Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

BAB III

METODE DAN PERANCANGAN SISTEM

3.1 Metode Penelitian

Metode penelitian yang digunakan dalam penelitian ini yaitu sebagai berikut.

1) Studi Literatur

   Penulis melakukan studi literatur melalui *paper* penelitian, buku, artikel, dan *slide* pembelajaran baik *online* maupun *offline* yang berkaitan dengan algoritma ant colony, algoritma backtracking, dan aplikasi tata letak barang.

2) Analisa Kebutuhan

   Penulis melakukan pencarian dan analisis kebutuhan-kebutuhan yang diperlukan dalam suatu aplikasi pengaturan tata letak barang. Kebutuhan-kebutuhan tersebut penulis dapatkan dengan cara melakukan wawancara terhadap karyawan gudang Toko Sepeda Populer. Kemudian, hasil wawancara dibandingkan dengan studi literatur yang telah dilakukan.

3) Perancangan Sistem

   Penulis melakukan perancangan alur kerja, database, dan *user interface* dari aplikasi.

4) Pemrograman

   Pada tahap ini dilakukan pembangunan / *coding* aplikasi berdasarkan rancangan yang telah dibuat dengan menggunakan bahasa pemrograman C#.

Rancangan Bangun ..., Ardy Wibowo Haryanto,FTI UMN, 2014

5) Uji Coba

Pada tahap ini dilakukan uji coba terhadap aplikasi yang telah dibuat. Penulis melakukan pengamatan apakah fungsi-fungsi yang terdapat dalam aplikasi telah berjalan sesuai dengan yang diinginkan atau belum dan pencarian bug. Hasil uji coba tersebut dievaluasi dan penulis melakukan perbaikan dan penambahan jika diperlukan.

3.2 Perancangan Sistem

Aplikasi pengaturan tata letak barang ini dibuat dengan menggunakan kombinasi dari algoritma ant colony dan backtracking sebagai algoritma utama untuk menentukan lokasi tata letak barang. Penentuan lokasi barang ditentukan dengan penelusuran *tree* yang dibangun berdasarkan prinsip algorima bactracking dari *root* hingga ke *leaf*. *Node-node* pada *tree* tersebut, merupakan perwujudan dari semua kategori, block, dan volume block yang ada. Setiap *parent node* memiliki lima *child node.*

Pergerakan dari *parent node* ke *child node* pada penelusuran *tree* dari *root* hingga ke *leaf* yang dilakukan pada awal dimulainya pencarian, diputuskan dengan menggunakan algoritma ant colony. Pergerakan yang dilakukan oleh algoritma ant colony tersebut, hanya dilakukan satu kali saja pada satu siklus pencarian hingga ditemukan satu satuan volume ruang terkecil yang kosong.

Terdapat dua kemungkinan yang ada ketika pergerakan yang dilakukan oleh algoritma ant colony telah selesai, pertama, ditemukan ruang kosong yang cukup untuk meletakkan satu buah barang, kedua, ditemukan ruang kosong tetapi tidak cukup untuk

meletakkan satu buah barang. Jika kemungkinan pertama terjadi, maka pencarian ruang untuk meletakkan satu buah barang telah selesai. Akan tetapi, jika kemungkinan kedua yang terjadi, maka dilakukan penelusuran balik dengan menggunakan algoritma backtracking hingga ditemukan ruang yang cukup untuk meletakkan satu buah barang.

Sebelum dilakukan pembangunan *tree*, diperlukan input berupa kategori-kategori dan block-block. Pada input yang berupa block terdapat input nama block, ukuran dari block yang terdiri dari panjang, lebar, dan tinggi, dan id kategori dari block tersebut. Pada aplikasi ini juga terdapat input barang. Pada input barang terdapat input nama barang, ukuran dari barang yang terdiri dari panjang, lebar, dan tinggi, dan id kategori dari barang tersebut. Untuk mendukung aplikasi yang dibuat, terdapat juga input pengguna. Pada input pengguna terdapat input username dan password.

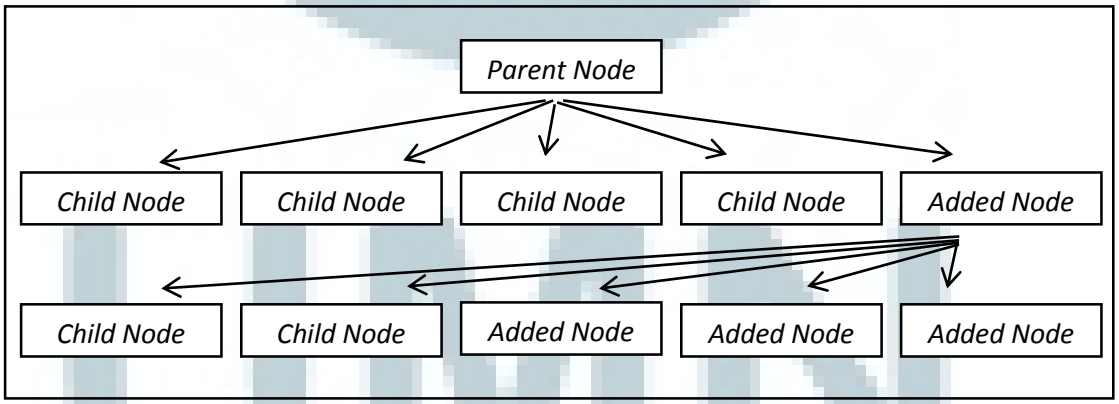
3.2.1 Flowchart Pembangunan Tree

Tree dibangun berdasarkan prinsip algoritma backtracking. *Node-node* yang terdapat pada *tree*, merupakan perwujudan dari semua kategori, block, dan volume block yang ada. Oleh karena itu, sebelum dilakukan pembangunan *tree*, diperlukan input berupa kategori-kategori dan block-block. Pada input yang berupa block terdapat input nama block, ukuran dari block yang terdiri dari panjang, lebar, dan tinggi, dan id kategori dari block tersebut. Jika kedua input tersebut telah dilakukan, maka akan dilakukan proses pembangunan tree. Setiap *parent node* memiliki lima *child node.*

Masing-masing *node* pada *tree* menyimpan variabel nama *node*, nilai *node* yang hanya bisa memiliki dua nilai yaitu nol jika belum penuh terisi barang dan satu jika

sudah penuh terisi barang, intensitas *pheromone trail* untuk antar state (T_{ij}), visibilitas dari suatu solusi yang akan dipilih oleh semut (η_{ij}) dan jumlah *pheromone trail* yang diendapkan (ΔT_{ij}). Pada awal pembuatan *node,* dilakukan inisialisasi variabel nama *node*, nilai *node*, T_{ij}, dan ΔT_{ij} dengan nilai nol, dan η_{ij}.

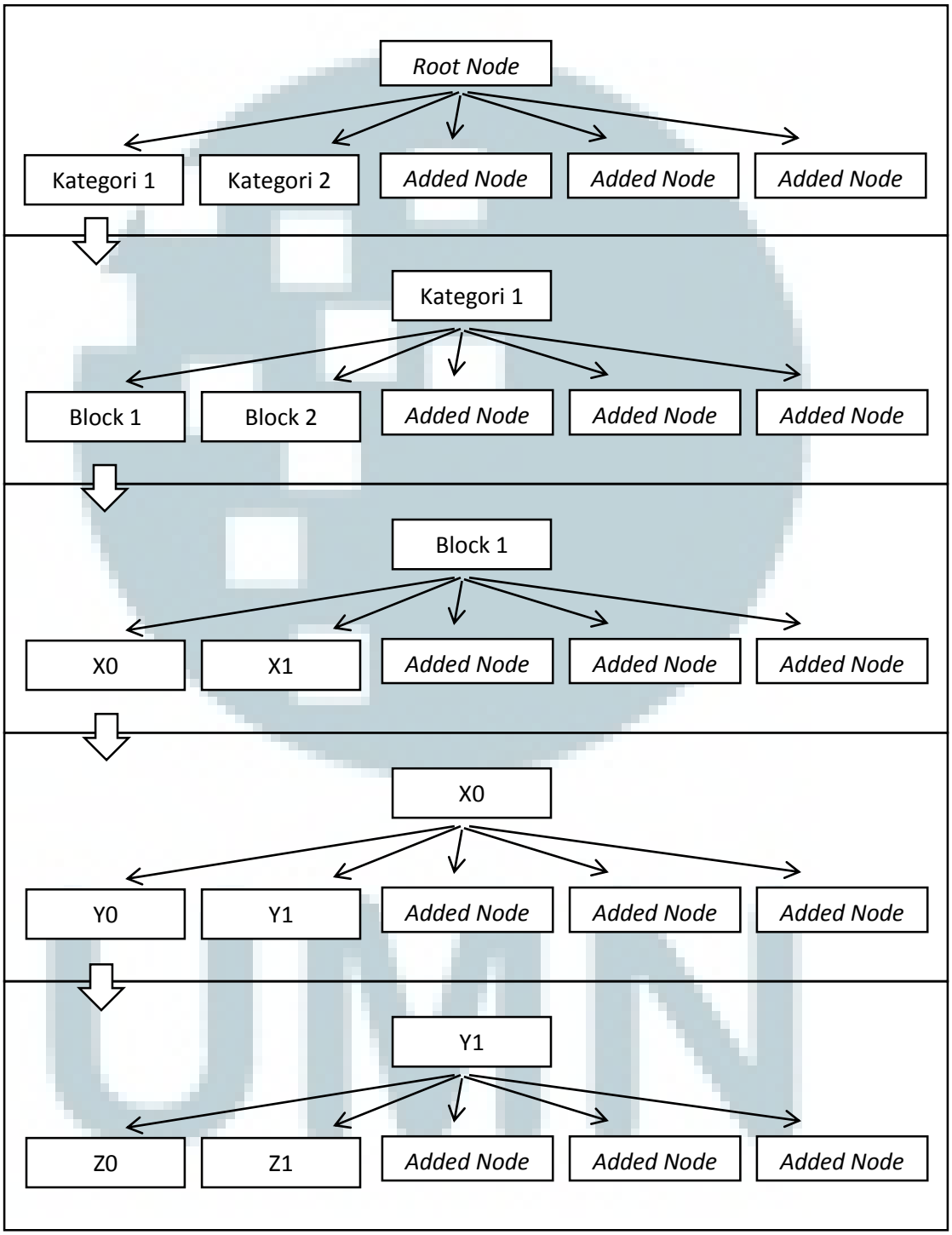
Untuk menyimpan *parent node* dan *child node*, terdapat variabel untuk menyimpan *parent node, child node* pertama atau *left child node, child node* kedua atau *middleleft child node, child node* ketiga atau *middle child node, child node* keempat atau *middleright child node,* dan *child node* kelima atau *right child node*. Jika jumlah *child node* kurang dari lima, maka ditambahkan suatu *added node* hingga genap menjadi lima *child node*. Jika jumlah *child node* lebih dari kelipatan lima, maka akan dibuat empat buah *child node* dan satu *added node,* kemudian, *added node* bertidak sebagai *parent node* dan sisa dari *child node* yang ada akan menjadi *child node* dari *parent node* tersebut, dst.



Gambar 3.1 Penyusunan *Child Node*

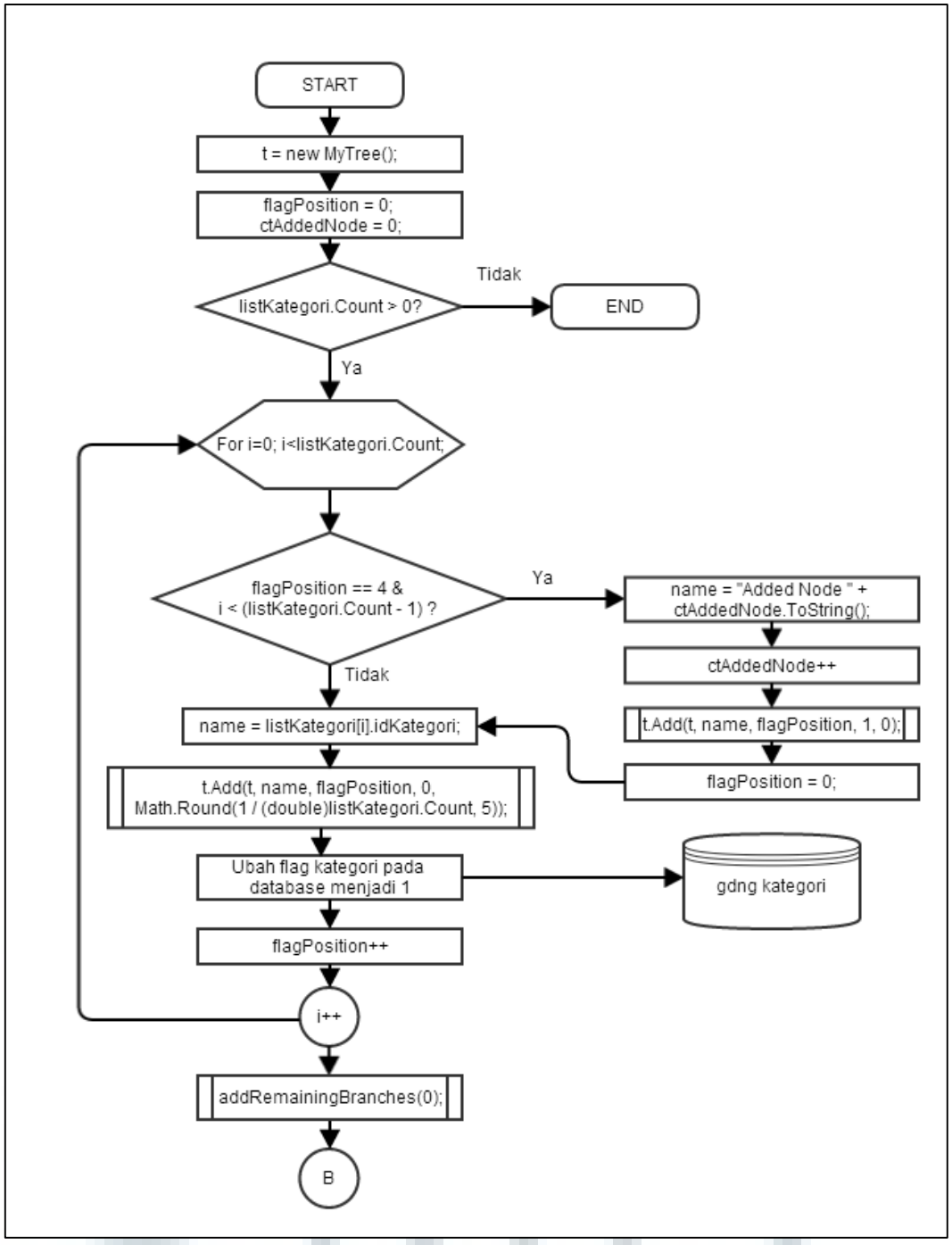
Terdapat enam tingkatan *tree* dilihat dari jenis *node*-nya dan cara penyusunannya (dengan menggunakan metode block stacking), secara urut atas ke

