PEMBUATAN *APPLICATION PROGRAMMING INTERFACE* (API) UNTUK PENGELOLAAN DATA SPASIAL PADA *RELATIONAL DATABASE MANAGEMENT SYSTEM* (RDBMS) MySQL

SKRIPSI

JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi sebagian persyaratan

Memperoleh gelar Sarjana Teknik

Disusun oleh:

ANDIK NUR ACHMAD

NIM. 0610630013 - 63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

JURUSAN TEKNIK ELEKTRO

MALANG

2012

LEMBAR PERSETUJUAN

PEMBUATAN *APPLICATION PROGRAMMING INTERFACE* (API) UNTUK PENGELOLAAN DATA SPASIAL PADA *RELATIONAL DATABASE MANAGEMENT SYSTEM* (RDBMS) MySQL

SKRIPSI

JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi sebagian persyaratan
Memperoleh gelar Sarjana Teknik

Oleh:

ANDIK NUR ACHMAD
NIM. 0610630013 - 63

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Muhammad Aswin, Ir., MT.	Adharul Muttaqin, ST. MT.
NIP. 19640626 199002 1 001	NIP. 19760121 200501 1 001

PENGANTAR

Alhamdulillah, segenap puji dan syukur penulis panjatkan kepada Allah SWT yang telah memberikan rahmat dan hidayahNya sehingga penulis dapat menyelesaikan skripsi dengan judul "**Pembuatan *Application Programming Interface* (API) Untuk Pengelolaan Data Spasial Pada *Relational Database Management System* (RDBMS) MySQL**", yang diajukan untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik.

Tidak banyak yang bisa penulis sampaikan kecuali ungkapan terima kasih kepada berbagai pihak yang telah dengan tulus ikhlas memberikan bimbingan, arahan, dan dukungan hingga penulisan tugas akhir ini dapat terselesaikan. Pada kesempatan kali ini, penulis mengucapkan banyak terima kasih kepada:

1. Kedua Orang Tua Penulis (Marsam dan Sudarwati) dan seluruh keluarga yang senantiasa tiada henti-hentinya memberikan doa dan restu demi terselesaikannya skripsi ini.
2. Bapak Dr. Ir. Sholeh Hadi Pramono, MS selaku Ketua Jurusan Teknik Elektro.
3. Bapak M. Azis Muslim, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro.
4. Bapak Waru Djuriatno, ST., MT selaku Ketua Kelompok Dosen Keahlian Teknik Rekayasa Komputer.
5. Bapak Ir. Muh. Aswin, MT selaku dosen pembimbing skripsi, yang telah banyak memberikan dukungan dan koreksi kepada penulis.
6. Bapak Adharul Muttaqin, ST., MT selaku dosen pembimbing skripsi,yang telah banyak memberikan dukungan dan koreksi kepada penulis.
7. Seluruh dosen dan karyawan Jurusan Teknik Elektro.
8. Nourma Devita Pratiwi, Anggakara Yudha P., Eka Prakarsa M., Norman Natanael, kawan dan kerabat yang telah membantu memberikan saran, bantuan, dan dukungan hingga terselesaikannya Tugas Akhir ini.

9. Teman-teman angkatan 2006 Teknik Elektro yang telah penulis anggap sebagai keluarga kedua, atas semua bantuan dan motivasinya selama ini.
10. Semua Asisten Laboratotium Informatika dan Komputer atas bantuan dan dukungan dalam penyelesaian Tugas Akhir ini.
11. Semua pihak yang tidak dapat saya sebutkan satu persatu atas dukungannya dan bantuannya.

Penulis menyadari sepenuhnya bahwa Tugas Akhir ini masih banyak kekurangan. Segala kritik dan saran yang membangun akan penulis terima demi kesempurnaan skripsi ini. Harapan penulis semoga skripsi ini bermanfaat bagi pembaca dan khususnya mahasiswa jurusan teknik elektro dimasa yang akan datang.

Malang, Februari 2012

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Andik Nur Achmad, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Februari 2012, **Pembuatan *Application Programming Interface* (API) Untuk Pengelolaan Data Spasial Pada *Relational Database Management System* (RDBMS) MySQL**, Dosen Ir. Muhammad Aswin, MT. dan Adharul Muttaqin, ST. MT.

Salah satu dari data khusus yang dikelola oleh basis data saat ini adalah data spasial, yaitu data yang berhubungan dengan ruang. Oleh karena itu, data spasial bisa merupakan data 0 dimensi, 1 dimensi, 2 dimensi, atau n dimensi. Titik adalah data spasial 0 dimensi,, garis adalah data spasial 1 dimensi, sedangkan area atau daerah adalah contoh data 2 dimensi. Karena sifatnya adalah n-dimensi, maka data spasial tidak dapat dikelola menggunakan sistem basis data spasial pada umumnya, yang notabene hanya menyimpan data alfanumerik saja. Diperlukan sebuah sistem basis data yang menyediakan fasilitas, yaitu untuk mengenali dan mengelola data spasial, yaitu sistem basis data spasial.

Perancangan dan implementasi *Application Programming Interface (*API*)* untuk pengelolaan data spasial ini dikerjakan dengan beberapa tahap. Tahapan tersebut meliputi proses analisa kebutuhan yang bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem yang akan dibangun. Perancangan dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisa kebutuhan. Proses selanjutnya adalah implementasi dari hasil perancangan API.Pada tahap ini akan dilakukan proses pembangunan API. Hasil dari proses pada bagian implementasi ini adalah sebuah API yang dapat dipakai untuk membangun basis data spasial dengan menggunakan sistem basis data relasional MySQL.

Hasil pengujian API untuk pengelolaan data spasial pada RDBMS MySQL ini menunjukkan bahwa API telah mampu menangani semua kasus yang dijadikan parameter pengujian sebagai parameter kesuksesan, yaitu pembuatan skema basis data spasial (CREATE DATABASE), pembuatan tabel spasial (CREATE TABLE), pengisian data ke dalam tabel spasial (INSERT), pembaharuan data pada tabel spasial (UPDATE), penghapusan data pada tabel spasial (DELETE), dan eksekusi query dengan menggunakan operator spasial.

Kata kunci: API, MySQL, basis data relasional, basis data spasial, operator spasial

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam dunia teknologi informasi, dapat dikatakan bahwa hal yang konstan hanyalah perubahan dan perkembangan. Hal ini berlaku pula pada teknologi basis data. Dahulu basis data hanya digunakan untuk menyimpan data yang sederhana seperti biodata pribadi, data pembayaran, data penggajian, dan lain-lain. Namun saat ini, basis data digunakan untuk menyimpan hal yang lebih kompleks, dan lebih menarik, seperti video, musik, gambar satelit, desain chip, atau peta. Fungsi utama basis data tetap sama, yaitu : penyimpanan dan pencarian data yang efisien. Namun demikian, dengan perkembangan tersebut semakin banyak tipe data yang harus dikelola oleh basis data.

Salah satu dari data khusus yang dikelola oleh basis data saat ini adalah data spasial, yaitu data yang berhubungan dengan ruang. Oleh karena itu, data spasial bisa merupakan data 0 dimensi, 1 dimensi, 2 dimensi, atau n dimensi. Titik adalah data spasial 0 dimensi,, garis adalah data spasial 1 dimensi, sedangkan area atau daerah adalah contoh data 2 dimensi. Karena sifatnya adalah n-dimensi, maka data spasial tidak dapat dikelola menggunakan sistem basis data spasial pada umumnya, yang notabene hanya menyimpan data alfanumerik saja. Diperlukan sebuah sistem basis data yang menyediakan fasilitas, yaitu untuk mengenali dan mengelola data spasial, yaitu sistem basis data spasial.

Sistem basis data spasial adalah sebuah sistem basis data yang menyediakan tipe data dan cara untuk mengelola data spasial. Tipe tersebut antara lain adalah POINT, LINE, dan REGION. POINT dapat mewakili pohon, lampu jalan, hydrant, dll. LINE dapat digunakan untuk merepresentasikan jalan, sungai, batas propinsi, dll. Sedangkan REGION dapat digunakan untuk menggambarkan rumah, danau, negara, dll. Relasi adalah bagaimana hubungan antara dua data spasial, misalnya apakah saling memotong (INTERSECT), berada di dalam (INSIDE), atau berdekatan (ADJACENT). Properti dari data spasial misalnya adalah panjang dari LINE (LENGTH), luas dari REGION (AREA), dll.

Sedangkan operasi mencari hasil dari relasi dua atau lebih data spasial, misalnya potongan (INTERSECTION), gabungan (UNION), dll.

Basis data relasional adalah basis data yang paling banyak digunakan saat ini.Semua DMBS terkemuka mendukung basis data relasional.Basis data relasional cukup popular karena kemudahan penggunaannya dan banyaknya referensi yang dapat digunakan untuk mempelajarinya. Oleh karena itu, dipilihlah basis data relasional sebagai basis data dasar yang akan dipakai dalam pengerjaan tugas akhir ini. Kemampuan penanganan tipe data spasial akan ditambahkan melalui layer aplikasi tersendiri yang berada di atas DBMS. Layer aplikasi ini berdiri secara independen dan tidak menjadi bagian dari DBMS.

1.2 Rumusan Masalah

Dari latar belakang yang telah diutarakan sebelumnya, rumusan masalah yang akan diselesaikan pada tugas akhir ini adalah sebagai berikut:

1. Mendefinisikan data aspek ruang (spasial) agar dapat disimpan pada basis data relasional.
2. Mendefinisikan operasi dan operator yang digunakan untuk melakukan manipulasi data spasial.
3. Mengimplementasikan operasi dan operator data spasial agar dapat dijalankan pada basis data relasional.

1.3 Batasan Masalah

Dalam upaya mengatasi permasalahan yang terdapat pada bagian rumusan masalah, perlu batasan yang digunakan adalah sebagai berikut:

1. Pemodelan data spasial dilakukan untuk basis data relasional.
2. Sistem manajemen basis data yang digunakan adalah MySQL.
3. Bahasa pemrograman yang digunakan adalah C.
4. Sistem operasi yang digunakan adalah Microsoft Windows XP SP3.

1.4 Tujuan

Tujuan utama dari tugas akhir ini adalah untuk merancang dan mengembangkan sebuah Application Programming Interface (API) yang digunakan untuk mengelola dan memanipulasi data spasial pada sistem

manajemen basis data relasional MySQL.

1.5 Manfaat

Manfaat yang dapat diperoleh oleh penyusun melalui pengerjaan tugas akhir ini adalah:

1. Penyusun dapat mengaplikasikan ilmu yang telah didapatkan selama menempuh perkuliahan di Jurusan Teknik Elektro Universitas Brawijaya.
2. Penyusun memahami data spasial, operasi, operator, dan query yang berhubungan dengan data spasial.
3. Penyusun mampu mengimplementasikan pemodelan data spasial, operasi, operator, dan query data spasial sehingga dapat digunakan pada basis data relasional.

1.6 Sistematika Penulisan

Sistematika penulisan dan gambaran untuk setiap bab pada laporan skripsi ini adalah sebagai berikut :

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat dan sistematika penulisan.

BAB II Dasar Teori

Membahas teori dasar dan teori penunjang yang berkaitan dengan basis data spasial, API, *Dinamic-Link Library*, dan MySQL.

BAB III Metodologi

Membahas metode yang digunakan dalam penulisan yang terdiri dari studiliteratur, perancangan perangkat lunak, implementasi perangkat lunak, pengujian perangkat lunak, serta pengambilan kesimpulan dan saran.

