang baik atau kuat walaupun terlihat pada gambar 4.5 terdapat pola-pola grafik yang menurun. Misalkan pada

minimum *support* 6% dan minimum *confidence* 70%, untuk k=2 nilai *Hiper Lift Ratio* berada pada nilai 4,8 dan selanjutnya seiring dengan bertambahnya k-partisi maka grafik juga cenderung menurun. Berdasar penjelasan tersebut, terlihat bahwa terdapat perubahan nilai akibat k-partisi, namun tidak dapat dikatakan bahwa tingkat kekuatan *rule* yang telah dihasilkan menggunakan *Hiper Lift Ratio* cenderung tidak atau kurang kuat karena sesuai dengan yang telah dipaparkan pada Bab Tinjauan Pustaka bahwa apabila nilai *hiper lift ratio* dari suatu rule > 1 maka dianggap akurat atau tingkat kekuatan *rule* yang dihasilkan baik. (Hashler dkk, 2006).



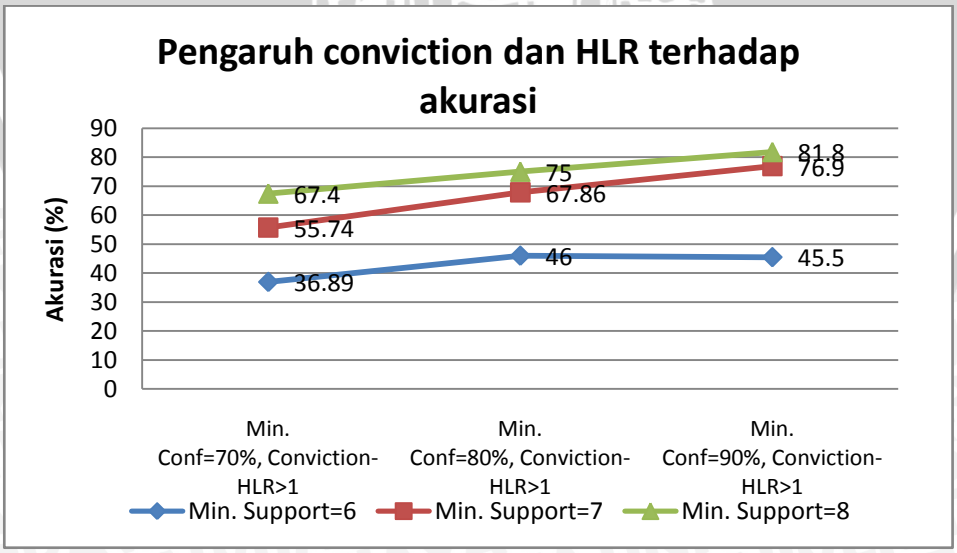
Gambar 4.6 Pengaruh k-partisi terhadap Nilai HLR dan Conviction

Berdasarkan analisa di atas yaitu pengaruh k-partisi terhadap jumlah *rule* dan pengaruh k-partisi terhadap tingkat kekuatan *rule* yang dihasilkan manggunakan *Conviction* dan *Hiper Lift Ratio* pada gambar 4.6, dapat disimpulkan bahwa perubahan nilai k-partisi memiliki pengaruh terhadap nilai *Conviction* dan *Hiper Lift Ratio* yang dihasilkan, hal ini terlihat dengan adanya pola-pola grafik baik menurun ataupun meningkat. Terdapat pula titik perpotongan antara nilai *hiper lift ratio* dengan *conviction* pada beberapa k-partisi, misalkan pada partisi 3,4,5 dan 11 pada *minimum support* 6% dan 7% ataupun

pada partisi 12 untuk *minimum support* 7% dan 8%, namun titik-titik perpotongan tersebut tidak membentuk satu pola tertentu karena bersifat acak pada beberapa k-partisi. Oleh karena itu meskipun terdapat pengaruh k-partisi terhadap tingkat kekuatan r*ule* yang dihasilkan oleh *Conviction* dan *Hiper Lift Ratio*, terlihat bahwa pada pengujian ini tetap menghasilkan *rule* dengan tingkat kekuatan yang baik, namun dengan nilai k-partisi yang tidak terlalu tinggi juga memiliki keunggulan yaitu semakin banyak kombinasi *rule* yang dihasilkan.