bawah yaitu *root node*, *node* kategori, *node* block, *node* xBlock atau panjang block, *node* yBlock atau tinggi block, dan *node* zBlock atau lebar block.

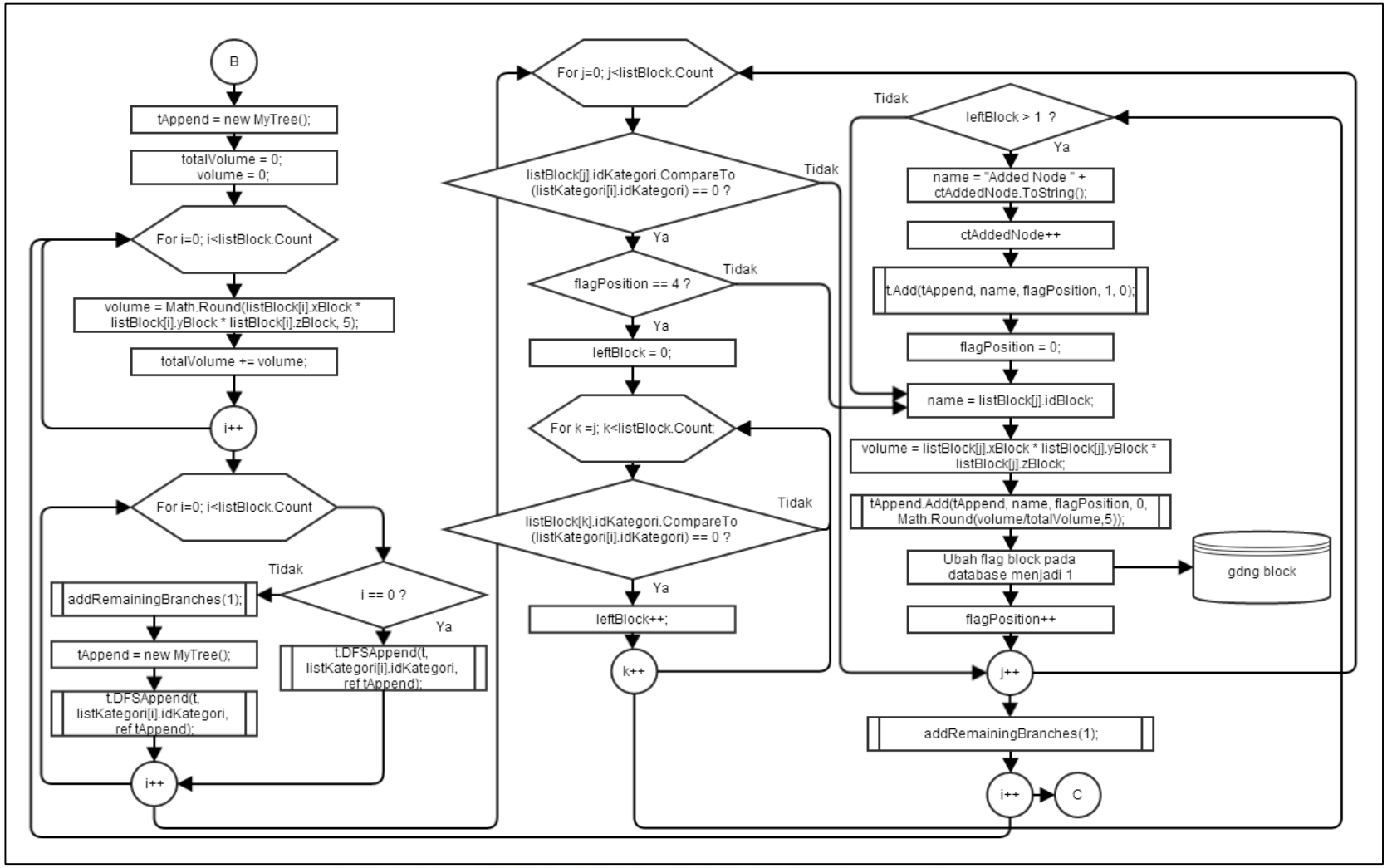


Gambar 3.2 Enam Tingkatan *Tree* Dilihat dari Jenis *Node*-nya dan Cara Penyusunannya (dengan menggunakan metode *block stacking*)

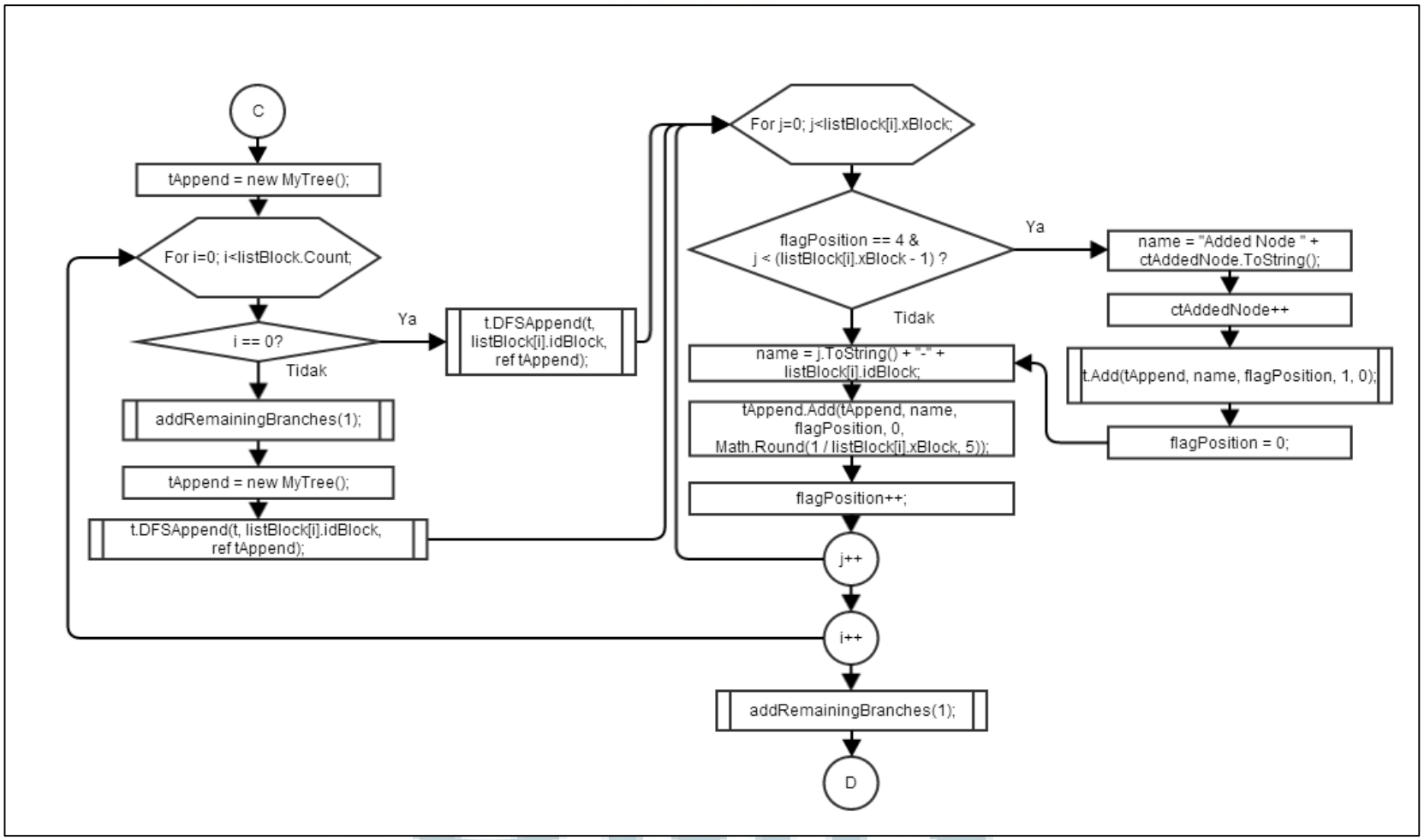
Berikut merupakan flowchart dari pembangunan *tree*.



