BAB III
METODE & PERANCANGAN

3.1 Metode Penelitian

Metode penelitian yang dilakukan penulis dengan rincian sebagai berikut: membaca buku atau jurnal baik *online* maupun cetak yang berkaitan dengan toko online serta algoritma FP-growth. Dan mengumpulkan data – data yang dibutuhkan dalam perancangan *module* katalog toko online.

1. Telaah Literatur

   Telaah literatur dilakukan dengan mempelajari jurnal – jurnal berkaitan dengan algoritma yang digunakan oleh penulis. Serta teori mengenai *Model View Controller* sebagai konsep dalam perancangan sistem katalog toko *online*.

2. Pengumpulan data

   Pengumpulan data pada tabel penjual dari sistem toko *online* yang telah berjalan, data – data ini yang nantinya digunakan oleh algoritma FP-Growth. Pengumpulan data menggunakan cara mengolah data – data mentah dari *database* toko *online* PT. Kandel.

3. Perancangan sistem

   Tahap perancangan dimulai dengan mengumpulkan dan mempersiapkan data – data yang diperlukan dalam pembuat module katalog toko online. Selain itu juga membuat flowcart sistem yang akan dibuat, serta membuat pseudocode atau sebuat outline kode yang dapat mempermudah penerapan saat penulisan kode asli (pembuat program).

4. Implementasi

Implementasi mencakup pembuat *source code* dan tampilan program (*User Interface*). *Source code* ialah suatu intruksi program dalam bentuk aslinya. Kata *source* membedakan *code* dari beberapa bentuk lain (sebagai contoh: *object code* dan *executable code*).

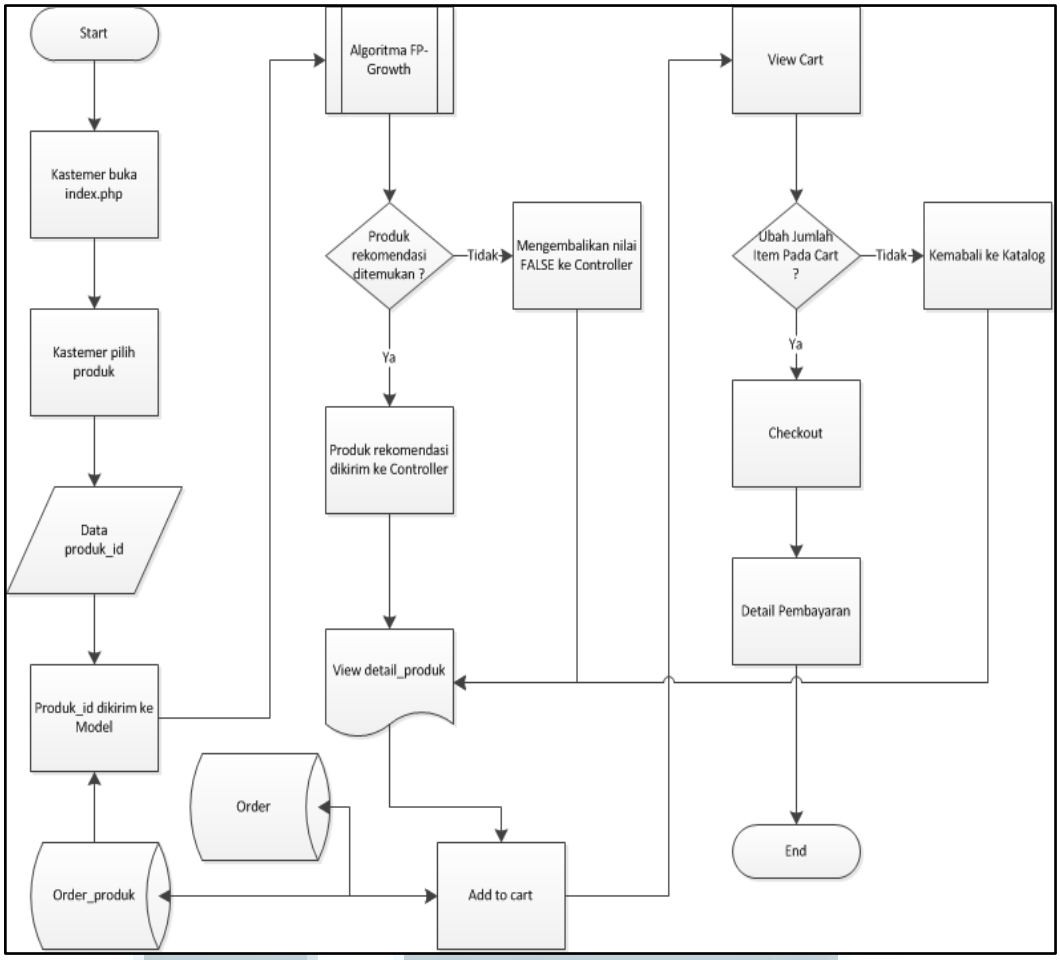
5. Ujicoba

Ujicoba dilakukan untuk mengetahui hasil yang didapat dari algoritma FP-Growth dan untuk mencapai tujuan dari pembuat sistem ini.