4.5.2. Analisa Hasil Pengujian Pengaruh *Conviction, Hiper Lift Ratio* dan *Confidence* Terhadap Tingkat Akurasi *Rule*

Pada tahapan ini, merupakan hasil pengujian pengaruh nilai *Conviction* , *Hiper Lift Ratio* dan *Confidence* terhadap akurasi dari *rule-rule* yang dihasilkan. Untuk nilai partisi dipilih pada k=4 agar kombinasi rule yang dihasilkan lebih banyak, sedangkan *rule* yang digunakan pada pengujian ini merupakan *rule-rule* yang nilai *Conviction* dan *Hiper Lift Ratio* memenuhi syarat sebagai *rule* yang memiliki tingkat akurasi yang baik. Untuk variasi nilai minimum *confidence* 70%, 80% dan 90%, sedangkan untuk minimum *support* diambil pada nilai 6%, 7% dan 8%. Hasil pada pengujian ini ditunjukkan pada gambar 4.6.



Gambar 4.7 Pengaruh Conviction dan HLR terhadap Nilai Akurasi

Sebagaimana ditunjukkan pada gambar 4.6, akurasi pada minimum *support* 8% dan minimum *confidence* 90%, memiliki nilai yang paling tinggi, yaitu 81,8%.

Pada grafik minimum *support* 7% terjadi peningkatan akurasi seiring dengan bertambahnya persentase minimum *confidence*. Untuk minimum *confidence* 70%, akurasi adalah 55,74%, kemudian pada minimum *confidence* 80% dan 90%, akurasi mengalami peningkatan menjadi masing-masing 67,86% dan 76,9%. Meskipun pada minimum *support* 6% terjadi penurunan akurasi yaitu pada minimum *confidence* 80% akurasinya sebesar 46% menjadi 45.5% pada minimum *confidence* 90%, namun penurunan tersebut tidak terlalu signifikan dan cenderung membentuk grafik datar. Oleh karena itu secara umum peningkatan akurasi ini juga terjadi pada grafik minimum *support* 6% dan 8%. Hal ini disebabkan selain *rule-rule* yang digunakan pada pengujian ini merupakan *rule* yang memiliki tingkat kekuatan yang baik berdasarkan pada nilai *conviction* dan *hiper lift ratio* juga karena ketika minimum *support* semakin meningkat maka dalam pembentukan *rule* akan disaring *itemset* data transaksi yang kemunculannya pada data transaksi tinggi, sedangkan untuk standart minimum *confidence* juga dipilih *itemset* yang memiliki *ratio* kemunculan item *antecedent* dan *consequent* secara bersamaan dengan kemunculan item *antecedent* atau nilai *confidence* dari *rule* yang semakin tinggi pula. Sehingga semakin meningkat nilai minimum *support* dan minimum *confidence* maka cenderung akan semakin meningkat pula akurasinya.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil penelitian tentang pencarian asosiasi topik dalam ayat Al-Qur'an dengan menerapkan metode *Multipass Direct Hashing and Prunning* (M-DHP), dapat disimpulkan bahwa:

1. Penerapan metode *Multipass Direct Hashing and Prunning* (M-DHP) untuk menemukan asosiasi topik dalam ayat Al-Qur'an meliputi beberapa tahap, yaitu sebagai berikut :
   a. Tahap *preprocessing*, merupakan tahap pembentukan data transaksi yang selanjutnya digunakan pada implementasi penelitian ini.
   b. Tahap implementasi *Multipass Direct Hashing and Prunning* (M-DHP), meliputi prosedur pembentukan kandidat *itemset*, menghitung *support itemset* data transaksi, pencarian *frequent itemset*, proses partisi *frequent 1-itemset* dan proses reduksi database data transaksi.
   c. Tahap *generate rule*, meliputi pembentukan *rule*, prosedur hitung *confidence*, menghitung nilai *conviction* dan *hiper lift ratio* dari tiap *rule.*
2. Nilai k-partisi berpengaruh pada nilai *conviction* dan *hiper lift ratio* dari tiap-tiap *rule*, namun tetap menghasilkan *rule* dengan tingkat kekuatan yang baik. Meskipun hasil dari k-partisi tetap menghasilkan tingkat kekuatan *rule* yang baik, tetapi dalam pemilihan k-partisi sebaiknya tidak terlalu tinggi karena semakin tinggi k-partisi maka kombinasi *rule* yang dihasilkan semakin rendah.
3. Berdasarkan pada penelitian ini menghasilkan *rule-rule* yang menampilkan pola keterkaitan topik dalam Al-Qur'an seperti yang ditunjukkan pada Lampiran 2, adapun contohnya antara lain:
   a. **Jika** keutamaan doa **maka** klasifikasi zikir, doa nabi
   b. **Jika** hukum shalat **maka** keutamaan shalat, hukum zakat