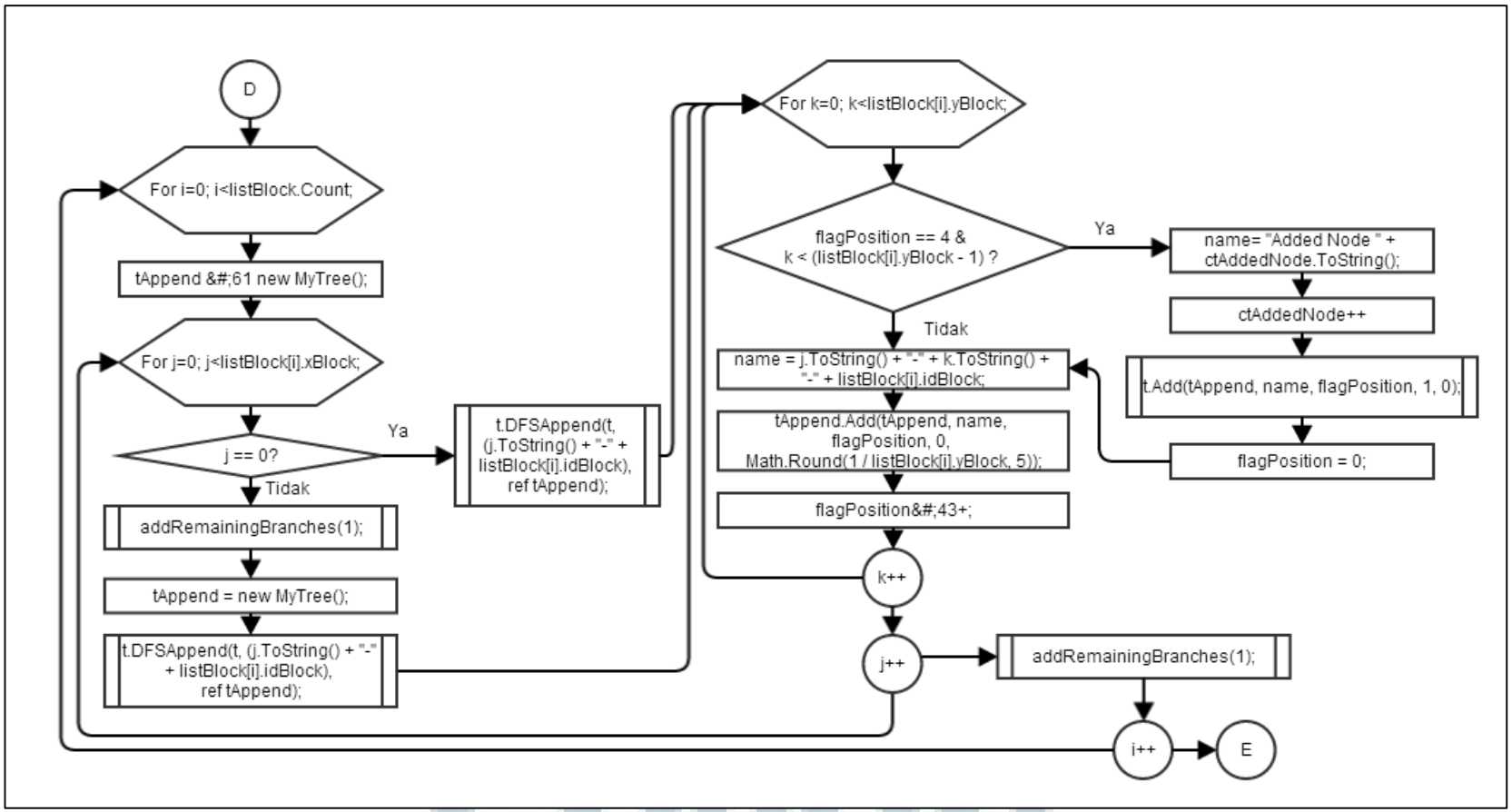
Gambar 3.3 Flowchart Pembangunan



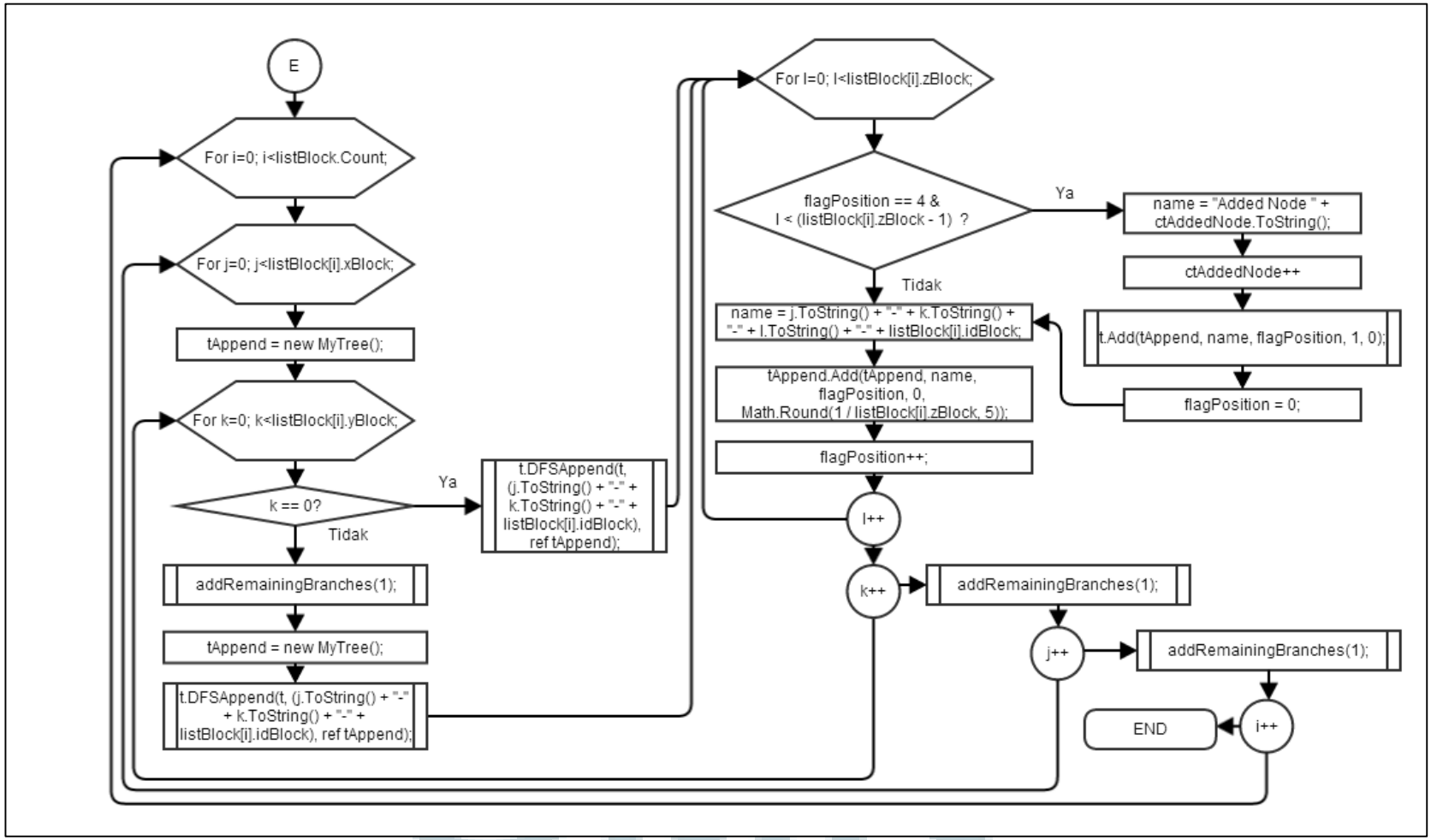
Gambar 3.3 Flowchart Pembangunan Tree (lanjutan)



Gambar 3.3 Flowchart Pembangunan (lanjutan)



Gambar 3.3 Flowchart Pembangunan Tree (lanjutan)



Gambar 3.3 Flowchart Pembangunan Tree (lanjutan)

3.2.2 Flowchart Algoritma Ant Colony

Pergerakan dari *parent node* ke *child node* pada penelusuran *tree* dari *root* hingga ke *leaf* yang dilakukan pada awal dimulainya pencarian, diputuskan dengan menggunakan algoritma ant colony. Pergerakan yang dilakukan oleh algoritma ant colony tersebut hanya dilakukan satu kali saja pada satu siklus pencarian hingga ditemukan satu satuan volume ruang terkecil yang kosong (*leaf node* atau solusi). Kemudian, dilakukan validasi apakah solusi yang didapatkan dari penggunaan algoritma ant colony benar-benar solusi yang dicari atau tidak. Solusi yang dicari yaitu solusi yang menyatakan terdapat ruang kosong yang cukup untuk meletakkan satu buah barang.

Satu barang yang dimasukkan dianggap sebagai seekor semut. Seekor semut akan bergerak dari *node* i (*parent node*) ke *node* j (*child node*) dengan probabilitas:

$$\boldsymbol{P_{ij}} = \frac{(\boldsymbol{T_{ij}^{\alpha}})(\boldsymbol{\eta_{ij}^{\beta}})}{\sum(\boldsymbol{T_{ij}^{\alpha}})(\boldsymbol{\eta_{ij}^{\beta}})} \quad \text{..... Rumus 3.1}$$

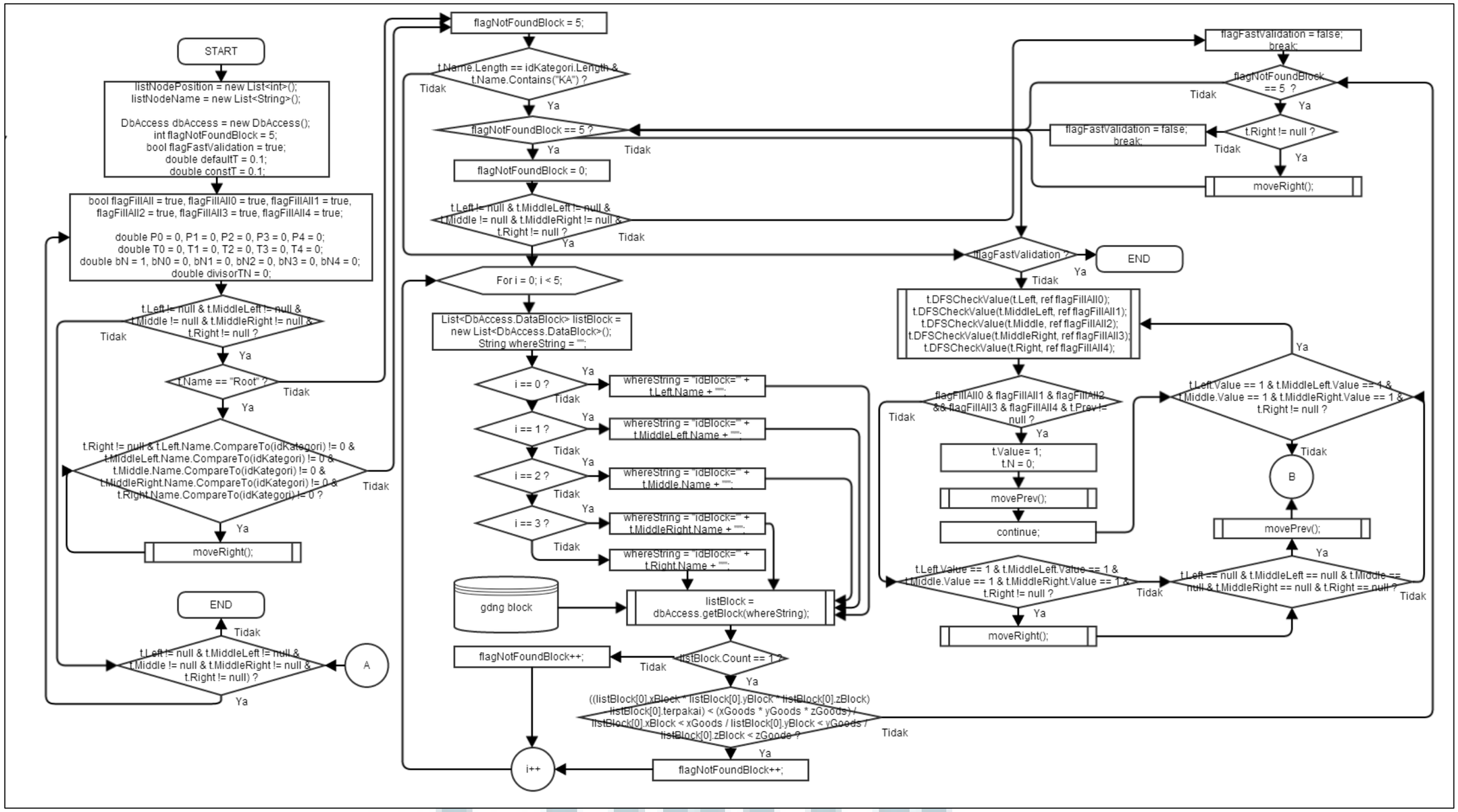
Sebelum dilakukan pergerakan dari *current node* ke salah satu *child node*-nya, demi mengemat waktu pemrosesan, maka dilakukan pengecekkan terlebih dahulu terhadap kelima *child node* apakah kelima *child node* dan *child node* dari *child node* tersebut, dst, memiliki ruang kosong atau tidak. Jika tidak, maka nilai *node* dan nilai $\boldsymbol{\eta_{ij}}$ *node* dari *current node* diubah menjadi nol dan naik ke *parent node* dari *current node*. Dengan diubahnya nilai $\boldsymbol{\eta_{ij}}$ *node* dari *current node*, maka probabilitas terpilihnya *node* tersebut menjadi lebih kecil.

Untuk pergerakan semut dari *root node* ke *node* kategori, dilakukan penambahan nilai $\boldsymbol{\eta_{ij}}$ *node* kategori senilai satu jika barang yang masuk memiliki

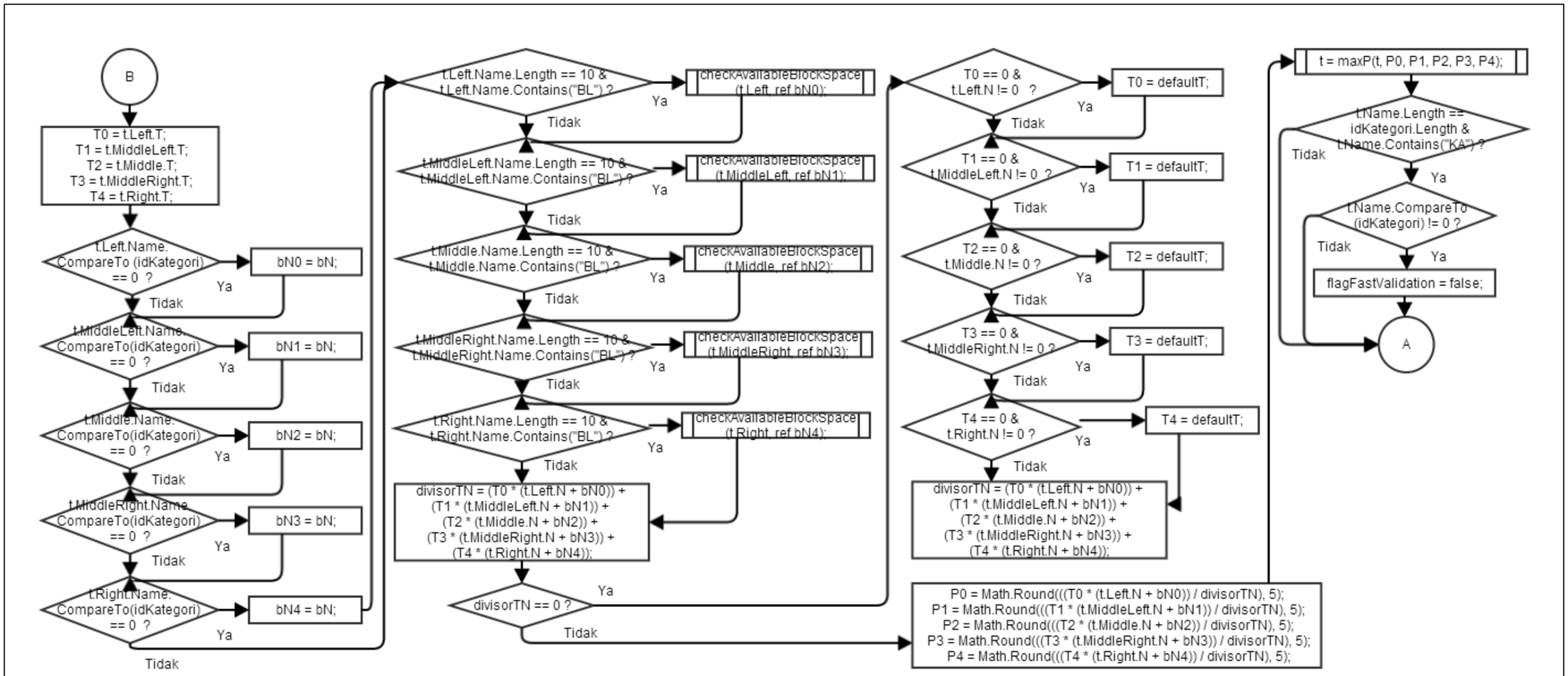
kategori yang sama dengan *node* kategori tersebut. Jika kategori barang tidak sesuai dengan semua *node* kategori pada tingkat pertama, maka secara otomatis semut bergerak ke *parent node* (*added node*) dari *node* kategori pada tingkat kedua, dst.