BAB IV Perancangan dan Implementasi

Membahas analisa kebutuhan untuk API dan perancangan API.

BAB V Pengujian

Membahas tentang implementasi hasil perancangan yang telah dibuat sehingga dihasilkan API yang dapat diuji coba.

BAB VI Kesimpulan dan Saran

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian API,serta saran–saran untuk pengembangan lebih lanjut.

BAB II
DASAR TEORI

2.1 Sistem Basis Data Spasial

2.1.1 Definisi

Sistem basis data spasial saat ini adalah sistem basis data pada umumnya dengan penambahan kemampuan dan fungsi untuk menangani data spasial. Kemampuan dan fungsi tersebut meliputi [YEU-07]:

- Tipe data spasial

Tipe data spasial dapat disimpan sebagai tipe data special sesuai dengan definisi dari "*simple feature*" oleh Open Geospatial Consortium (OGC,1999) atau sebagai *binary large object* (BLOB) pada basis data. *Simple feature*yang didefinisikan OGC ditangani oleh fungsionalitas dari database sendiri. Sedangkan BLOB adalah tipe data generic dimana semua data binary dapat disimpan. Namun konsekuensinya, data BLOB harus diindex dan dimanipulasi menggunakan komponen perangkat lunak tambahan.

- Index spasial

Index spasial adalah mekanisme untuk memfasilitasi akses ke basis data spasial berdasarkan koordinat yang tersimpan dalam ruang 2 dimensi. Ada banyak pilihan index yang berbeda-beda, seperti R-Tree, quadtree, dan B-Tree. Masing-masing memiliki kelebihan dan keurangan bergantung pada format data dan kebutuhan spesifik dari aplikasi.

- Operator spasial

Operator spasial adalah sekumpulan fungsi pemroses data dan proses yang dapat dimanfaatkan melalui penggunaan Structured Query Language (SQL) untuk melakukan query dan mengambil isi basis data yang dipilih, menggabungkan tabel berdasarkan criteria spasial atau non-spasial, dan menghasilkan hasil proses dalalm format tertentu.

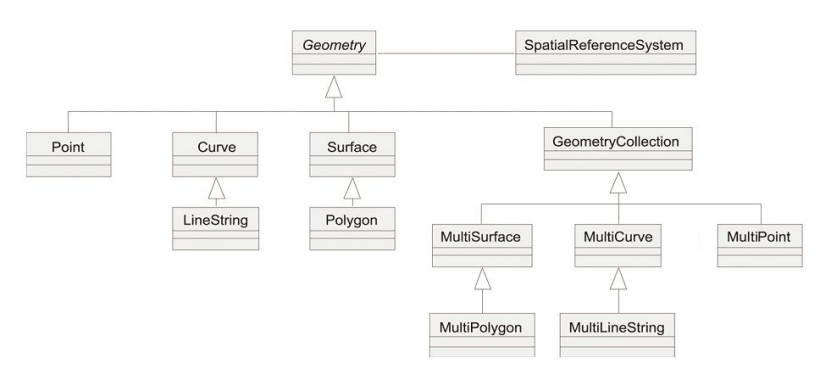
- *Routine* aplikasi spasial

Routine ini meliputi berbagai macam komponen perangkat lunak untuk aplikasi basis data yang spesifik seperti *spatial data loading, versioning and long*

transaction control, performance tuning, database backup and replication.

2.1.2 Tipe Data Spasial

Open Geospatial Consortium (OGC) telah mendefinisikan tipe data spasial yang menjadi acuan implementasi basis data spasial di seluruh dunia. Tipe data yang disebut *geometry object model* ini dapat dilihat pada gambar 1. Tiap tipe data dalam *geometry object model* didefinisikan dalam sebuah class. Masing-masing class tersebut memiliki metode, properti dan definisi relasi spasial yang berbeda-beda.



Gambar 2.1 Geometry Object Model oleh OGC

Di dalam *geometry object model* ini terdapat sebuah *class* yang menjadi *super class* bagi semua tipe data spasial yaitu Geometry. Geometry memiliki 4 *sub class* langsung yaitu Point, Curve, Surface, dan Geometry Collection. Geometry Collection adalah koleksi dari beberapa macam geometry yang berbeda.MultiPoint, MultiCurve, dan MultiSurface adalah sub tipe yang spesifik untuk menangani koleksi homogen dari tipe Point, Curve, dan Surface.Point dan MultiPoint adalah tipe data yang digunakan untuk menangani data 0 dimensi.Curve, MultiCurve dan *sub class*nya digunakan untuk menangani data 1 dimensi.Surface, MultiSurface, dan *sub class*nyadigunakan untuk menangani data 2 dimensi.

2.1.3 Pemrosesan Data Spasial

Pemrosesan data spasial dilakukan dengn membuat query baru yang dikembangkan dari query yang telah ada. Query ini lebih kompleks dari query yang telah ada, karena melibatkan pemrosesan data dua dimensi, bahkan tiga

dimensi. Query baru yang digunakan untuk pemrosesan data spasial ini disebut query spasial. Beberapa unsur pembangun query spasial adalah operator spasial dan relasi spasial.

2.1.3.1 Klasifikasi Operator Spasial

Operator spasial adalah salah satu unsur pembangun query spasial. Operator spasial dapat diklasifikasikan dengan beberapa cara. Cara pertama adalah dengan membagi berdasarkan jumlah *operand* yang diperlukan, yaitu *unary* dan *binary*. Operator *unary* adalah operator yang membutuhkan satu *operand* saja. Biasanya operator ini digunakan untuk mendapatkan property dari sebuah objek spasial, seperti: *area*, *length*, dan *volume*. Operator binary adalah operator yang membutuhkan dua buah operand dan dipakai untuk mendapatkan relasi antar objek spasial, seperti : *distance*, *intersect*, *union*, dan *intersection*.

Cara lain untuk mengklasifikasikan operator spasial adalah dengan mengklasifikasikan operator sebagai operator topological, projective, atau metric. Operator topological adalah operator yang bergubungan dengan relasi topological seperti : touch, intersect, contain, cross, dan disjoint. Sedangkan operator projective adalah operator yang berkaitan dengan convexity sebuah objek spasial. Secara harfiah, convexity berarti kecembungan, Dalam konteks polygon, terdapat 6 buah tingkat convexity, yaitu:

a. Convex : yaitu setiap garis yang melewati sisi polygon pasti akan melewati polygon sebanyak tepat dua kali.
b. Non-convex : yaitu setiap garis yang lelewati sisi polygon bisa melewati polygon lebih dari dua kali.
c. Simple : yaitu setiap garis pembangun polygon tidak saling berpotongan.
d. Concave : yaitu polygon yang memenuhi criteria Non-convex dan simple.
e. Star-shaped : yaitu polygon yang memenuhi criteria simple, dan convex atau concave.
f. Self-intersecting : yaitu polygon yang sisi-sisinya saling berpotongan satu sama lain.

Contoh dari operator ini adalah convex hull, yaitu mencari bentuk polygon convex terkecil yang dapat dibuat dari sekumpulan titik. Operator metric adalah operator yang berhubungan dengan pengukuran objek spasial, contohnya adalah : distance dan direction.

OGC mengklasifikasikan operator spasial menjadi 3 jenis operator, yaitu basic operator, topological operator, dan spatial analysis operator. Basic operator merupakan operator yang berkaitan dengan property umum dari objek spasial. Topological operator sama seperti yang telah dijelaskan sebelumnya. Sedangkan spatial analysis operator merupakan operator yang dipakai untuk melakukan analisa terhadap sebuah atau sekumpulan objek spasial. Klasifikasi operator yang dibuat oleh OGC dapat dilihat pada tabel 2.1.

Tabel 2.1 Klasifikasi Operator spasial oleh OGC

Klasifikasi	Operator	Fungsi Operator
Basic Operator	Area	Mendapatkan luas dari sebuah objek geometri
	Length	Mendapatkan panjang dari sebuah objek geometri
	Boundary	Mendapatkan sisi-sisi dari sebuah objek geometri
	Center	Mendapatkan titik tengah dari sebuah objek geometri
Topological Operator	Equal	Melakukan pengecekan apakah dua objek geometri adalah sama dan sebangun
	Intersect	Melakukan pengecekan apakah dua objek geometri saling berpotongan.
	Disjoint	Melakukan pengecekan apakah dua objek geometri saling tidak saling berhubungan.
	Touch	Melakukan pengecekan apakah dua objek geometri saling bersentuhan.
	Contain	Melakukan pengecekan apakah salah satu objek geometri

		mengandung objek geometri yang lain.
Spatial Analysis Operator	Distance	Mendapatkan jarak antara dua objek geometri
	Intersection	Mendapatkan daerah perpotongan antara dua objek geometri
	Union	Mendapatkan daerah gabungan dari dua objek geometri.
	Difference	Mendapatkan daerah salah satu objek geometri yang bukan merupakan objek geometri lainnya.
	SymDifference	Mendapatkan daerah yang merupakan daerah yang bukan menjadi perpotongan dari dua objek geometri.

2.1.3.2 Relasi Spasial

Relasi antar objek spasial adalah relasi topologi. Relasi topologi menganalisa posisi relatif sebuah objek spasial dalam ruang tertentu. Dalam basis data spasial, relasi topologi dianalisa menggunakan operator topologi.Dasar dari relasi topologi adalah mencari semua kemungkinan relasi yang ada antara dua objek spasial. Eggenhofer dan Herring mencatat jika ada dua buah polygon di dalam sebuah ruang, maka akan terdapat 16 kombinasi relasi yang mungkin terjadi. Kombinasi relasi ini didapatkan dari keberadaan daerah *intersection* antara *boundary* (δA) dan *interior* (A°) dari objek spasial, dalam hal ini adalah region. Boundary adalah garis luar dari region, sedangkan interiror adalah garas dalam dari region. Kombinasi tersebut dapat dilihat pada table 2.2.

Tabel 2.2 Spesifikasi Relasi Topologi Antara Interior dan Boundary dari Objek Spasial

	$\delta A \cap \delta B$	$A^\circ \cap B^\circ$	$\delta A \cap B^\circ$	$A^\circ \cap \delta B$
r_0	Ø	Ø	Ø	Ø

r_1	¬Ø	Ø	Ø	Ø
r_2	Ø	¬Ø	Ø	Ø
r_3	¬Ø	¬Ø	Ø	Ø
r_4	Ø	Ø	¬Ø	Ø
r_5	¬Ø	Ø	¬Ø	Ø
r_6	Ø	¬Ø	¬Ø	Ø
r_7	¬Ø	¬Ø	¬Ø	Ø
r_8	Ø	Ø	Ø	¬Ø
r_9	¬Ø	Ø	Ø	¬Ø
r_{10}	Ø	¬Ø	Ø	¬Ø
r_{11}	¬Ø	¬Ø	Ø	¬Ø
r_{12}	Ø	Ø	¬Ø	¬Ø
r_{13}	¬Ø	Ø	¬Ø	¬Ø
r_{14}	Ø	¬Ø	¬Ø	¬Ø
r_{15}	Ø	¬Ø	¬Ø	¬Ø

Dari 16 kombinasi ini, 8 diantaranya tidak valid, dan 2 kombinasi merupakan simetri. Kombinasi r_2, r_4, r_5, r_8, r_9, r_{12}tidak valid karena jika terdapat perpotongan antara *interiorA – boundaryB* atau *boundaryA – interiorB*, maka harus ada perpotongan antara *interior A – interior B* juga [EGE-90]. Ketidakvalidan relasi r_8 dan r_{15} karena jika kedua region tidak saling berpotongan di *boundary*-nya maka terdapat salah satu kemungkinan yang harus terjadi, yaitu *interior*-nya tidak berpotongan, atau *interior*-nya berpotongan dan salau satu kemungkinan antara *interior A – boundary B* atau *boundary A – interior B* berpotongan. Hal ini tidak dipenuhi oleh relasi r_8 dan r_{15}. Sehingga terdapat 8 relasi yang valid, seperti yang terdapat pada tabel 3. Akan tetapi, karena relasi r6 dan r10 juga r7 dan r11 adalah relasi yang sama, maka hanya terdapat 6 relasi yang mungkin terjadi antara dua region, yaitu disjoing, *contain*, *meets*, *equals*, *cover*, dan *intersect*.