c. **Jika** keutamaan doa, terkabulnya doa **maka** klasifikasi zikir, doa nabi

4. Tingkat akurasi tertinggi yang dapat dihasilkan oleh sistem adalah sebesar 81,87% pada minimum *support* 8% dan minimum *confidence* 90%. Selain pengaruh dari *rule* yang digunakan untuk pengujian merupakan *rule* dengan tingkat kekuatan yang baik berdasar pada nilai *conviction* dan *hiper lift ratio* dari setiap *rule*, juga disebabkan oleh semakin meningkatnya minimum *support* dan minimum *confidence* yang mempengaruhi peningkatan akurasi sistem, karena ketika minimum *support* semakin meningkat maka dalam pembentukan *rule* akan disaring *itemset* data transaksi yang kemunculannya pada data transaksi tinggi, sedangkan untuk standart minimum *confidence* juga dipilih *itemset* yang memiliki *ratio* kemunculan item *antecedent* dan *consequent* secara bersamaan dengan kemunculan item *antecedent* atau nilai *confidence* dari *rule* yang semakin tinggi pula.

5.2. Saran

Beberapa saran untuk pengembangan lebih lanjut yang dapat diberikan oleh penulis adalah:

1. Menambah topik-topik lain untuk dikaji sehingga keterkaitan topik yang terkandung dalam ayat dari surat-surat pada Al Qur’an secara keseluruhan dapat ditemukan.

DAFTAR PUSTAKA

[AGR-94] Agrawal, S., Srikant, R. 1994. *Fast Algorithm for Mining Association Rules*. Proceedings of the 20th VLDB Conference Santiago, Chile. San Jose.

[ANO-04] Anonymous.2004.*Al Qur'an digital versi 2.1*. http://www.alquran-digital.com. Diakses pada tanggal 9 april 2012.

[AZE-06] Azevedo, P.J., Jorge,A.M. 2006.*Comparing Rule Measures for Predictive Association Rules*. Departamento de Informatica Universidade do Porto. Portugal.

[BEL-84] Bell, D. A., Deen,S.M. 1984. *Hash Trees Versus B-Trees*. Wiley Heyden Ltd. UK.

[GOO-01] Goodrich, M. T., Tamassia, R. 2001. *Data tructures & Algorithms in Java*. John Wiley & Sons, Inc. California.

[HAD-05] Hadhiri, C. SP. 2005. *Klasifikasi Kandungan Al-Qur'an*. Gema Insani Press. Jakarta

[HAS-06] Hashler, M., Hornik, K. 2006. *New Probabilistic Interest MeasuresFor Association Rules*. Departement of Statistics and Mathematics WU Vienna University. Vienna.

[HOL-00] Holt, J. D., Chung, S. M. 2000. *Efficient Mining of Association Rules in Text Databases*. Department of Computer Science and Engineering Wright StateUniversity .Ohio, USA.

[KUS-09] Kusrini, Luthfi, ET. 2009. *Algoritma Data Mining*. Penerbit Andi. Yogyakarta.

[LAR-05] Larose, Daniel T. 2005. *Discovering Knowledge in Data. An Introduction to Data mining.* John willey & Sons. New Jersey.

[MCM-07] McMillan, M. 2007. *Data Structures and Algorithms Using C#*. Cambridge University Press. New York.

[ORL-01] Orlando, S., Palmeri, P., Perego, R. 2001.*Enhancing the Apriori algorithm for Frequent Set Counting*.CNUCE-CNR.Italy

[PAR-95] Park, J. S., Chen M., Yen, P.S. 1995. *An Effective Hash-Based Algorithm for Mining Association Rules*. IBM Research Center. New York.

[SHI-02] Shiddieqy, Hasbi, T. 2002.*Ilmu-ilmu Al Qur'an: Ilmu-ilmu Pokok dalam Menafsirkan Al Qur'an*.Pustaka Rizki Putra.Semarang

[VEE-10] Veeramalai, S., Jaisankar, N., Kannan, A. 2010. *"Efficient Web Log Mining Using Enhanced Apriori Algorithm with Hash Tree and Fuzzy"*.International journal of computer science & information Technology (IJCSIT) vol.2, No4.