Untuk pergerakan semut dari *node* kategori ke *node* block, dilakukan pengurangan nilai $\boldsymbol{\eta_{ij}}$ *node* block hingga menjadi nol jika barang yang masuk memiliki ukuran (baik panjang, lebar, maupun tinggi) atau volume yang lebih besar dari ukuran atau volume *node* block tersebut. Jika ukuran atau volume barang tidak sesuai dengan semua ukuran atau volume *node* block pada tingkat pertama, maka secara otomatis semut bergerak ke *parent node* (*added node*) dari *node* block pada tingkat kedua, dst. Jika tidak terdapat ruang pada semua *node* block yang memiliki kategori yang sama, maka pada akhirnya semut bergerak menuju node kategori yang memiliki kategori yang berbeda dari kategori barang yang dimasukkan. Bila hal tersebut terjadi, maka proses pencarian ruang dihentikan (ruang tidak ditemukan).

Jika divisor bernilai nol (akibat dari $\boldsymbol{T_{ij}}$ semua child node bernilai nol), maka hitung ulang nilai dari divisor dengan memberi nilai $\boldsymbol{T_{ij}}$ semua child node dengan nilai default dari $\boldsymbol{T_{ij}}$ yang telah ditetapkan, yaitu 0,1.



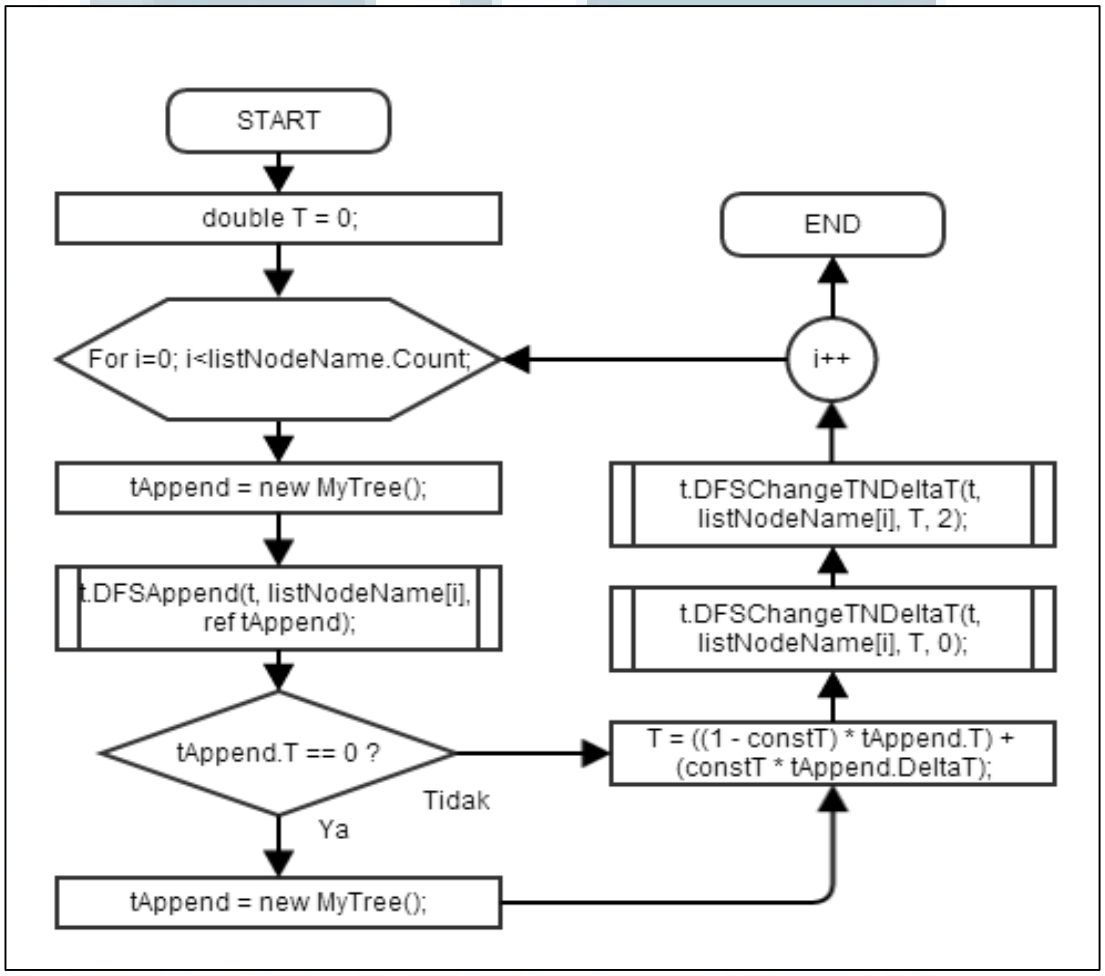
Gambar 3.4 Flowchart Algoritma Ant Colony



Gambar 3.4 Flowchart Algoritma Ant Colony (lanjutan)

UMN

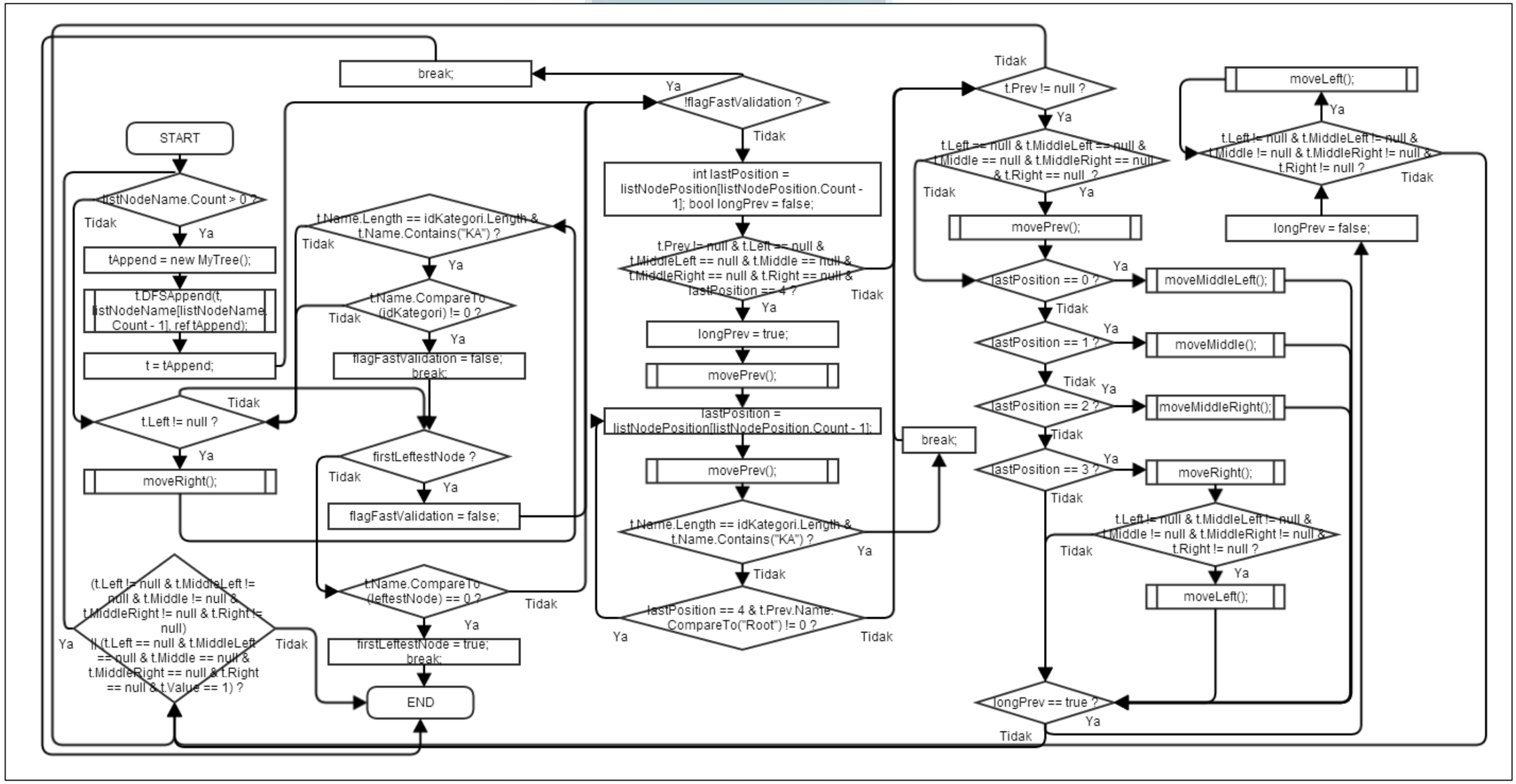
Jika ditemukan ruang kosong yang cukup untuk meletakkan satu buah barang, maka intensitas *pheromone trail* untuk antar *state* (T_{ij}) dan jumlah *pheromone trail* yang diendapkan (ΔT_{ij}) untuk setiap *edge* yang dilewati dari *root node* hingga ditemukan sebuah *leaf node* di-*update*. Selain itu, update *value* dengan nilai satu dan visibilitas dari suatu solusi yang akan dipilih oleh semut (η_{ij}) dengan nilai nol untuk semua *leaf node* yang menggambarkan satu satuan volume terkecil penyimpanan yang telah terisi barang.



Gambar 3.4 Flowchart Algoritma Ant Colony
(*global pheromone trail update*) (lanjutan)

3.2.3 Flowchart Algoritma Backtracking

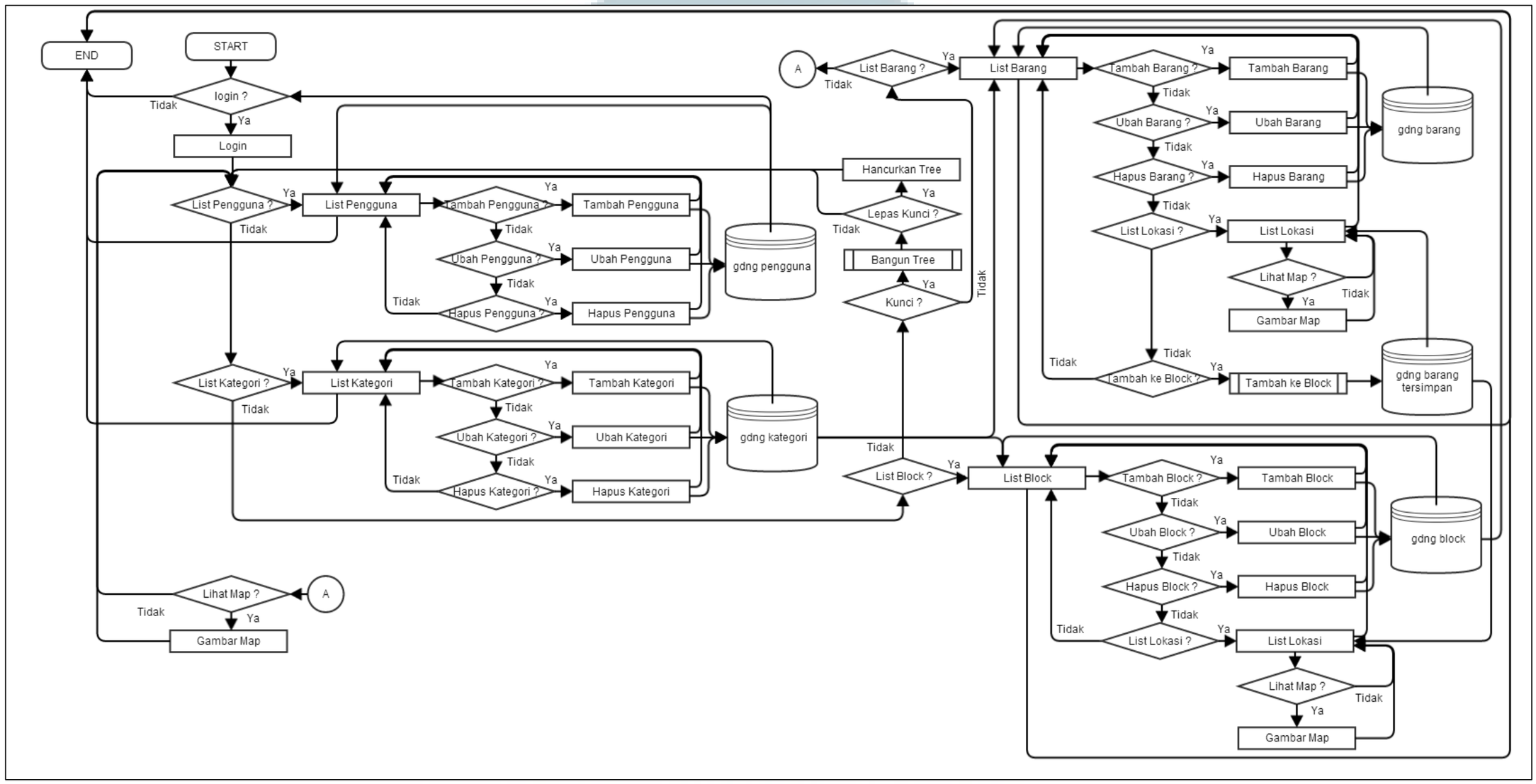
Jika solusi yang didapatkan dari penggunaan algoritma ant colony menyatakan tidak ditemukan ruang kosong yang cukup untuk meletakkan satu buah barang, maka dilakukan backtracking dengan *leaf node* atau solusi yang didapatkan dari pemrosesan dengan menggunakan algoritma ant colony tersebut sebagai titik awal. Backtracking dilakukan hingga didapatkan ruang kosong yang cukup untuk meletakkan satu buah barang atau semua node yang berada pada kategori yang sama telah dikunjungi (tidak terdapat ruang kosong yang cukup untuk meletakkan satu buah barang).