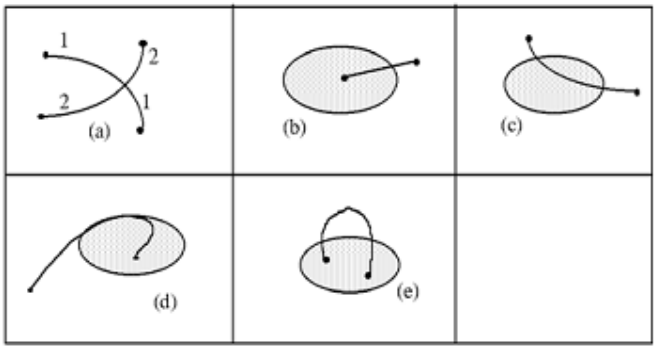
Tabel 2.3 Relasi Spasial yang Mungkin dari 2 Region

	$\delta A \cap \delta B$	$A^\circ \cap B^\circ$	$\delta A \cap B^\circ$	$A^\circ \cap \delta B$	Relasi
r_0	Ø	Ø	Ø	Ø	*A disjoint B*
r_1	¬Ø	Ø	Ø	Ø	*A meets B*
r_3	¬Ø	¬Ø	Ø	Ø	*A equals B*
r_6	Ø	¬Ø	¬Ø	Ø	*A inside B*
r_7	¬Ø	¬Ø	¬Ø	Ø	*A covers B*
r_{10}	Ø	¬Ø	Ø	¬Ø	*A inside B*
r_{11}	¬Ø	¬Ø	Ø	¬Ø	*A covers B*
r_{15}	Ø	¬Ø	¬Ø	¬Ø	*A overlaps B*

Clementini et.al melakukan pengembangan dari hasil kerja Eggenhofer dan Herring. Clementini et.al mencari kombinasi tidak hanya untuk region, tetapi juga untuk point dan line. Kemungkinan yang didapatkan berjumlah 256 kombinasi. Namun dari 256 kombinasi relasi, hanya 52 kombinasi yang valid [CLE-93]. Karena 52 kombinasi relasi dianggap terlalu banyak, maka diajukan hanya 5 relasi yang paling dibutuhkan pada aplikasi spasial, yaitu *cross*, *contain*, *intersect*, *touch*, dan *disjoint*. Definisi dari setiap relasi tersebut adalah sebagai berikut [CLE-93]:

a. Cross

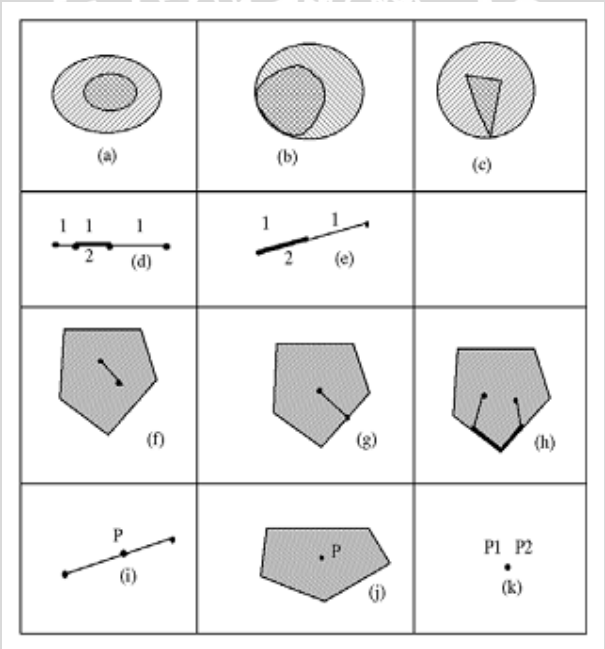
Dua buah *line* dinyatakan saling bersilangan jika kedua *line* tersebut memiliki sebuah titik internal yang sama. Sebuah *garis* berpotongan dengan *region* jika sebagian garis tersebut terletak di dalam *region*, dan sebagian lainnya terletak di luar *region*. Operator ini hanya dapat diberlakukan pada pasangan objek spasial *line – line* dan *line – region*. Gambar 3 menunjukkan kemungkinan perpotongan dua buah objek spasial.



Gambar 2.2 Penggambaran relasi cross antara 2 *line* (a), *line* dan *region* (b-e)

b. Contain

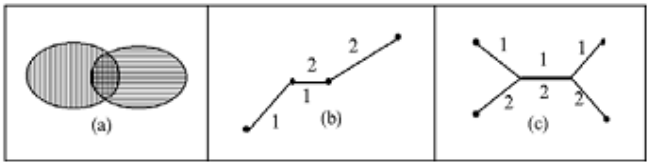
Operator ini dapat digunakan untuk semua kemungkinan pasangan objek spasial. Sebuah objek A dikatakan mengandung (contain) objek B bila objek B berada di dalam objek A. Semua kemungkinan pada relasi ini ditunjukkan pada gambar 4.



Gambar 2.3 Penggambaran relasi contain antara 2 region (a-c), 2 line (d-e), line dan region (f-h), point dan line (i), point dan region (j), 2 point (k)

c. Intersect

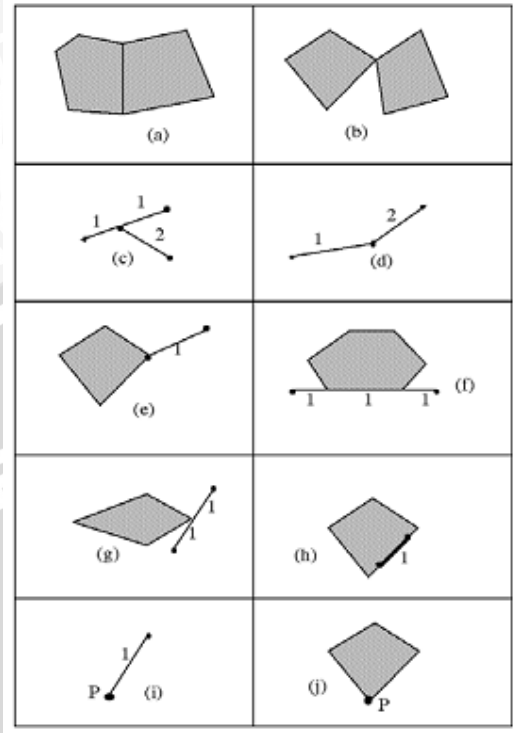
Dua buah objek spasial dikatakan beririsan jika hasil irisan antara 2 buah objek spasial tersebut merupakan objek spasial yang memiliki dimensi sama dengan objek asalnya. Operator ini hanya dapat digunakan untuk *line* dan *region* saja. Gambar 2.4 menunjukkan kemungkinan dari relasi ini.



Gambar 2.4 Penggambaran relasi *intersect* antara 2 *region* (a), dan 2 *line* (b-c)

d. Touch

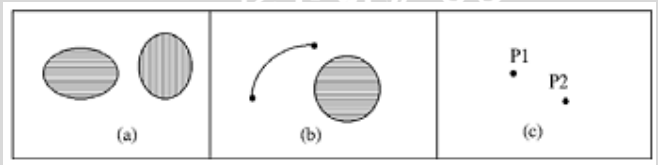
Dua buah objek spasial dikatakan saling menyentuh jika *common area* antara dua buah objek tersebut terletak pada *boundary*-nya. Point tidak memiliki boundary. Line memiliki boundary yaitu point awal dan point akhir jika berpasangan dengan point. Semua point dalam line adalah boundary bagi line jika berpasangan dengan region. Sedangkan pada region, boundary-nya merupakan line pembatas region tersebut. Operator ini dapat digunakan untuk semua kemungkinan pasangan objek spasial. Gambar 2.5 menunjukan kemungkinan relasi operator ini.



Gambar 2.5 Penggambaran relasi touch antara 2 region (a-b), 2 line (c-d), line dengan region (e-h), point dengan line (i), dan point dengan region (j).

e. Disjoint

Dua buah objek spasial dikatakan saling asing apabila kedua objek spasial tersebut tidak memiliki daerah pertemuan sama sekali. Operator ini dapat digunakan untuk semua pasangan objek spasial. Gambar 2.6 menunjukkan sebagian kemungkinan relasi ini.



Gambar 2.6 Penggambaran relasi disjoint antara 2 region (a), line dengan region (b), dan 2 point (c)

2.1.3.3 Query Spasial

Query spasial adalah query yang digunakan untuk mencari jawaban dari pertanyaan seputar data spasial. Beberapa jenis query spasial menurut Antonio Corral [MAN-05], adalah :

a. Point Location Query

   Query ini digunakan untuk mencari objek spasial berdasarkan point tertentu. Misalnya, “Di propinsi apa terletak kota Malang?”

b. Range Query

   Query ini digunakan untuk mencari objek spasial dalam lingkup region. Misalnya, “Hydrant nomor berapa saja yang terdapat pada kelurahan Tunjung Sekar?”.

c. Join Query

   Query ini melibatkan dua atau lebih operasi topologi. Misalnya, “Sungai apa yang melintasi jawa timur tetapi tidak bermuara di laut jawa?”.

d. Nearest Neighbor Query

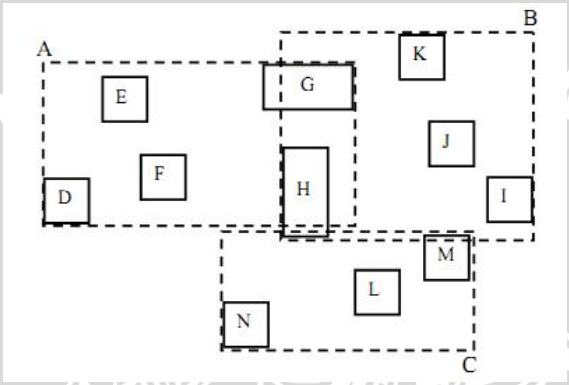
   Query ini mencari objek spasial yang paling dekat dengan objek spasial yang ditentukan. Misalnya, “Mana hydrant yang paling dekat dengan pasar blimbing?”.

2.1.4 Index Spasial

Pemrosesan query yang efisien membutuhkan beberapa strategi. Sistem basis data harus dilengkapi dengan struktur index, metode pemrosesan query, dan optimisasi query yang sesuai. Index digunakan untuk mempercepat pencarian data, dengan membuat struktur tersendiri dari atribut atau kolom tertentu menggunakan salah satu teknik struktur data. Struktur data yang biasa dipakai untuk melakukan index pada basis data adalah B-Tree. Basis data spasial, yang datanya merupakan representasi lebih dari satu dimensi, membutuhkan mekanisme index tertentu yang mampu menangani data 2 dimensi. Salah satu dari metode index tersebut adalah R-Tree.