3.2 Perancangan Sistem

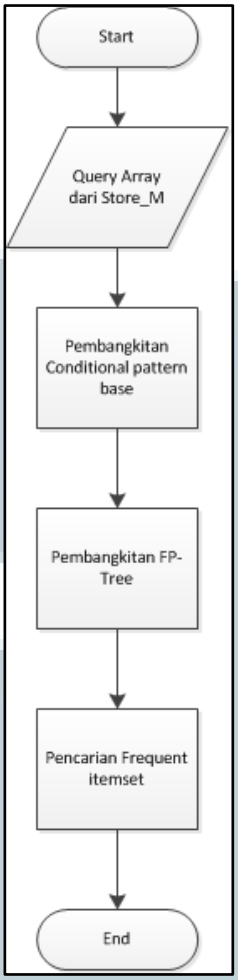
Dalam tahap ini sangatlah penting berpengaruh saat pembuat program. Pembuatan *flowcart* sangat membantu dalam pembangun sistem *module* katalog online adapun *flowcart* tersebut dapat terlihat pada gambar 3.1.

Dari gambar 3.1 dapat dijelaskan mengenai alur sistem yang akan dibangun. Pertama *User*/Pengunjung yang mengakses *detail page* akan mengirimkan *produk_id* ke sistem, selanjutnya sistem akan ditangkap *produk_id* tersebut oleh sebuah model (*Model Store_M*). Dalam *model* ini akan mengeksekusi perintah *sql* dan melakukan *query* dari *database.* Jika *produk_id* tersebut ditemukan pada *table order_produk,* maka *model* tersebut akan mengirimkan hasil *query* tersebut ke *controller* untuk dapat melakukan proses pembangkitan FP-Tree. Jika *produk_id* tersebut tidak ditemukan di*database* maka *model* akan mengirimkan nilai *false* pada *controller* dan *controller* tidak akan menampilkan produk rekomendasi pada kolom rekomendasi.



Gambar 3. 1 Flowcart module katalog

Pada gambar 3.1 terdapat sub-proses yang kemudian dapat dijabarkan menjadi gambar 3.2. Flowchart dimulai dari data *query* yang dikirimkan oleh *model* berupa *array* akan diolah pada *controller*.



Gambar 3. 2 Flowchart Algoritma FP-Growth

Penggalian itemset yang frequent dengan menggunakan algoritma *FP-Growth* akan dilakukan dengan cara membangkitkan struktur data *tree* atau disebut dengan *FP-Tree*. Metode *FP-Growth* dapat dibagi menjadi 3 tahapan utama yaitu sebagai berikut:

1. Tahap pembangkitan *conditional pattern base*

   Membuat *Header Item*, *Header* dalam hal ini selain sebagai *header* suatu item ke FP-Tree juga sebagai jenis item dasar yang memenuhi minimum *support*. Setelah mendapatkan item dan nilai support-nya, maka item yang tidak frequent dibuang dan item diurutkan berdasarkan nilai *support*-nya. Header

untuk item, disiapkan pada suatu array tertentu dan ditambahkan ketika membuat FP-Tree

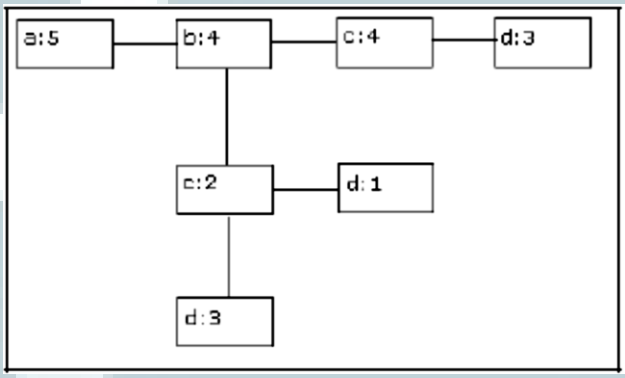
2. Tahap pembangkitan *conditional FP-Tree*