Gambar 3.5 Flowchart Algoritma Backtracking

3.2.4 Flowchart Aplikasi Pengaturan Tata Letak Barang

Fungsi utama dari aplikasi ini yaitu menentukan tata letak barang. Hal-hal yang dapat dilakukan pengguna dalam aplikasi ini yaitu login, logout, mengunci dan melepas kunci pengaturan kategori dan block, melihat *list*, menambah, mengubah, dan menghapus data pengguna (khusus pengguna dengan tipe pengguna superadmin), kategori, block, dan barang, melihat *list* lokasi, melihat *map*, dan menambah data barang tersimpan.



Gambar 3.6 Flowchart Aplikasi Pengaturan Tata Letak Barang

3.2.5 Rancangan Penggunaan Algoritma Ant Colony dan Backtracking

Berikut merupakan rancangan penggunaan algoritma ant colony dan backtracking dalam aplikasi pengaturan tata letak barang.

A. Struktur Pohon Solusi Berdasarkan Prinsip Algoritma Backtracking

Untuk mendapatkan solusi-solusi yang memiliki kemungkinan untuk diambil, dibangun sebuah *tree* dari *root* hingga ke *leaf*, dimana *leaf node* merupakan salah satu solusi tersebut. Solusi-solusi yang ada berupa sebuah satuan ukuran volume terkecil ruang pada block tertentu dan kategori tertentu yang dapat ditempati oleh barang. Faktor-faktor penting yang berdampak pada penyusunan struktur *tree*:

1) Kategori barang

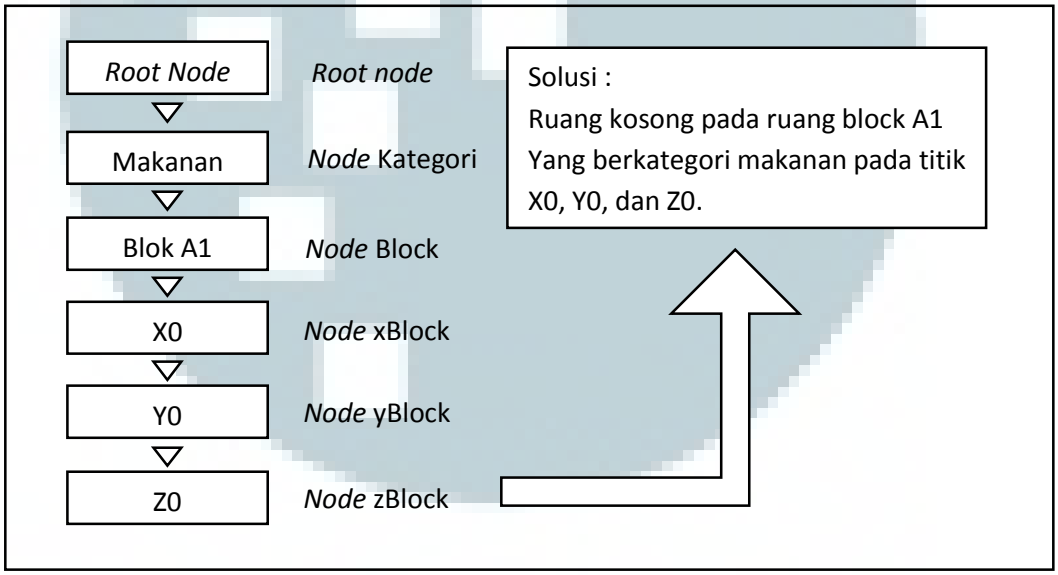
   Kategori barang menjadi *node* yang berada pada tingkat dua dari *tree* (*node* kategori) (setelah node tingkat satu (*root node*)). Satu kategori barang digambarkan menjadi satu buah *node* kategori.

2) Block

   Block menjadi *node* yang berada pada tingkat tiga dari tree (node block). Satu block digambarkan menjadi satu buah *node* block. Di bawah *node* block, terdapat *node* xBlock yang merupakan *node* tingkat empat dan merepresentasikan panjang block. Di bawah *node* xBlock, terdapat *node* yBlock yang merupakan *node* tingkat lima dan merepresentasikan tinggi block. *Node* yang berada pada posisi paling bawah (*leaf node*) yaitu zBlock. *Node* zBlock merupakan *node* tingkat enam yang merepresentasikan lebar block.

Panjang, tinggi, dan lebar dari block merupakan kumpulan satuan ukuran terkecil, yaitu 10cm. Satu ukuran terkecil yang terdapat pada panjang block menjadi satu buah *node* xBlock, satu ukuran terkecil yang terdapat pada tinggi block menjadi satu buah *node* yBlock, dan satu ukuran terkecil yang terdapat pada lebar block menjadi satu buah *node* zBlock.

Berikut merupakan visualisasi tingkatan *node* pada potongan *tree* hingga ditemukan sebuah solusi:



Gambar 3.7 Visualisasi Tingkatan *Node* pada Potongan *Tree* hingga Ditemukan Satu Solusi

B. Inisialisasi Variable-Variable Algoritma Ant Colony

Algoritma ant colony yang digunakan dalam aplikasi ini membutuhkan inisialisasi variabel-variabel, yaitu sebagai berikut.

1) Intensitas *pheromone trail* untuk antar *state* (T_{ij}).
2) Visibilitas dari suatu solusi yang akan dipilih oleh semut (η_{ij}).

3) Parameter pengendali intensitas *pheromone trail* (α), nilai $\alpha \geq 0$.

4) Parameter pengendali visibilitas (β), nilai $\beta \geq 0$.

5) Banyak *state* (n).

6) Tetapan penguapan *pheromone trail* (ρ), nilai $0 > \rho > 1$.

$\boldsymbol{T_{ij}}$ masing-masing *edge* yang menuju ke *node* diinisialisasi dengan nilai nol dan nilainya disimpan pada *node* yang dilakukan pada saat pembangunan *tree*. $\boldsymbol{\eta_{ij}}$ *root node* dan *added node* diinisialisasi dengan nilai nol dan $\boldsymbol{\eta_{ij}}$ selain *root node* dan *added node* diinisialisasi dengan menggunakan probabilitas kemungkinan terhadap total jumlah yang ada, yaitu sebagai berikut.

1) *Node* Kategori

$$\boldsymbol{\eta_{ij}}_{\boldsymbol{x}} = \frac{\mathbf{1}}{\boldsymbol{totalKategori}} \quad \text{..... Rumus 3.2}$$

2) *Node* Block

$$\boldsymbol{\eta_{ij}}_{\boldsymbol{x}} = \frac{\boldsymbol{volumeBlock_x}}{\sum_{\boldsymbol{x=0}}^{\boldsymbol{x=y}} \boldsymbol{volumeBlock_{x*}}} \quad \text{..... Rumus 3.3}$$

Keterangan,

* = volume block yang memiliki kategori yang sama.

3) Node xBlock / panjang block

$$\boldsymbol{\eta_{ij}}_{\boldsymbol{x}} = \frac{\mathbf{1}}{\boldsymbol{panjangBlock}} \quad \text{..... Rumus 3.4}$$

4) Node yBlock / tinggi block

$$\boldsymbol{\eta_{ij}}_{\boldsymbol{x}} = \frac{\mathbf{1}}{\boldsymbol{tinggiBlock}} \quad \text{..... Rumus 3.5}$$

5) Node zBlock / lebar block

$$\eta_{ij_x} = \frac{1}{lebarBlock}$$ ..... Rumus 3.6

α dan β diinisialisasi dengan nilai 1, n merupakan jumlah *node* yang ada pada *tree*, dan ρ diinisialisasi dengan nilai 0,1. Hanya terdapat satu *pheromone trail update* saja (*global pheromone trail update*) dalam penggunaan algoritma ant colony pada aplikasi ini.

C. Input dan Output Pencarian Solusi

Masukkan berupa sebuah barang dengan membawa informasi kategori barang, panjang barang, tinggi barang, dan lebar barang. Setelah dilakukan pencarian solusi dengan menggunakan kombinasi algoritma ant colony dan backtracking, terdapat dua jenis keluaran, yaitu tersedia ruang kosong untuk meletakkan barang dan tidak tersedia ruang kosong untuk meletakkan barang. Untuk solusi tersedia ruang kosong untuk meletakkan barang, terdapat informasi berupa blok beserta titik awal dan akhir peletakkannya.

Contoh:

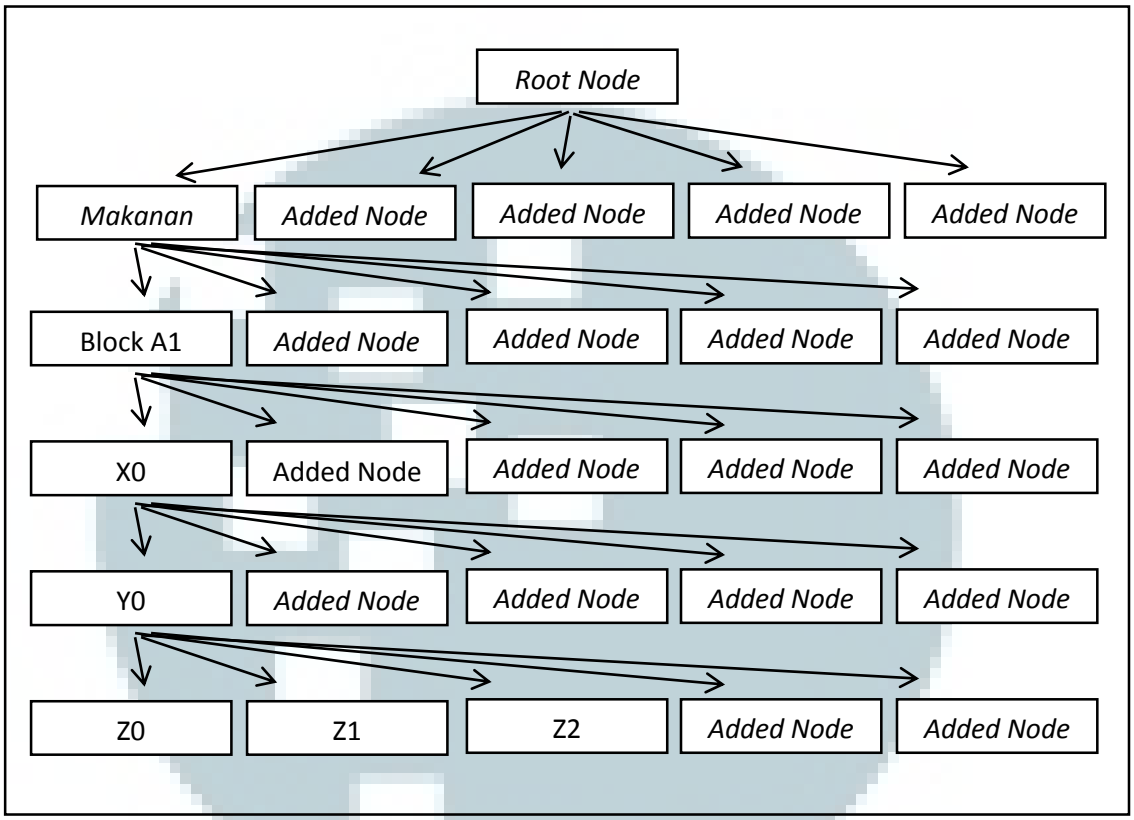
Input:

1) Barang A berkategori makanan, panjang 10 cm, tinggi 10 cm, dan lebar 20 cm.
2) Barang B berkategori makanan, panjang 10 cm, tinggi 10 cm, dan lebar 30 cm.