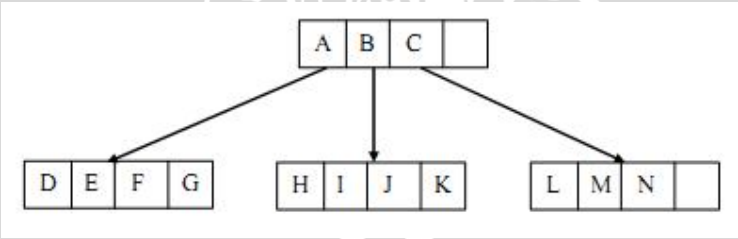
R-Tree struktur data hierarkis yang dibuat berdasarkan B^+-Tree. Struktur data ini digunakan untuk meng-organisasi-kan data objek geometri x-dimensi dengan merepresentasikan objek tersebut ke dalam *minimum bounding x-*

dimensional rectangles (MBR). Tiap node dalam R-Tree berkorespondensi terhadap MBR yang mengikat objek anaknya (objek yang ada di dalamnya). Leave pada tree menyimpan pointer ke objek dalam basis data, alih-alih menyimpan pointer ke node anaknya. Node ini diimplementasikan sebagai *disk page*. Contoh implementasinya adalah sebagai berikut. Misalnya terdapat beberapa MBR dalam sebuah ruang, seperti yang terlihat pada gambar 8. MBR tersebut adalah D, E, F, G, H, I, J, K, L, M, dan N. Masing-masing MBR berisi objek spasial yang tidak digambarkan.



Gambar 2.7 Contoh MBR dan MBR yang melingkupinya

R-Tree yang berkorespondensi dengan MBR-MBR tersebut dijelaskan pada gambar 9.



Gambar 2.8 R-Tree dari representasi MBR pada gambar 2.7

2.2 MySQL

MySQL adalah sebuah Relational Database Management System (RDBMS) yang free dan open source dengan lisensi GNU General Public License. Fitur yang dimiliki oleh MySQL adalah sebagai berikut [DUB-09]:

- Mudah digunakan. MySQL adalah DBMS yang berperforma tinggi namun relative sederhana dan mudah diadministrasi dibandingkan dengan sistem yang lebih besar.
- Dukungan Query language. MySQL mendukung SQL (Structured Query Language), bahasa standard untuk sistem basis data modern.
- *Capability*. Server MySQL adalah multi-threaded, sehingga banyak klien dapat terhubung pada waktu yang sama. Setiap klien dapat menggunakan database secara bersamaan. MySQL dapat diakses secara interaktif menggunakan beberapa antarmuka yang memungkinkan input query danmelihat hasil query melalui*command-line client*, browser Web, atau GUI*client*. Selain itu,antarmuka pemrograman juga tersedia untuk banyak bahasa, seperti C, Perl, Java, PHP, Python, dan Ruby.
- Konektivitas dan keamanan. MySQL mendukung jaringan secara penuh. Basis data dapat dimanapun melalui internet. Namun demikian, MySQL memiliki mekanisme control akses, sehingga seseorang tidak dapat melihat data yang bukan haknya. Untuk memberikan keamanan tambahan, MySQL mendukung koneksi terenkripsi menggunakan protocol Secure Sockets Layer (SSL).
- Portabilitas. MySQL dapat berjalan pada berbagai varian linux dan unix, juga windows dan netware.
- Ukuran yang kecil. MySQL memiliki ukuran yang relative kecil, jika dibandingkan dengan DBMS lain.

Biaya. MySQL adalah adalah project yang tersedia di bawah beberapa lisensi Pertama, MySQL tersedia di bawah lisensi GNU GPL, yang berarti MySQL dapat digunakan tanpa biaya. Kedua, jika sebuah organisasi lebih memilih untuk tidak terikat pada lisensi GPL, lisensi komersial juga tersedia.

2.3 MySQL User Defined Function

MySQL memberikan fasilitas bagi pengguna untuk membuat fungsi dalam bentuk *dynamic-link library(dll)* yang dibuat dalam bahasa C. File dll ini nantinya akan dihubungkan ke MySQL sebagai tambahan modul. Cara ini memungkinkan

pengguna membuat fungsi kompleks, tanpa harus mengubah dan mengkompilasi ulang *source code* MySQL.

2.4 *Application Programming Interface* (API)

Application Programming Interface (API) adalah sekumpulan routine, struktur data, variable, atau objek dan kelas, yang digunakan sebagai antarmuka bagi komponen – komponen perangkat lunak untuk berinteraksi satu sama lain, atau untuk melakukan fungsi – fungsi tertentu. [ANO-12]. API dapat berupa Dynamic Link Library (DLL), JAR, atau format lain.

2.5 *Dynamic-Link Library (DLL)*

Dynamic-link Library (DLL) adalah file *executable* yang berlaku sebagai *shared library / function.*Fungsi – fungsi yang ada di dalamnya, dapat digunakan / diakses oleh aplikasi yang menggunakannyaa pada saat runtime. Oleh karena itu, suatu aplikasi yang akan menggunakan fungsi yang terdapat pada DLL, DLL tersebut tidak perlu di link-kan pada waktu kompilasi.

BAB III
METODOLOGI

Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam perancangan, implementasi dan pengujian dari aplikasi perangkat lunak yang akan dibuat. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan kemungkinan arah pengembangan aplikasi selanjutnya.

3.1 Studi Literatur

Studi literatur dilakukan untuk menggali informasi dan pengetahuan sebagai landasan untuk menentukan bagaimana perancangan dan implementasi API pada tugas akhir ini. Hal-hal yang perlu digali antara lain adalah : bagaimana cara menyimpan data spasial pada suatu RDBMS, menentukan operasi dan operator spasial yang digunakan pada basis data spasial, dan menentukan cara paling tepat untuk mengimplementasikan operator spasial tersebut untuk RDBMS MySQL.

3.2 Perancangan dan Implementasi Sistem

Perancangan sistem dilakukan dalam 2 tahap. Analisis kebutuhan dilakukan pada tahap pertama, kemudian dilanjutkan dengan perancangan sistem berdasarkan hasil analisis kebutuhan pada tahap kedua. setelah semua kebutuhan sistem didapatkan melalui tahap analisa kebutuhan. Secara garis besar, perancangan API untuk tugas akhir ini meliputi:

a. Pembuatan skema model spasial agar sistem basis data relasional MySQL dapat menyimpan data spasial.
b. Pembuatan rancangan fungsi-fungsi yang berkaitan dengan penyimpanan dan pengelolaan data spasial.
c. Pembuatan rancangan fungsi-fungsi untuk mengimplementasikan konsep operator spasial.

Implementasi sistem dilakukan dengan mengacu pada perancangan aplikasi. Secara garis besar, pada tahap ini akan diimplementasikan semua fungsi yang telah dirancang pada tahap perancangan, menggunakan PL/SQL dan bahasa

C. Fungsi – fungsi yang dibuat menggunakan bahasa C akan dibentuk menjadi sebuat berkas *Dynamic-Link Library (DLL)*, yang akan dikaitkan pada MySQL pada saat runtime.

3.3 Pengujian dan Analisis

Pada tahap ini akan dilakukan pengujian untuk menguji fungsionalitas API. Contoh kasus pengujian adalah sebagai berikut:

1. Pembuatan database spasial (CREATE DATABASE)
2. Pengisian data ke dalam tabel spasial (INSERT)
3. Pembaharuan data pada tabel spasial (UPDATE)
4. Penghapusan data pada tabel spasial (DELETE)
5. Query dengan menggunakan operator spasial melibatkan 1 tabel.
6. Query dengan menggunakan operator spasial melibatkan 2 tabel.

Dalam pengujian ini, query spasial dimasukkan melalui program antarmuka sederhana yang dibuat menggunakan bahasa C dan memanfaatkan API ini.

3.4 Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian API telah selesai dilakukan.Kesimpulan diambil dari hasil pengujian dan analisis terhadap API yang dibangun. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan API selanjut nya.

BAB IV
PERANCANGAN DAN IMPLEMENTASI

Bab ini membahas mengenai perancangan dan implementasi sistem. Perancangan dilakukan dalam dua tahap, yaitu analisis kebutuhan dan perancangan sistem. Tahap analisis kebutuhan dilakukan dengan membuat daftar kebutuhan, sedangkan tahap perancangan sistem dilakukan dengan menggambarkan skema dan alur logika pemrograman menggunakan diagram alir (Diagram alir).

Implementasi sistem dilakukan setelah tahap perancangan sistem. Pada tahap ini, semua alur logika akan diimplementasikan dalam bentuk kode program. Agar lebih dapat dipahami, cuplikan kode program ditampilkan setelah

4.1 Perancangan

4.1.1 Analisis Kebutuhan

Analisis kebutuhan dilakukan dengan studi literatur tentang spesifikasi basis data spasial. Hasil analisa kebutuhan, berupa daftar kebutuhan, dituliskan pada tabel 4.1.

Tabel 4.1 Daftar Kebutuhan

No	Kebutuhan
1	API harus menyediakan skenario penyimpanan data spasial
2	API harus menyediakan fasilitas untuk membuat tabel feature
3	API harus menyediakan fasilitas untuk menyimpan data spasial
4	API harus menyediakan fasilitas untuk mengubah data spasial
5	API harus menyediakan fasilitas untuk menghapus data spasial
6	API harus menyediakan fasilitas untuk menghitung panjang geometri
7	API harus menyediakan fasilitas untuk menghitung area geometri
8	API harus menyediakan fasilitas untuk menentukan pusat geometri
9	API harus menyediakan fasilitas untuk mendapatkan pembatas objek geometri
10	API harus menyediakan fasilitas untuk memeriksa apakah 2 objek geometri berpotongan
11	API harus menyediakan fasilitas untuk memeriksa apakah 2 objek geometri bersilangan
12	API harus menyediakan fasilitas untuk menghitung jarak antara 2 objek geometri

13	API harus menyediakan fasilitas untuk mencari perpotongan antara 2 geometri

4.1.2 Perancangan Sistem

4.1.2.1 Perancangan Skema Basis Data Spasial

Penyimpanan data spasial pada suatu sistem basis data menurut Open Geospatial Consortium (OGC) ada 2, yaitu menggunakan *Geometry Type*, dan *Predefined Data Type* [OGC-10]. Penyimpanan menggunakan *Geometry Type* adalah metode yang lebih sederhana. Namun demikian, untuk menggunakan metode ini, DBMS harus memiliki fasilitas bagi pengguna untuk membuat tipe data sendiri (*User defined data types*). Sayangnya MySQL tidak memberikan fasilitas pembuatan *User defined Data Types*, sehingga metode penyimpanan yang digunakan pada API adalah *Predefined Data Types*.

Dengan metode *Predefined Data Types*,.tipe data yang dapat digunakan adalah tipe data bawaan dari MySQL, seperti Integer, Float, Double, Varchar, Char, BLOB, dll. Sesuai dengan spesifikasi OGC untuk metode *Predefined Data Types*, maka tabel – tabel yang dibuat adalah : geometry_columns, spatial_ref_sys, geometry, dan geometry_types sebagai tambahan.

1. geometry_types

   Tabel ini dipergunakan untuk menyimpan data tentang tipe data spasial yang didukung dalam sistem. Atribut tabel ini dapat dilihat pada tabel 4.2.

Tabel 4.2 Atribut Tabel geometry_types

Nama Attribut	Tipe data	Keterangan
ID_GEOMETRY_TYPE.	integer	Menyimpan ID unique untuk setiap tipe geometri.
GEOMETRY_TYPE_NAME.	varchar	Menyimpan nama-nama tipe geometri

2. geometry_columns

Tabel ini dipergunakan untuk menyimpan metadata yang berhubungan dengan tabel geometrid an tabel *feature.* Atribut tabel ini dapat dilihat pada tabel 4.3.