LAMPIRAN

Lampiran 1 Data Topik Al-Qur'an Dalam Lingkup Pokok Bahasan Ibadah

Kode Item	Item
1	Keutamaan bersuci
2	Membersihkan bejana air
3	Kebersihan pakaian
4	Haid
5	Najis
6	Disyariatkannya wudhu
7	Hukum wudhu
8	Rukun wudhu
9	Yang membatalkan wudhu
10	Hukum mandi besar
11	Yang diharamkan dan diperbolehkan orang yang junub
12	Disyariatkannya tayammum
13	Rukun tayammum
14	Cara tayammum
15	Yang diperbolehkan bagi orang bertayammum
16	Keutamaan Shalat
17	Hukum shalat
18	Syarat shalat
19	Rukun shalat
20	Hal yang disunnahkan dalam shalat
21	Tempat shalat
22	Waktu shalat
23	Azan
24	Mengqadha shalat
25	Menqashar shalat
26	Shalat Jum'at
27	Shalat sunnah
28	Shalat Khauf
29	Shalat 'ied
30	Sujud tilawah
31	Harta
32	Hukum zakat
33	Pembagian zakat
34	Zakat tanaman dan buah
35	Keutamaan puasa dan pahalanya
36	Rukun-rukun puasa
37	Yang membatalkan puasa
38	Bulan ramadhan

39	Mengqadha puasa
40	Haji
41	Ihram
42	Masuk tanah haram
43	Thawaf
44	Sa'i antara Safa dan Marwah
45	Wukuf di padang Arafah
46	Hari raya kurban
47	Hadyu dan kurban
48	Bercukur
49	Kembali dari haji dan umrah
50	Ihshar
51	Kafarat haji
52	Sumpah
53	Nazar
54	Keutamaan majelis-majelis zikir
55	Etika zikir
56	Klasifikasi zikir
57	tempat zikir
58	Waktu-waktu zikir
59	Keutamaan doa
60	Perintah berdoa
61	Etika berdoa
62	Waktu berdoa dan sebabnya
63	Doa para nabi
64	Doa orang-orang beriman
65	Doa malaikat untuk orang-orang beriman
66	Terkabulnya doa
67	Berdoa bukan kepada Allah

Lampiran 2 Contoh Data Hasil pengujian pada k-partisi 4, minimum support 8% dan min confidence 80%

RULE	SUPPORT	CONFIDENCE	CONVICTION	HPL
keutamaan_doa=>klasifikasi_zikir	12.088	100	Infinity	1.585
doa_orang_beriman=>klasifikasi_zikir	14.286	92.857	5.077	1.472
terkabulnya_doa=>klasifikasi_zikir	15.385	87.5	2.901	1.387
waktu_zikir=>klasifikasi_zikir	20.879	95	7.253	1.506
doa_nabi=>klasifikasi_zikir	19.780	90	3.626	1.426
waktu_berdoa_dan_sebabnya=>klasifikasi_zikir	26.374	88.889	3.264	1.409
hukum_shalat=>keutamaan_shalat	12.088	91.667	8.571	3.241
keutamaan_doa=>doa_nabi	10.989	90.909	8.582	4.178
hukum_shalat=>hukum_zakat	12.088	91.667	8.571	3.241
hukum_shalat=>klasifikasi_zikir	10.989	83.333	2.176	1.321
hukum_zakat=>klasifikasi_zikir	23.077	80.769	1.886	1.280
terkabulnya_doa=>doa_nabi	15.385	87.5	6.242	4.021
keutamaan_majelis_zikir=>klasifikasi_zikir	13.187	92.308	4.714	1.463
keutamaan_doa=>klasifikasi_zikir doa_nabi	10.989	90.909	8.824	4.642
hukum_shalat=>keutamaan_shalat hukum_zakat	10.989	83.333	4.813	4.256
keutamaan_doa terkabulnya_doa=>klasifikasi_zikir doa_nabi	8.791	100	Infinity	5.107
keutamaan_doa doa_nabi=>klasifikasi_zikir terkabulnya_doa	8.791	80	4.231	5.253
keutamaan_doa doa_nabi=>klasifikasi_zikir waktu_berdoa_dan_sebabnya	8.791	80	3.681	3.064
keutamaan_doa waktu_berdoa_dan_sebabnya=>klasifikasi_zikir doa_nabi	8.791	100	Infinity	5.107
waktu_berdoa_dan_sebabnya terkabulnya_doa=>klasifikasi_zikir doa_nabi	8.791	80	4.011	4.085
keutamaan_shalat waktu_zikir=>hukum_zakat klasifikasi_zikir	8.791	80	3.846	3.502
hukum_shalat klasifikasi_zikir=>keutamaan_shalat hukum_zakat	8.791	80	4.011	4.085

sumpah terkabulnya_doa=>klasifikasi_zikir doa_nabi	8.791	88.889	7.220	4.539
sumpah doa_nabi=>klasifikasi_zikir terkabulnya_doa	8.791	80	4.231	5.253