Faktor-faktor penting yang berdampak pada penyusunan struktur *tree*:

1) Satu kategori: Makanan.
2) Satu block: Block A1, panjang 10 cm, tinggi 10 cm, lebar 30 cm.

Dari faktor-faktor tersebut, maka dibangunlah tree sebagai berikut.



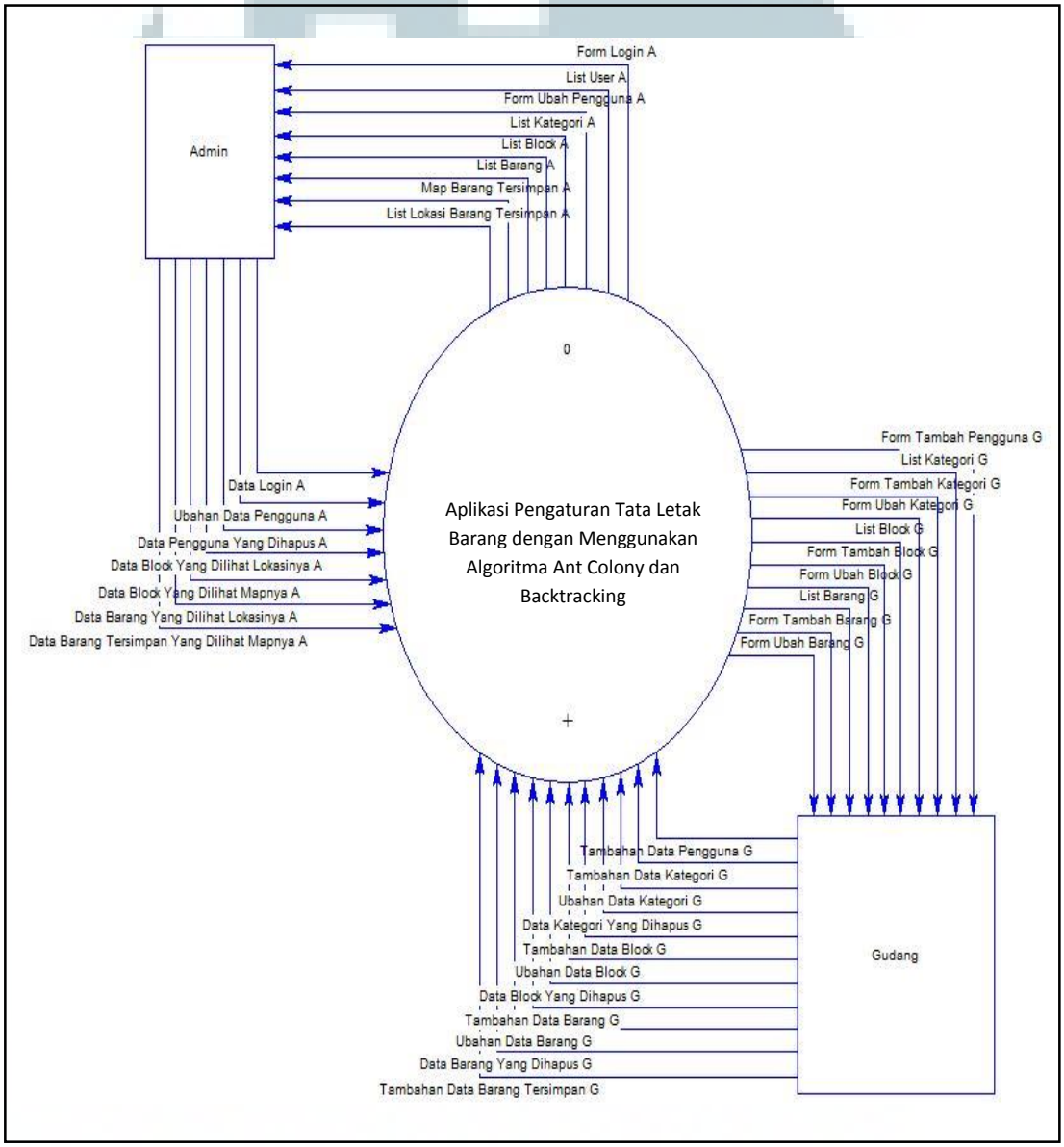
Gambar 3.8 Contoh *Tree*

Output:

1) Tersedia ruang kosong untuk meletakkan barang, Blok A1, X Awal: X0, X Akhir: X0, Y Awal: Y0, Y Akhir: Y0, Z Awal: Z0, Z Akhir: Z1.
2) Tidak tersedia ruang kosong untuk meletakkan barang. Hal ini disebabkan karena hanya tersisa ruang pada satu titik saja (X0, Y0, Z2) setelah barang A berhasil diletakkan.

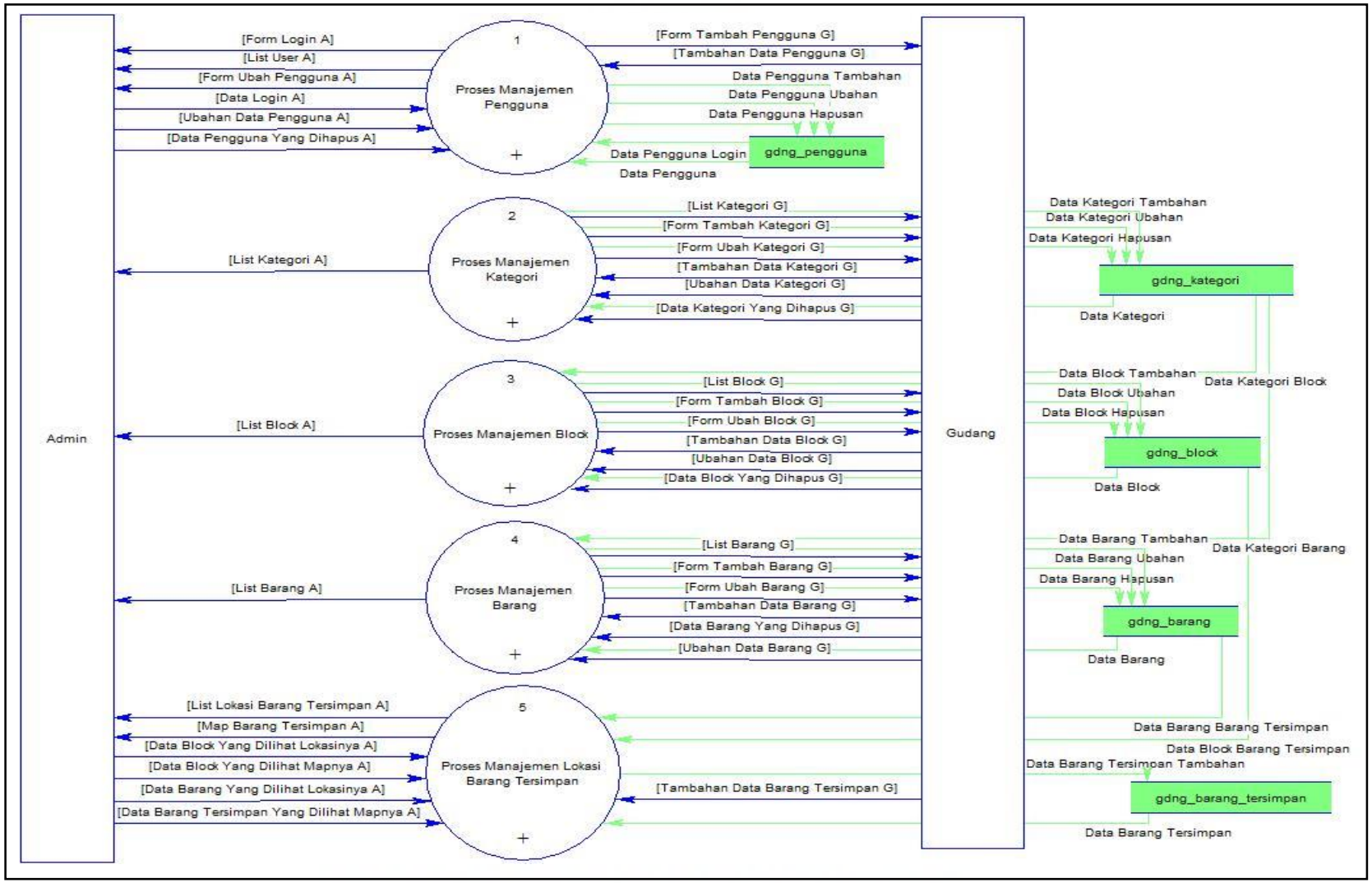
3.2.6 Data Flow Diagram

Pada c*ontext diagram* terdapat dua entitas yaitu *superadmin* dan *admin* dengan *data flow* menuju ke proses atau input yang diberikan kepada proses, dan *data flow* yang menuju ke entitas atau baik data maupun *feedback* yang diberikan kepada entitas. Terdapat satu proses utama, yaitu sistem pengaturan tata letak barang.



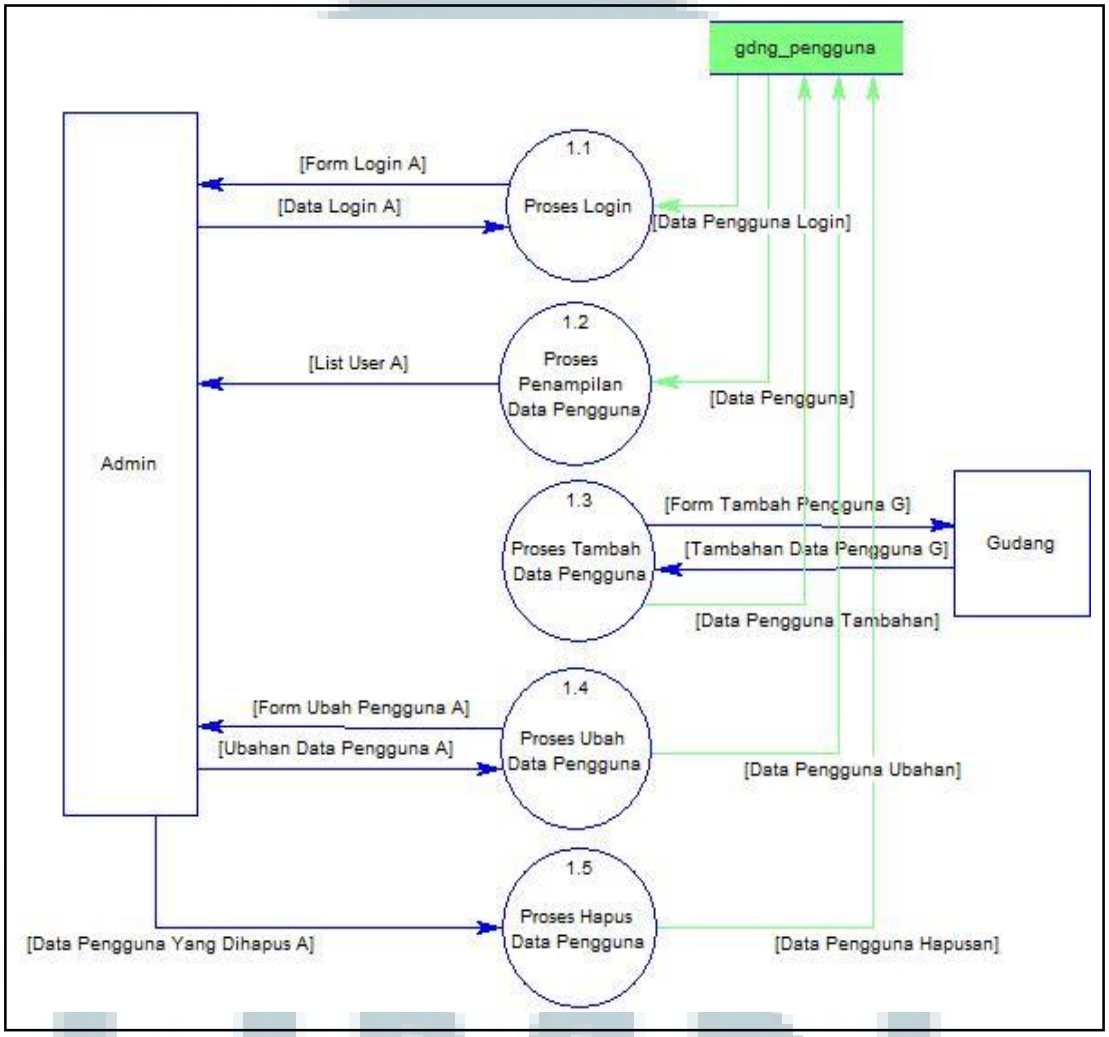
Gambar 3.9 DFD *Context Diagram* Aplikasi Pengaturan Tata Letak Barang

Context Diagram dirincikan menjadi DFD level 1, dimana terdapat lima subproses yaitu proses manajemen pengguna, proses manajemen kategori, proses manajemen block, proses manajemen barang, dan proses manajemen lokasi barang tersimpan. Pada DFD level 1 terdapat lima database yaitu gdng_pengguna, gdng_kategori, gdng_block, gdng_barang, dan gdng_barang_tersimpan. Selain itu, terdapat data *flow* yang yang lebih terperinci arahnya ke subproses-subproses, baik data *flow* yang menuju *database* maupun entitas.