Tabel 4.3 Atribut Tabel geometry_columns

Nama Attribut	Tipe data	Keterangan
F_TABLE_CATALOG	Varchar	Menyimpan data nama catalog tabel *feature*
F_TABLE_SCHEMA	Varchar	Menyimpan data nama schema tabel *feature*;
F_TABLE_NAME	Varchar	Menyimpan data nama tabel *feature*;
F_GEOMETRY_COLUMN.	Varchar	Menyimpan data nama kolom pada tabel feature yang merupakan kolom geometri. Kolom ini mereferensi dari tabel geometri
G_TABLE_CATALOG,		Menyimpan data nama catalog tabel geometri.
G_TABLE_SCHEMA,		Menyimpan data nama schema tabel geometri
G_TABLE_NAME		Menyimpan data nama tabel geometri yang mengimplementasikan kolom geometri.
STORAGE_TYPE.		Menyimpan data tipe penyimpanan yang digunakan pada kolom geometri tertentu:
GEOMETRY_TYPE.		Menyimpan data tipe geometri yang digunakan

		pada suatu kolom geometri. Kolom ini mereferensi tabel geometry_type
COORD_DIMENSION.		Menyimpan data dimensi geometri;
SRID.		Menyimpan informasi *Spatial Reference System* yang digunakan untuk tiap kolom geometri. Kolom ini mereferensi tabel spatial_ref_sys.

3. spatial_ref_sys

   Tabel ini adalah tabel yang menyimpan semua data berkaitan dengan *Spatial Reference System*. Atribut tabel ini dapat dilihat pada tabel 4.4.

Tabel 4.4 Atribut Tabel spatial_ref_sys

Nama Attribut	Tipe data	Keterangan
SRID	Integer	Berupa bilangan bulat yang mengidentifikasi setiap baris data *Spatial Reference System*;
AUTH_NAME	Varchar	Berupa nama yang digunakan sebagai standar penamaan bagi suatu *Spatial Reference System*.;
AUTH_SRID	Integer	ID dari *Spatial Reference System* oleh authornya;
SRTEXT	Varchar	Representasi *Spatial Reference System* berupa *Well Known Text (WKT)*..

4. geometry

Tabel geometri adalah tabel yang digunakan untuk menyimpan semua data spasial. Atribut tabel ini dapat dilihat pada tabel 4.5.

Tabel 4.5 Atribut Tabel geometry

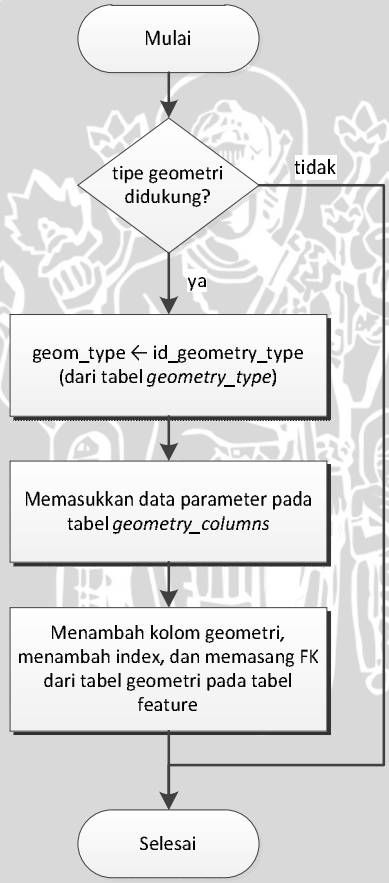
Nama Attribut	Tipe data	Keterangan
GID	Integer	Menyimpan ID unique untuk setiap baris data geometri.
ETYPE	Integer	Menyimpan data bilangan bulat yang merepresentasikan tipe geometri.
ESEQ	Integer	Menyimpan data penanda *sequence*, untuk tipe data geometri yang tersusun atas multi komponen.
X0	Double	Menyimpan data absis pertama penyusun geometri.
Y0	Double	Menyimpan data ordinat pertama penyusun geometri.
X1	Double	Menyimpan data absis kedua penyusun geometri.
Y1	Double	Menyimpan data ordinat kedua penyusun geometri.
X2	Double	Menyimpan data absis ketiga penyusun geometri.
Y2	Double	Menyimpan data ordinat ketiga penyusun geometri.
X3	Double	Menyimpan data absis keempat penyusun geometri.
Y3	Double	Menyimpan data ordinat keempat penyusun geometri.

X4	Double	Menyimpan data absis ke-lima penyusun geometri.
Y4	Double	Menyimpan data ordinat ke-lima penyusun geometri.

4.1.2.2 Perancangan Prosedur Skema

4.1.2.2.1 Prosedur Pembuatan Kolom Geometri

Prosedur ini bertugas membuat kolom untuk menyimpan data objek geometri, dan merelasikan tabel feature dengan tabel geometri. Alur logika prosedur ini digambarkan pada gambar 4.1.

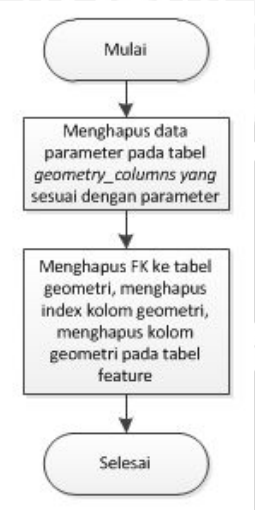


Gambar 4.1 Diagram alir Prosedur tambah_kolom_geometri

4.1.2.2.2 Prosedur Penghapusan Kolom Geometri

Prosedur ini bertugas menghapus kolom geometri. Secara umum, kerja prosedur ini adalah menghapus FK yang terhubung ke tabel geometri, kemudian

menghapus kolom penyimpan objek geometri. Alur logika prosedur ini digambarkan pada gambar 4.2.

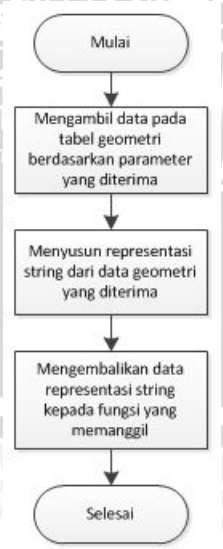


Gambar 4.2 Diagram alir Prosedur hapus_kolom_geometri

4.1.2.3 Perancangan Prosedur Pendukung

4.1.2.3.1 Prosedur penampil data geometri

Prosedur ini digunakan untuk membuat representasi data geometri ke dalam bentuk WKT (Well Known Text). Prosedur ini akan dapat disisipkan pada perintah select pada MySQL. Alur logika prosedur ini ditunjukkan pada gambar 4.3.



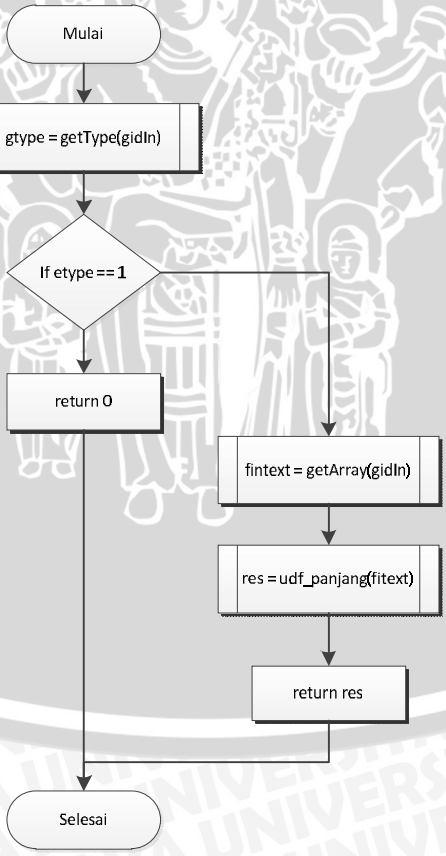
Gambar 4.3 Diagram alir Prosedur penampil data geometri

4.1.2.4 Perancangan Operator Dasar

4.1.2.4.1 Operator Length

Operasi ini menghasilkan panjang dari geometri. Pada API ini, operator ini diberi nama panjang(). Jika parameter yang diberikan pada operator ini adalah objek geometri Garis atau Poligon, operator akan mengembalikan nilai tertentu, sedangkan untuk Titik akan mengembalikan NULL. Alur logikanya digambarkan pada gambar 4.4 dan gambar 4.5.

Fungsi Panjang(): Stored function
Parameter
gidIn : integer
Variabel
gtype : tinyint
fintext : varchar(50)
res : double



Gambar 4.4 Diagram alir Fungsi Operator Length/panjang

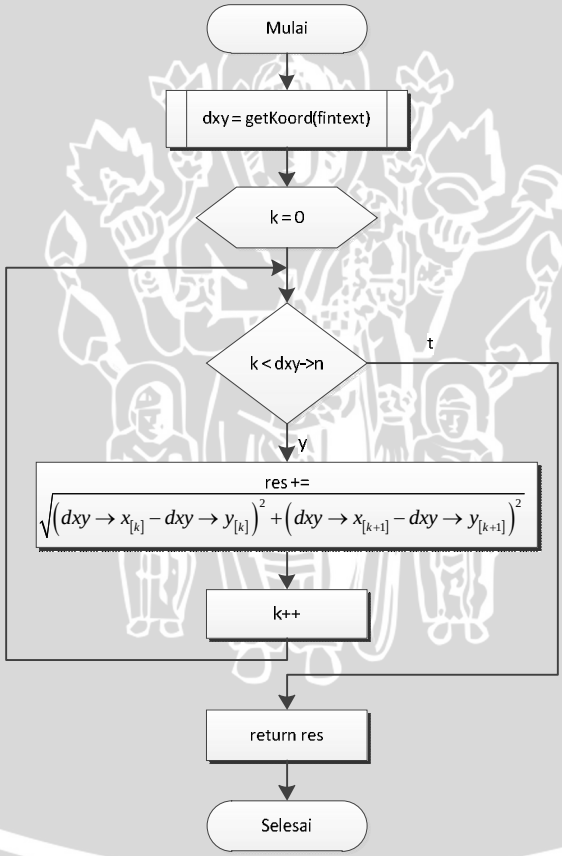
Fungsi udf_panjang() : UDF

Parameter

fintext : char*

Variabel

dxy :
Struct{
dx : double*
dy : double*
n : int
} : XY
res : double
k : int

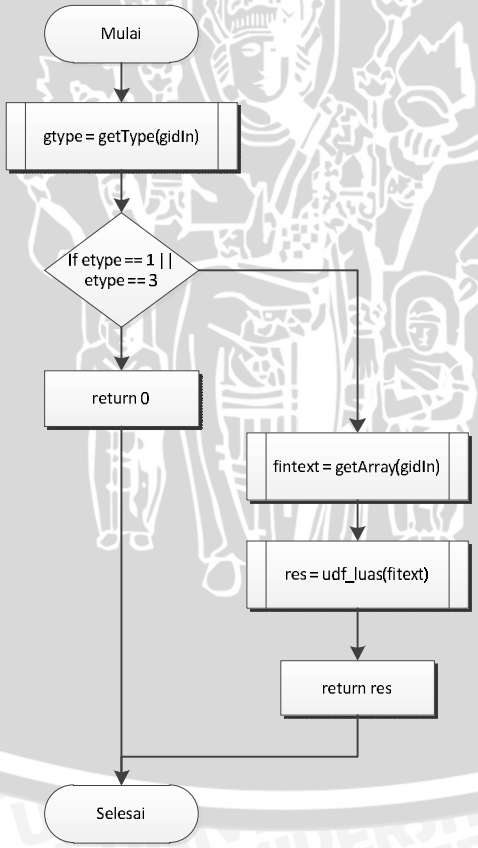


Gambar 4.5 Diagram alir Fungsi udf_panjang

4.1.2.4.2 Operator Area

Operator Area digunakan untuk menghitung luasan geometri 2 dimensi, yaitu Poligon. Pada API ini, operator Area diberi nama Luas. Algoritma operator ini ditunjukkan pada gambar 4.6 dan 4.7:

Fungsi luas(): Stored function
Parameter
gidIn : integer
Variabel
gtype : tinyint
fintext : varchar(50)
res : double
Return
Double



Gambar 4.6 Diagram alir Fungsi Operator Area/Luas

Fungsi udf_luas() : UDF

Parameter

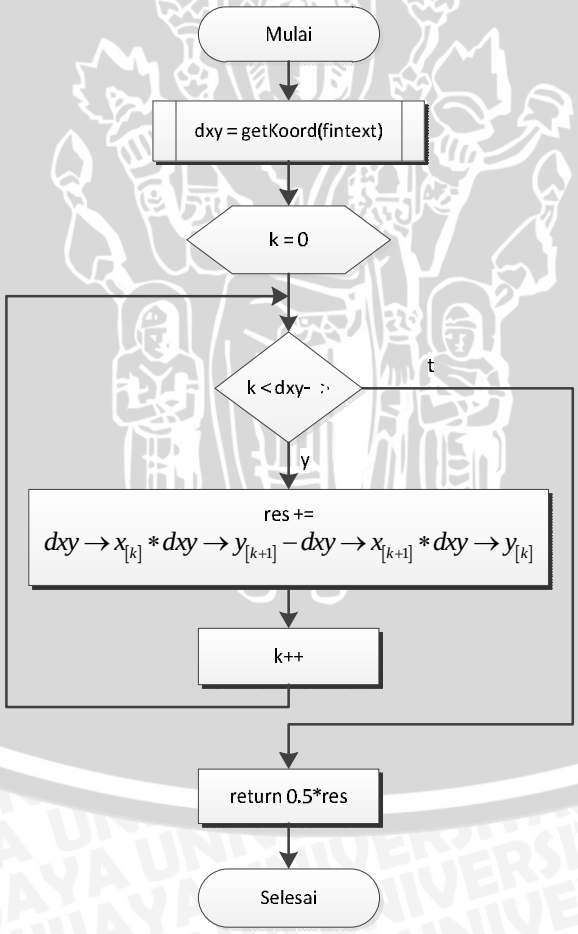
fintext : char*

Variabel

dxy :
Struct{
dx : double*
dy : double*
n : int
} : XY
res : double
k : int

Return

double



Gambar 4.7 Diagram alir Fungsi udf_luas

4.1.2.4.3 Operator Boundary

Operator boundary berfungsi untuk mencari geometri Garis yang membatasi geometri berdimensi 2, misalnya Poligon. Pada API ini, operator Boundary diberi nama Batasan. Operator ini mengembalikan Garis yang tersusun dari titik-titik pembangun Poligon.

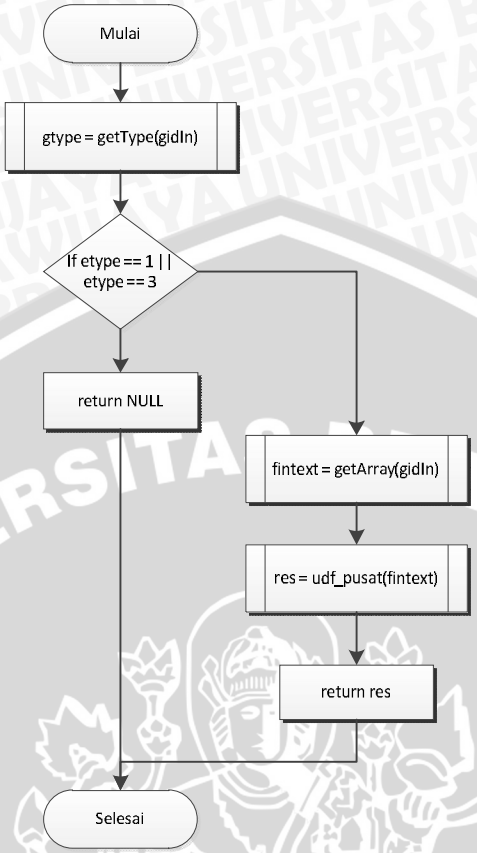
Alur logika operator ini digambarkan pada gambar 4.8.

Gambar 4.8 Diagram alir Fungsi Operator Boundary/Batasan

4.1.2.4.4 Operator Center

Operator ini digunakan untuk mencari titik pusat dari geometri Poligon. Pada API ini, operator Center diberi nama Pusat. Alur logika operator ini digambarakan pada gambar 4.9 dan gambar 4.10.

Fungsi pusat(): Stored function
Parameter
gidIn : integer
Variabel
gtype : tinyint
fintext : varchar(50)
res : double
Return
double



Gambar 4.9 Diagram alir Fungsi Operator Center/Pusat

Fungsi udf_pusat() : UDF

Parameter

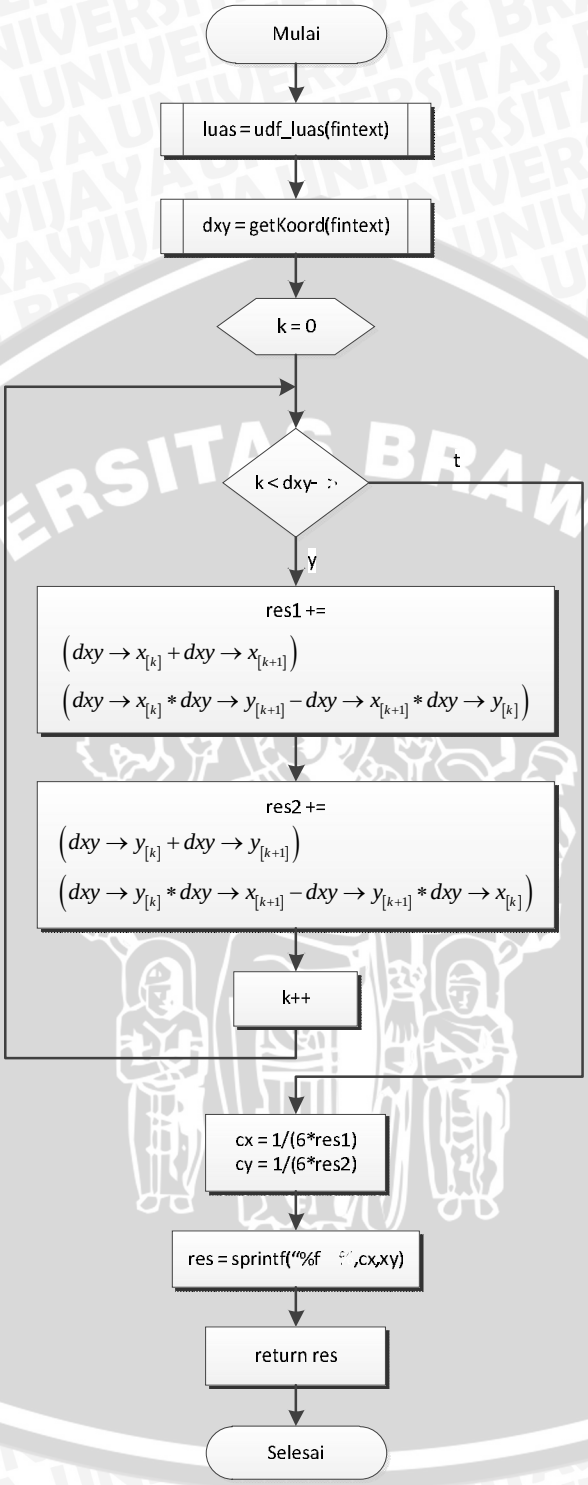
fintext : char*

Variabel

dxy :
Struct{
dx : double*
dy : double*
n : int
} : XY
res1,res2 : double
res : char*
k : int

Return

*char**



Gambar 4.10 Diagram alir Fungsi udf_pusat

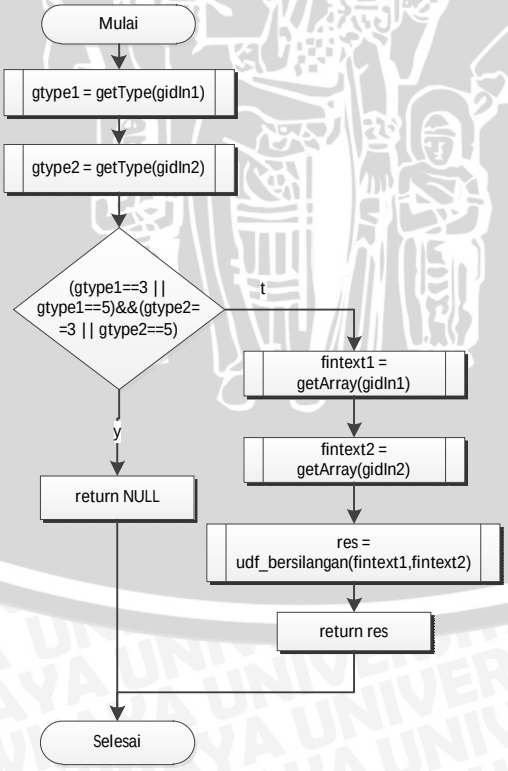
4.1.2.5 Perancangan Operator Topologi

4.1.2.5.1 Operator Cross

Operator ini digunakan untuk mencari tahu apakah dua objek geometri saling bersilangan. Objek geometri yang dapat diproses adalah pasangan objek Garis – Garis atau Garis – Poligon. Pada API ini, operator Cross ini dinamakan Bersilangan.

Alur logika dari operator ini ditunjukkan pada gambar 4.11 dan 4.12.