FP-Tree dibangun dengan mencari *item* sesuai urutan pada *item* yang *frequent*. Data transaksi tidak perlu diurutkan, dan untuk tiap *item* yang ditemukan bisa langsung dimasukkan ke dalam *FP-Tree*. Sesudah membuat *root*, tiap *item* yang ditemukan dimasukkan berdasarkan *path* pada *FP-Tree*. Jika item yang ditemukan sudah ada, maka nilai *support* item tersebut yang ditambahkan. Namun jika path belum ada, maka dibuat node baru untuk melengkapi path baru pada *FP-Tree* tersebut. Hal ini dilakukan selama item pada transaksi masih ada yang *qualified*, artinya memenuhi nilai minimum *support*. Jadi, *item-item* yang ditemukan dalam transaksi akan berurutan memanjang ke bawah. Dalam struktur *FP-Tree*, diterapkan alur path dari *child* hingga ke *root*. Jadi, suatu *path* utuh dalam *FP-Tree* adalah dari *child* terbawah hingga ke *root*. Tiap *node* pada *FP-Tree* memiliki *pointer* ke *parent*, sehingga pencarian - harus dimulai dari bawah.

3. Tahap pencarian *frequent itemset*.

Pencarian dilakukan berdasarkan keterlibatan *item* pada suatu *path*. Di setiap *path*, diperiksa semua kombinasi yang mungkin dimana *item* tersebut terlibat. Di iterasi berikutnya dilakukan dengan melibatkan *item* berikutnya, tanpa melibatkan *item* sebelumnya, sehingga *pattern* yang sama tidak akan ditemukan dua kali pada *path* yang sama. Bila *item* pertama suatu hasil kombinasi bukan *item* terakhir (sebelum *root*), maka kombinasi *itemset* tersebut masih bisa dikembangkan lagi.

Setelah mengolah FP-Tree menjadi pattern-pattern, diperlukan proses akumulasi pattern-pattern yang ditemukan mengingat pattern yang sama dapat ditemukan pada path yang berbeda. Untuk itu digunakan struktur data PatternTree (lihat Gambar 3.3). Setiap node di PatternTree merepresentasikan dan menyimpan frekuensi suatu pattern. PatternTree terdiri atas PatternTreeNode yang menyimpan nilai item, nilai support dan dilengkapi dengan dua pointer yaitu untuk horisontal dan vertical



Gambar 3. 3 Pembangkitan FP-Tree

Misalnya pada *node* d:1 di atas, berarti terdapat *pattern* a-c-d bernilai *support* 1. Kemudian bila ada *pattern* a-c-d lagi bernilai *support* n yang ditemukan dari *FP-Tree* maka nilai *support* 1 tersebut menjadi n+1. Contoh hasil lengkap dari PatternTree tersebut:

- a:5 menggambarkan bahwa ada pattern a sebanyak 5
- b:4 menggambarkan bahwa ada pattern a-b sebanyak 4
- c:4 menggambarkan bahwa ada pattern a-b-c sebanyak 4
- d:3 menggambarkan bahwa ada pattern a-b-c-d sebanyak 3
- c:2 menggambarkan bahwa ada pattern a-c sebanyak 2
- d:1 menggambarkan bahwa ada pattern a-c-d sebanyak 1
- d:3 menggambarkan bahwa ada pattern a-d sebanyak 3

Pattern yang tidak memenuhi minimum support, dihapus dari daftar *pattern*. *Pattern– pattern* yang tersisa kemudian diurutkan untuk memudahkan pembuatan *rules*.

Struktur data-struktur data yang digunakan dalam algoritma *FP-Grow*th adalah *FP-Tree* yang tersusun dari *FPTNode*, *PatternTree* yang terdiri dari *PatternTreeNode*, dan juga *FItemset*. *FP-Tree* adalah struktur *tree* yang menyimpan *item-item* yang telah ditemukan pada saat membaca data. Kemudian *FP-Tree* ini terdiri atas *FPTNode* yang membentuk suatu *path* untuk dicari terdapat *pattern* apa saja pada *path* tersebut. *FPTNode* adalah *node* pada *FP-Tree* yang menyimpan informasi *item* pada *node*, nilai *support*, *pointer* ke *parent*, dan *pointer* ke *node* berikutnya yang mempunyai *item* yang sama (*next*). *PatternTree* adalah struktur untuk menyimpan semua *pattern* yang ditemukan pada *FP-Tree*. *FItemset* menyimpan *frequent item set* yang telah ditemukan. Format ini digunakan terutama bila ingin menghasilkan *rule*, untuk membandingkan suatu *frequent item set* dengan *subset*-nya, akan lebih mudah dengan struktur seperti ini daripada mengolah *frequent item set* masih dalam bentuk *tree*.