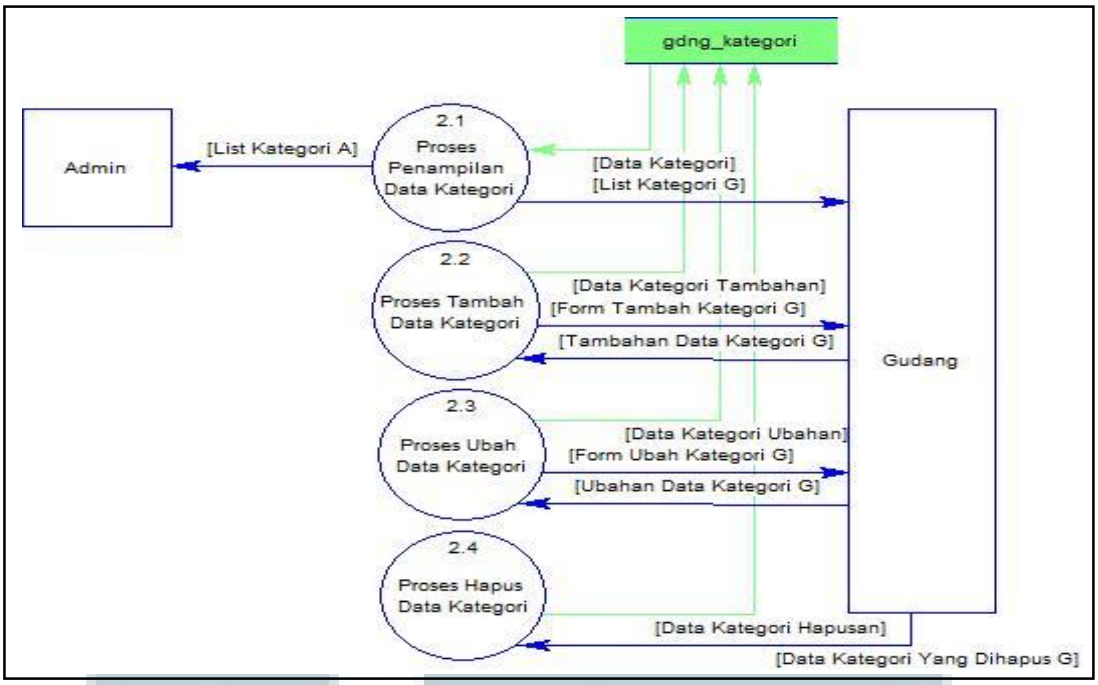
Gambar 3.10 DFD Level 1 Aplikasi Pengaturan Tata Letak Barang

Pada proses manajemen pengguna, terdapat proses login, penampilan data pengguna, tambah data pengguna, ubah data pengguna, dan hapus data pengguna. Terdapat database gdng_pengguna.



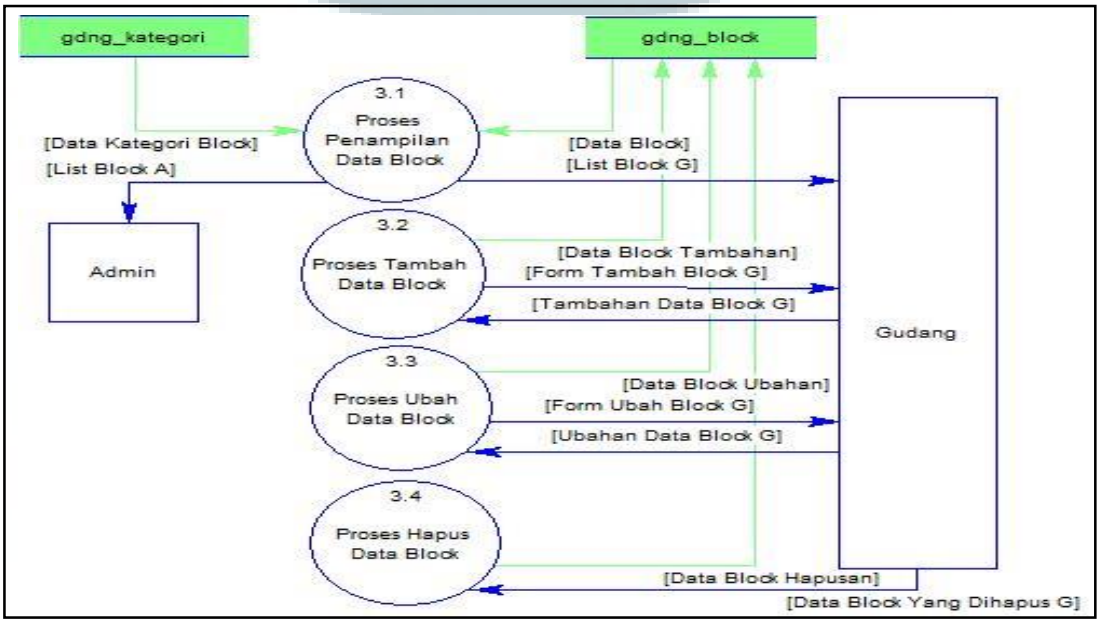
Gambar 3.11 DFD Level 2 Proses Manajemen Pengguna

Pada proses manajemen kategori, terdapat proses penampilan data kategori, tambah data kategori, ubah data kategori, dan hapus data kategori. Terdapat database gdng_kategori.



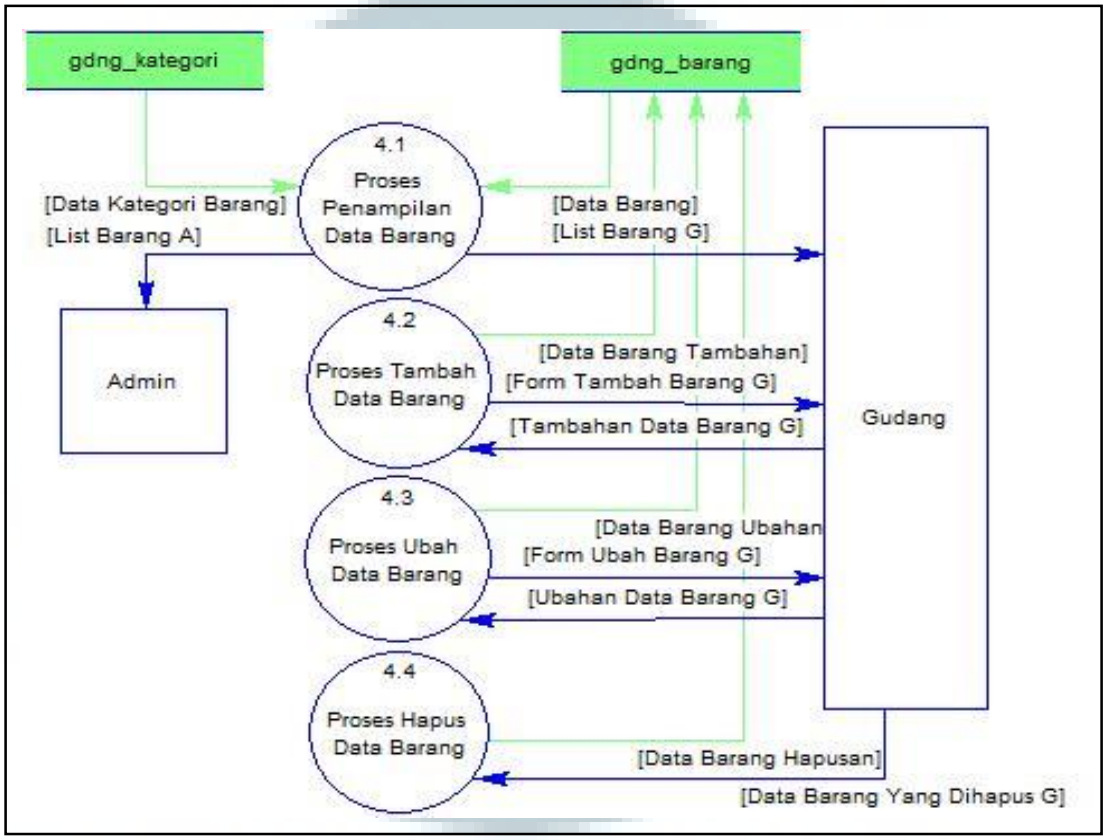
Gambar 3.12 DFD Level 2 Proses Manajemen Kategori

Pada proses manajemen block, terdapat proses penampilan data block, tambah data block, ubah data block, dan hapus data block. Terdapat database gdng_kategori dan gdng_block.



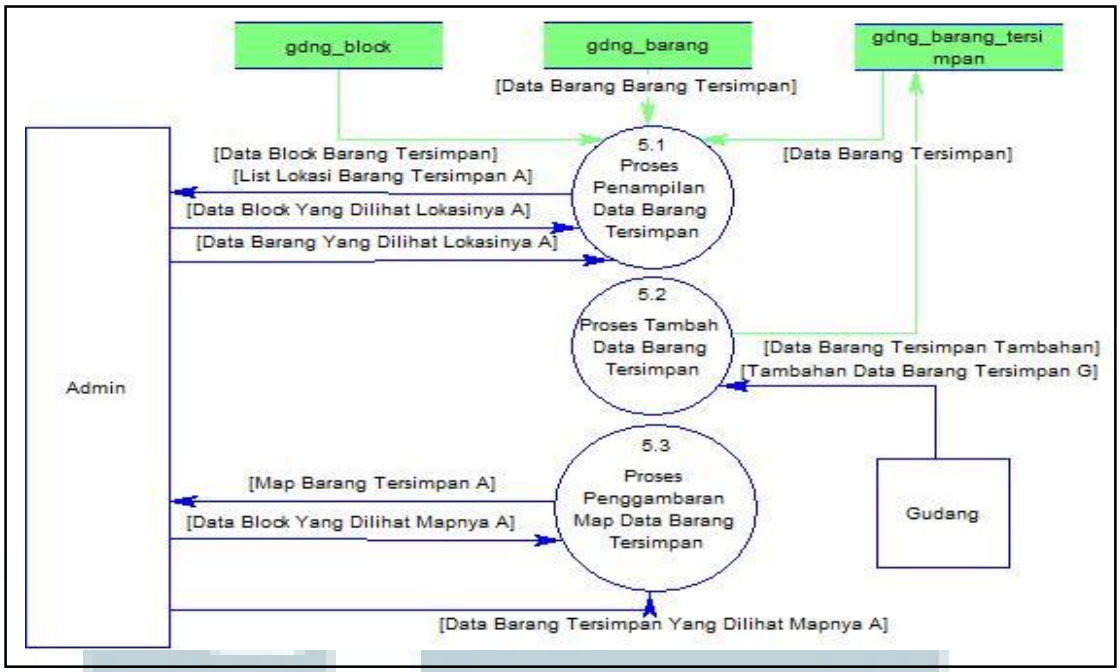
Gambar 3.13 DFD Level 2 Proses Manajemen Block

Pada proses manajemen barang, terdapat proses penampilan data barang, tambah data barang, ubah data barang, dan hapus data barang. Terdapat database gdng_kategori dan gdng_barang.



Gambar 3.14 DFD Level 2 Proses Manajemen Barang

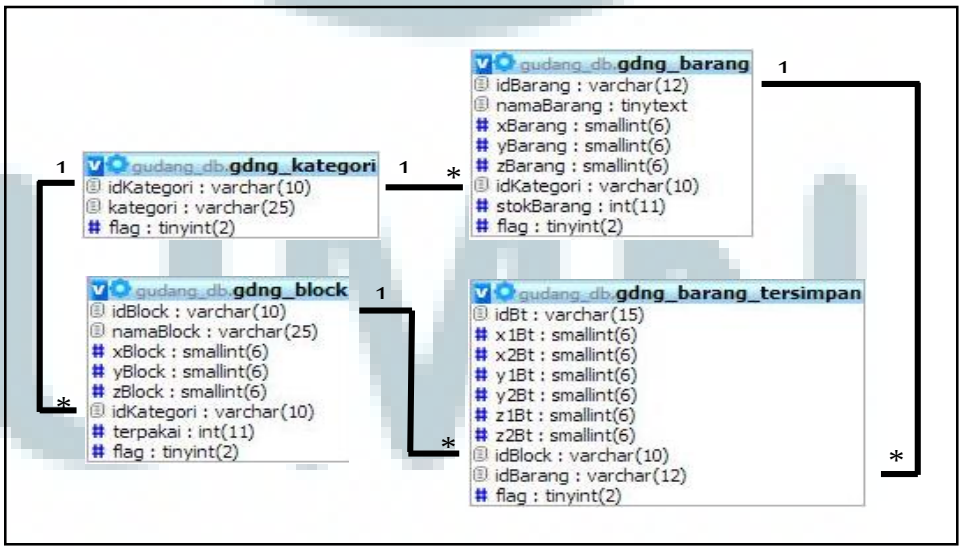
Pada proses manajemen lokasi barang tersimpan, terdapat proses penampilan data barang tersimpan, tambah data barang tersimpan, dan penggambaran *map* data barang tersimpan. Terdapat database gdng_kategori, gdng_barang, dan gdng_barang_tersimpan.