Fungsi bersilangan(): Stored function
Parameter
gidIn : integer
Variabel
Gtype1,gtype2 : tinyint
fintext1,fintext2 : varchar(50)
res : double
Return
tinyint



Gambar 4.11 Diagram alir Fungsi Operator Cross/Bersilangan

Fungsi udf_bersilangan() : UDF

Parameter

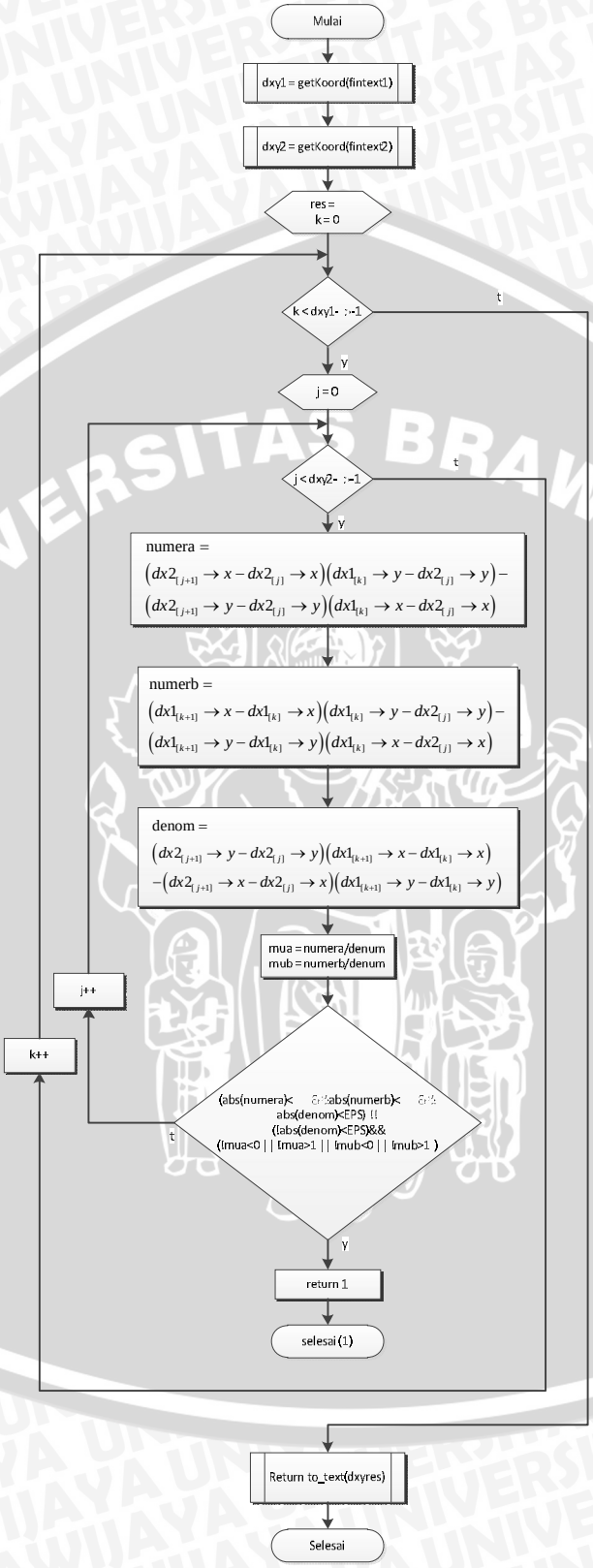
fintext1,fintext2 : char*

Variabel

dxy1,dxy2 :
Struct{
dx : double*
dy : double*
n : int
} : XY
k,j : int
mua,mub,numera,numerb,denom : double

Return

Tinyint



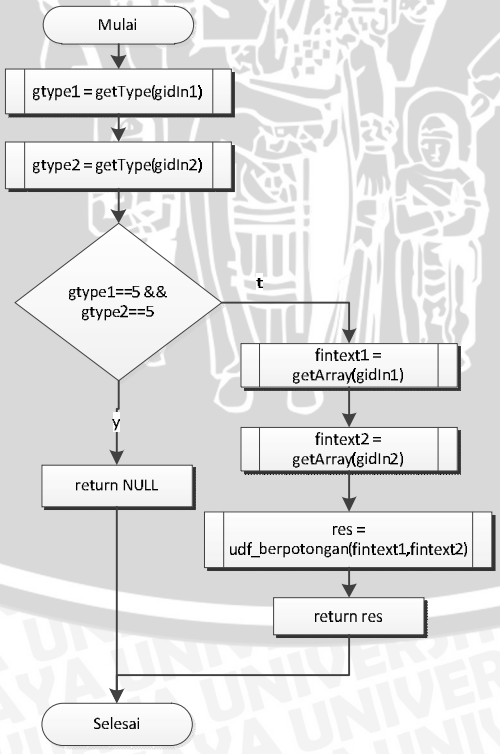
Gambar 4.12 Diagram alir fungsi udf_bersilangan

4.1.2.5.2 Operator Intersect

Operator ini digunakan untuk mencari tahu apakah dua objek geometri 2 dimensi saling berpotongan. Objek geometri yang dapat diproses hanya pasangan objek Poligon – Poligon. Pada API ini, operator Intersect dinamakan Berpotongan.

Alur logika operator ini digambarkan pada gambar 4.13 dan 4.14

Fungsi berpotongan(): Stored function
Parameter
gidIn : integer
Variabel
Gtype1,gtype2 : tinyint
fintext1,fintext2 : varchar(50)
res : double
Return
tinyint



Gambar 4.13 Diagram alir Fungsi Operator Intersect / Berpotongan

Fungsi udf_berpotongan() : UDF

Parameter

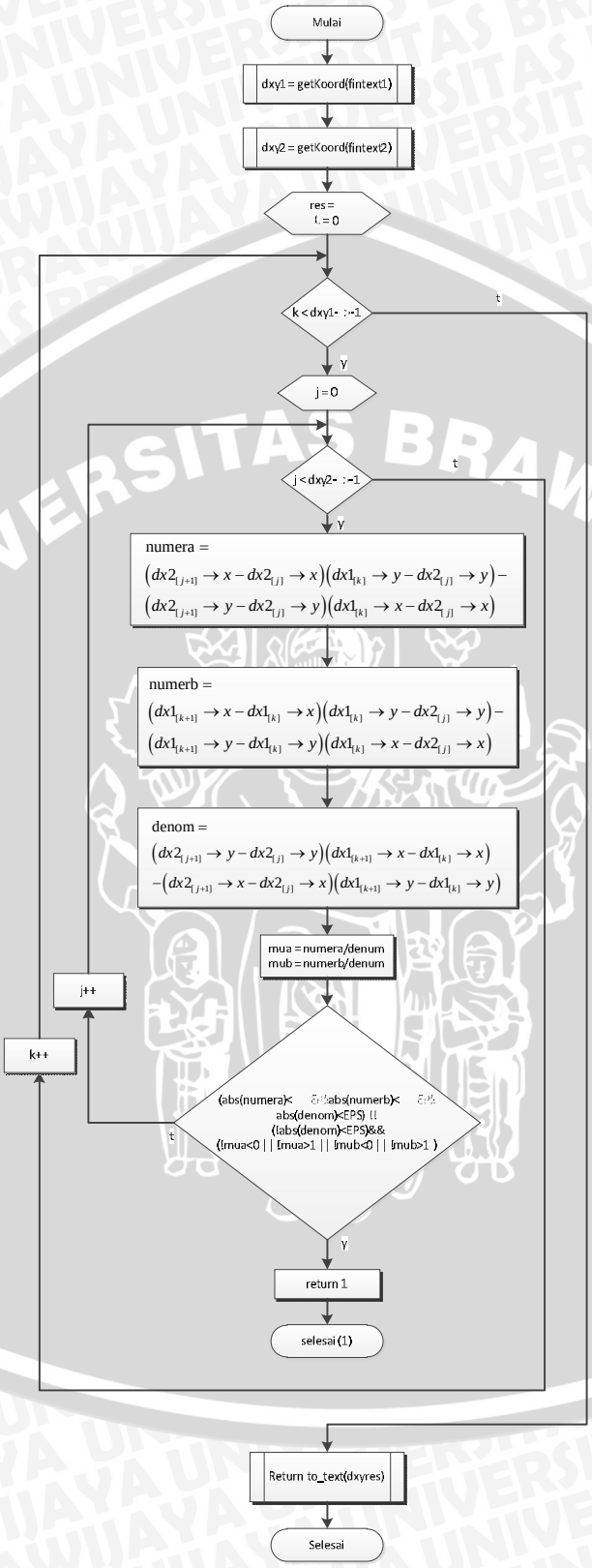
fintext1,fintext2 : char*

Variabel

dxy1,dxy2 :
Struct{
dx : double*
dy : double*
n : int
} : XY
k,j : int
mua,mub,numera,numerb,denom : double

Return

Tinyint



Gambar 4.14 Diagram alir Fungsi udf_berpotongan

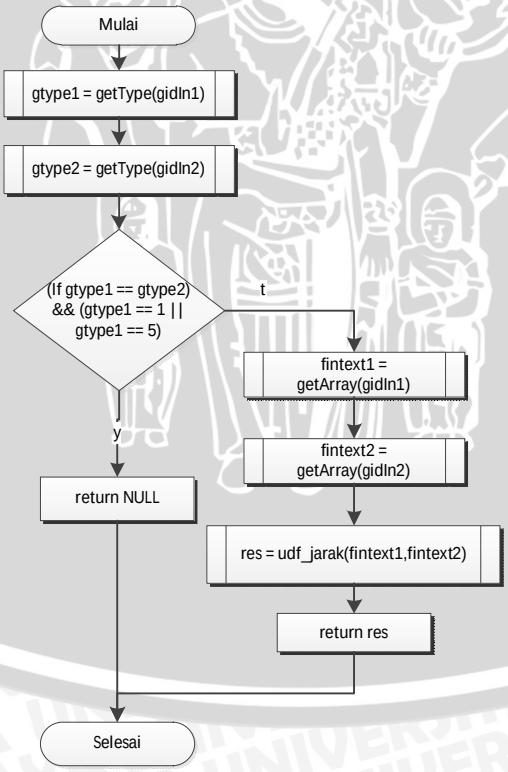
4.1.2.6 Perancangan Operator Analisis Spasial

4.1.2.6.1 Operator Distance

Operator ini digunakan untuk menghitung jarak antara dua objek geometri. Objek geometri yang dapat diproses adalah pasangan objek Titik – Titik, Titik – Poligon, atau Poligon - Poligon. Pada API ini, operator Distance dinamakan Jarak.

Alur logika operator ini digambarkan pada gambar 4.15 dan 4.16

Fungsi jarak(): Stored function
Parameter
gidIn : integer
Variabel
gtype1,gtype2 : tinyint
fintext1,fintext2 : varchar(50)
res : double
Return
double



Gambar 4.15 Diagram alir Fungsi Operator Distance/Jarak

Fungsi udf_jarak() : UDF

Parameter

fintext1,fintext2 : char*

Variabel

*titik1,titik2 : char**

dxy1,dxy2 :

Struct{

dx : double*

dy : double*

n : int

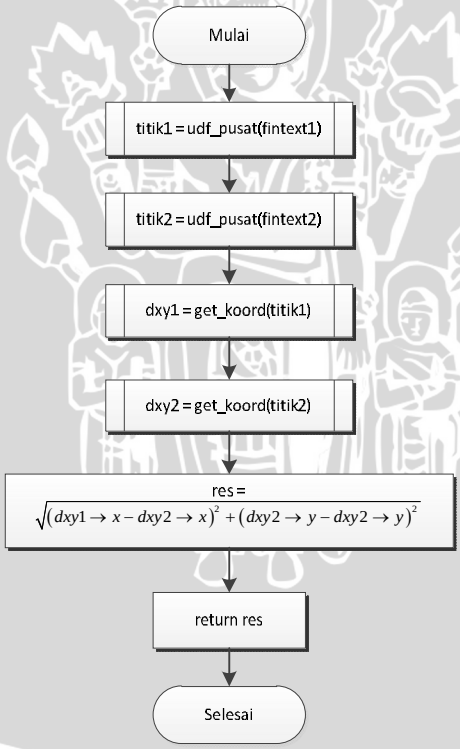
} : XY

k : int

res : double

Return

double



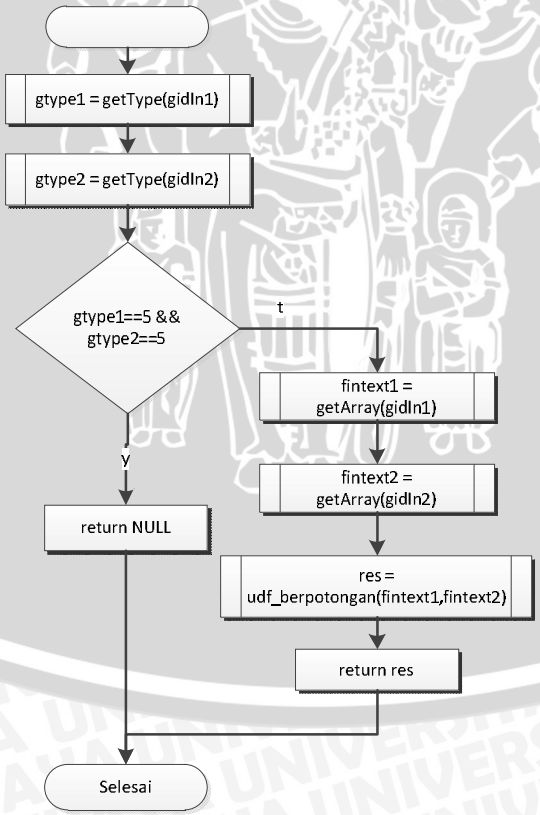
Gambar 4.16 Diagram alir Fungsi udf_jarak

4.1.2.6.2 Operator Intersection

Operator ini digunakan untuk mencari daerah perpotongan antara 2 objek geometri 2 dimensi, yaitu Poligon. Objek geometri yang dapat diproses hanya pasangan Poligon - Poligon. Pada API ini, operator Intersection dinamakan Perpotongan.