Gambar 3.15 DFD Level 2 Proses Manajemen Lokasi Barang Tersimpan

3.2.7 Entity Relationship Diagram

Berikut ini merupakan *Entity Relationship Diagram* dari aplikasi pengaturan tata letak barang.



Gambar 3.16 ERD Aplikasi Pengaturan Tata Letak Barang

3.2.8 Struktur Tabel

Database yang digunakan dalam aplikasi pengaturan tata letak barang ini adalah MySQL. Tabel-tabel yang ada pada database yaitu sebagai berikut.

Nama table : gdng_pengguna

Fungsi : Tabel ini digunakan untuk menyimpan data pengguna

Tabel 3.1 Struktur Tabel gdng_pengguna

Nama Field	**Tipe Data**	**Panjang**	**Keterangan**
idPengguna	varchar	10	*Primary key*
username	varchar	25	
password	varchar	32	
tipePengguna	varchar	15	
flag	tinyint		*Default*: 0

Nama table : gdng_kategori

Fungsi : Tabel ini digunakan untuk menyimpan data kategori

Tabel 3.2 Struktur Tabel gdng_kategori

Nama Field	**Tipe Data**	**Panjang**	**Keterangan**
idKategori	varchar	10	*Primary key*
kategori	varchar	25	
flag	tinyint		*Default*: 0

Nama table : gdng_block

Fungsi : Tabel ini digunakan untuk menyimpan data block

Tabel 3.3 Struktur Tabel gdng_block

Nama Field	**Tipe Data**	**Panjang**	**Keterangan**
idBlock	varchar	10	*Primary key*
namaBlock	varchar	25	
xBlock	smallint		
yBlock	smallint		
zBlock	smallint		
idKategori	varchar	10	*Foreign key*: *reference* ke tabel gdng_kategori

Tabel 3.3 Struktur Tabel gdng_block (lanjutan)

Nama Field	Tipe Data	Panjang	Keterangan
terpakai	int		*Default*: 0
flag	tinyint		*Default*: 0

Nama table : gdng_barang

Fungsi : Tabel ini digunakan untuk menyimpan data barang

Tabel 3.4 Struktur Tabel gdng_barang

Nama Field	Tipe Data	Panjang	Keterangan
idBarang	varchar	12	*Primary key*
namaBarang	tinytext		
xBarang	smallint		
yBBarang	smallint		
zBBarang	smallint		
idKategori	varchar	10	*Foreign key*: *reference* ke tabel gdng_kategori
stokBarang	int		*Default*: 0
flag	tinyint		*Default*: 0

Nama table : gdng_barang_tersimpan

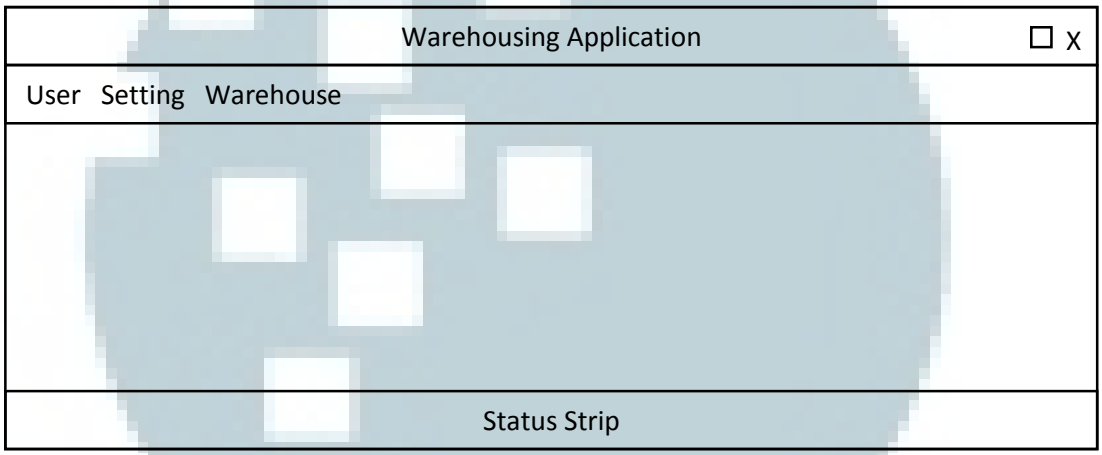
Fungsi : Tabel ini digunakan untuk menyimpan data barang tersimpan

Tabel 3.5 Struktur Tabel gdng_barang_tersimpan

Nama Field	Tipe Data	Panjang	Keterangan
idBt	varchar	15	*Primary key*
x1Bt	smallint		
x2Bt	smallint		
y1Bt	smallint		
y2Bt	smallint		
z1Bt	smallint		
z2Bt	smallint		
idBlock	varchar	10	*Foreign key*: *reference* ke tabel gdng_block
idBarang	varchar	12	*Foreign key*: *reference* ke tabel gdng_barang
flag	tinyint		*Default*: 0

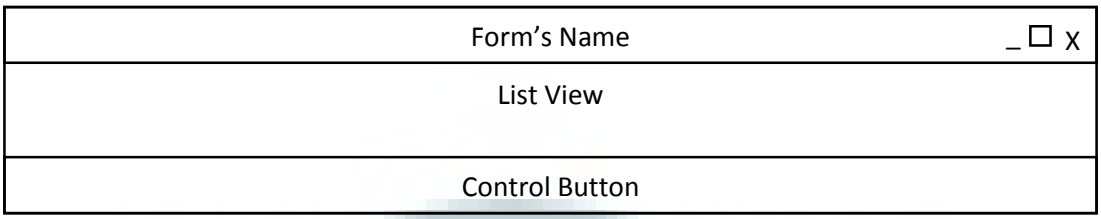
3.2.9 Rancangan Antarmuka

Terdapat lima *template form* utama pada perancangan *user-interface*. Kelima *template form* utama tersebut, yaitu *template form* halaman utama, *template form* untuk menampilkan data, *template form* untuk menambah atau mengubah data, *template form* untuk menampilkan lokasi, dan *template form* untuk menggambarkan lokasi.



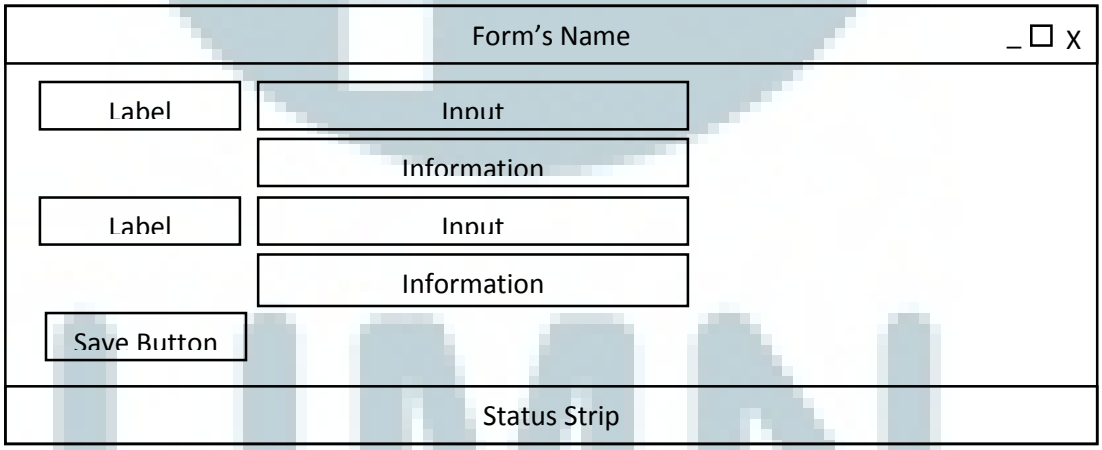
Gambar 3.17 *Template Form* Halaman Utama

Pada *template form* ini terdapat tiga menu utama, yaitu menu "User", "Setting", dan "Warehouse". Pada menu "User" terdapat menu *login* atau *logout* dan *user list*. Menu "Setting" berisikan menu-menu yang berfungsi untuk mengatur faktor-faktor yang mempengaruhi penyusunan struktur *tree*, yaitu menu kategori dan block. Selain itu, terdapat juga menu *lock* atau *unlock* yang berfungsi untuk membangun dan menghancurkan tree. Pada menu "Warehouse" terdapat menu *goods* dan *map*. Bagian bawah *form* terdapat "Status Strip" yang berfungsi untuk menampilkan kondisi aplikasi yang sedang berjalan dan pesan.



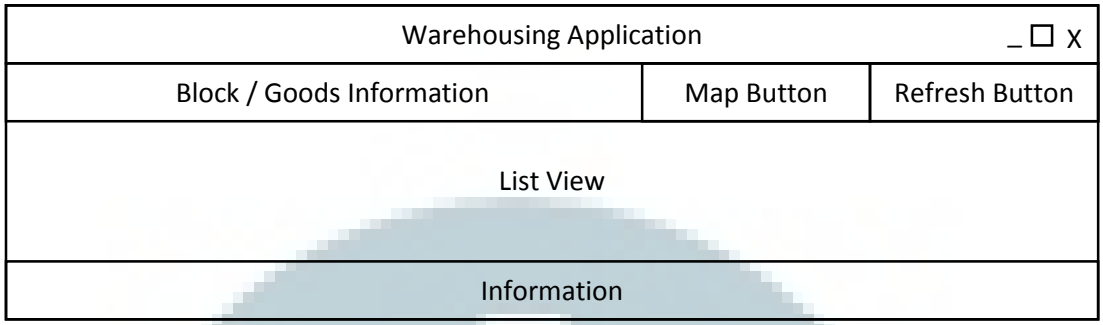
Gambar 3.18 *Template Form* untuk Menampilkan Data

Template form ini digunakan untuk menampilkan data pengguna, kategori, block, dan barang. Bagian bawah form terdapat "Control Button". Untuk data pengguna, kategori, dan barang, "Control Button" berisikan *add button* saja, sedangkan untuk data block, "Control Button" berisikan *add button* dan *refresh list button*. Pada *template form*, juga terdapat "Context Menu Strip". Untuk data pengguna dan kategori, "Context Menu Strip" berisikan menu *edit* dan *delete*, untuk data block, "Context Menu Strip" berisikan menu *edit*, *delete*, dan *show location*, dan untuk data barang, "Context Menu Strip" berisikan menu *edit, delete, add to block,* dan *show location*.



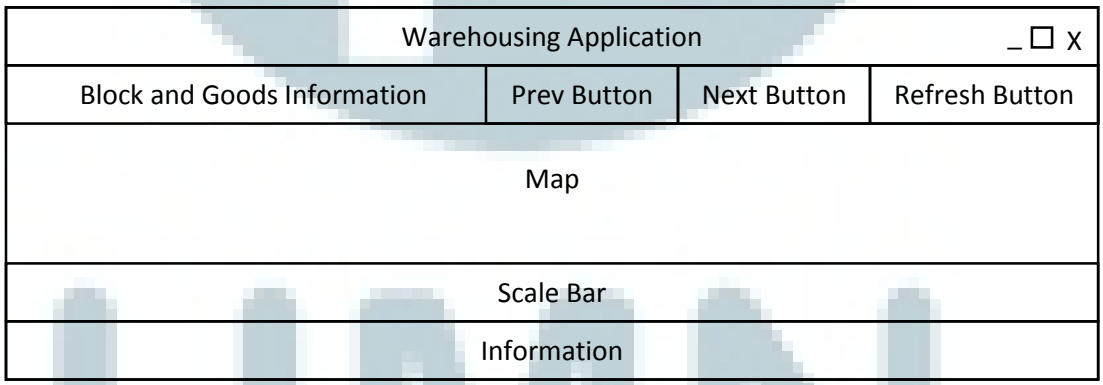
Gambar 3.19 *Template form* untuk Menambah atau Mengubah Data

Template form ini digunakan untuk login dan menambah atau mengubah data pengguna, kategori, block, dan barang. "Input" dapat berupa *label*, *textbox*, *combobox*, atau *numeric up down*. "Information" berisikan petunjuk pengisian. Bagian bawah *form* terdapat "Status Strip" yang berfungsi untuk menampilkan pesan.



Gambar 3.20 *Template form* untuk menampilkan lokasi

Template form ini digunakan untuk menampilkan lokasi barang tersimpan. Form ini dapat dipanggil dari *form* data block dan *form* data barang. "Map Button" ada jika *form* tersebut dipanggil dari *form* data block. Pada *template form*, juga terdapat "Context Menu Strip". "Context Menu Strip" berisi menu *map* untuk satu item barang tersimpan. Bagian bawah *form* terdapat "Information" yang berfungsi untuk menampilkan keterangan dari *column header*.



Gambar 3.21 *Template Form* untuk Menampilkan Lokasi

Template form ini digunakan untuk menggambarkan lokasi barang tersimpan pada suatu block. Bagian bawah *form* terdapat "Information" yang berfungsi untuk menampilkan keterangan dari suatu komponen pada *map*.