Alur logika operator ini digambarkan pada gambar 4.17 dan 4.18.

Fungsi perpotongan(): Stored function
Parameter
gidIn : integer
Variabel
gtype1,gtype2 : tinyint
fintext1,fintext2 : varchar(50)
res : double
Return
varchar



Gambar 4.17 Diagram alir Fungsi Operator Intersection / Perpotongan

Fungsi udf_perpotongan() : UDF

Parameter

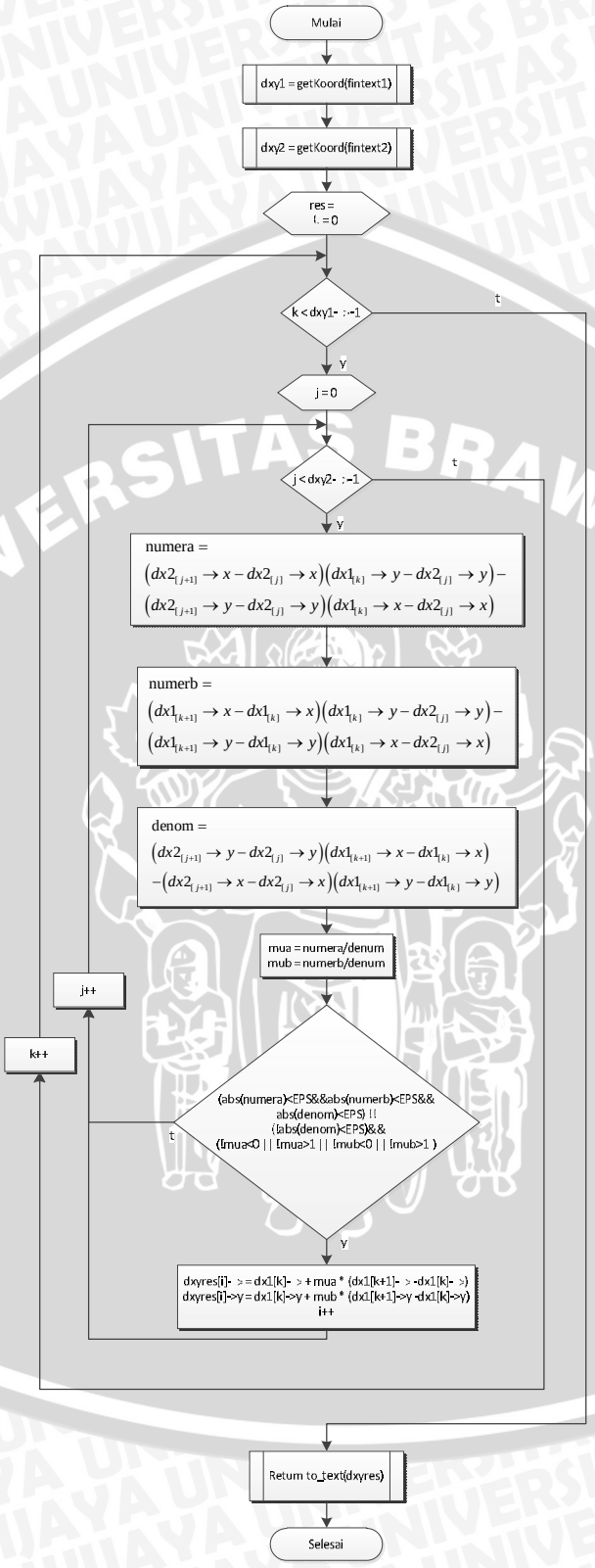
fintext : char*

Variabel

dxy1, dxy2, dxyres :
Struct{
dx : double*
dy : double*
} : XY
res1,res2 : double
res : char*
j,k,i : int

Return

*char**



Gambar 4.18 Diagram alir Fungsi udf_perpotongan

4.2 Implementasi

Setelah tahap perancangan selesai dilakukan, tahap selanjutnya adalah tahap implementasi. Tujuan dari tahap implementasi ini merupakan proses transformasi hasil perancangan perangkat lunak. Pembahasan terdiri dari lingkungan implementasi (spesifikasi perangkat lunak dan perangkat keras), implementasi antarmuka aplikasi dengan sintaks dari bahasa pemrograman yang digunakan.

4.2.1 Lingkungan Implementasi

API dibuat dalam 2 bahasa, yaitu PL/SQL Stored Routine dan bahasa pemrograman C. Penulis menggunakan lingkungan implementasi dengan perangkat keras berupa sebuah laptop.

4.2.1.1 Lingkungan Implementasi Perangkat Keras

Sebuah laptop dengan spesifikasi sebagai berikut:

- *Processor* : Intel(R) Core(TM) 2 Duo-T5750 2 CPU @2.00 GHz
- *Memory* : 2048 MB RAM
- *System type* : 32-bit Operating System
- *System model* : Compaq v3908

4.2.1.2 Lingkungan Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan:

- Sistem operasi : Windows XP SP2 32-bit
- RDBMS : MySQL 5.1.33
- Bahasa pemrograman : C
- *Integrated Development Environment* (IDE) : Microsoft Visual C++ 2008 Express Edition

4.5.2 Implementasi Prosedur *Library*

API ini dibuat menggunakan 2 bahasa, PL/SQL dan bahasa C. PL/SQL dipergunakan untuk membuat Stored Routine. Stored Routine diperlukan untuk mengambil data spasial pada tabel geometri. Meskipun demikian, semua proses yang melibatkan perhitungan diimplementasikan menggunakan bahasa C. Semua

prosedur yang ditulis dengan bahasa C akan diterjemahkan oleh compiler dan dibangun dengan pendekatan *procedural programming*. Hasil compile tidak diarahkan menjadi file *executable*, namun menjadi file DLL. Dengan demikian, API akan dihasilkan dalam 2 bentuk, yaitu kumpulan Stored Routines dan sebuah file DLL.

4.5.2.1 Implementasi Prosedur Penambahan Kolom Geometri

Prosedur ini digunakan untuk menambahkan kolom geometri ke dalam tabel feature. Prosedur ini diimplementasikan menggunakan PL/SQL, berupa Stored Procedure dengan nama **tambah_kolom_geometri**. Prosedur ini tidak memberikan nilai kembalian. Deskripsi prosedur ditampilkan pada gambar 4.19.

Deskripsi tambah_kolom_geometri:

```
1   DECLARE cek TINYINT;
2   DECLARE geom_type TINYINT;
3
4   SET geom_type = 0;
5   SET cek = 0;
6
7   SELECT COUNT(*) INTO cek FROM geometry_type WHERE
8   geometry_type_name LIKE tipe_geometri;
9
10  IF cek > 0 THEN
11  SELECT id_geometry_type INTO geom_type FROM geometry_type
12  WHERE geometry_type_name LIKE tipe_geometri;
13
14  SET @ddl = CONCAT('ALTER TABLE ',nama_tabel,' ADD COLUMN
15  ',nama_kolom_geom,' INT UNSIGNED NOT NULL, ADD INDEX fk_',
16  nama_tabel,'_geometri (',nama_kolom_geom,'), ADD CONSTRAINT
17  fk_',nama_tabel,'_geometri FOREIGN KEY (',nama_kolom_geom,')
18  REFERENCES geometri(gid) ON DELETE RESTRICT ON
19  UPDATE CASCADE');
20
21  PREPARE stmt FROM @ddl;
22  EXECUTE stmt;
23  DEALLOCATE PREPARE stmt;
24
25  INSERT INTO geometry_columns (f_table_name,
26  f_geometry_column, g_table_name, geometry_type,
27  coord_dimension, srid)
28  VALUES(nama_tabel,nama_kolom_geom,'geometri',geom_type,
29  dimension,sridin);
30
31  END IF;
```

Gambar 4.19 Deskripsi Prosedur tambah_kolom_geometri

Berikut ini adalah penjelasan dari prosedur tambah_kolom_geometri:

1. Baris 1 s.d. 5 adalah deklarasi dan inisialisasi variable yang akan digunakan, yaitu cek dan geom._type
2. Baris 7 s.d. 12 adalah pengecekan apakah tipe geometri yang dimasukkan ke dalam parameter fungsi sesuai dengan yang ada dalam database. Jika ada maka diambil id_geometri nya dan dimasukkan ke dalam variable geom_type.
3. Baris 14 s.d. 19 adalah penyusunan query alter tabel untuk menambahkan kolom pada tabel feature, menambahkan index, dan membuat foreign key ke tabel geometri.
4 Baris 21 s.d. 23 adalah membuat query alter tabel tadi menjadi *prepared statement*, pengeksekusian, dan pembebasan *prepared statement.*
5 Baris 25 s.d. 29 adalah query untuk memasukkan metadata tentang kolom geometri yang baru saja dibuat, tabel geometry, dimensi, dan SRID.

4.5.2.2 Implementasi Prosedur Penghapusan Kolom Geometri

Prosedur ini digunakan untuk menghapus kolom geometri ke dalam tabel feature. Prosedur ini diimplementasikan menggunakan PL/SQL, berupa Stored Procedure dengan nama **hapus_kolom_geometri**. Prosedur ini tidak memberikan nilai kembalian. Deskripsi prosedur ditampilkan pada gambar 4.20.

Deskripsi hapus_kolom_geometri:

```
1   SET @ddl = CONCAT('ALTER TABLE ',nama_tabel,' DROP COLUMN
2   ',nama_kolom_geom);
3   PREPARE stmt FROM @ddl;
4   EXECUTE stmt;
5   DEALLOCATE PREPARE stmt;
6
7   DELETE FROM geometry_columns WHERE f_table_name =
8   nama_tabel AND f_geometry_column = nama_kolom_geom;
```

Gambar 4.20 Deskripsi prosedur hapus_kolom_geometri

Berikut ini adalah penjelasan dari prosedur hapus_kolom_geometri:

1. Baris 1 s.d. 2 adalah penyusunan query untuk menghapus kolom geometri dari tabel feature.

2. Baris 3 s.d. 5 adalah pembuatan *prepared statement* dari query yang dibuat pada baris 1 s.d. 2, pengeksekusian, dan pembebasan *prepared statement*.
3 Baris 7 s.d. 8 adalah penghapusan metadata dari kolom geometri yang dihapus, pada tabel *geometry_column*.

4.5.2.3 Implementasi Prosedur Penampil Data Geometri

Prosedur ini digunakan untuk menampilkan data spasial dalam bentuk WKT. Prosedur ini diimplementasikan menggunakan PL/SQL, berupa Stored Procedure dengan nama **to_text**. Deskripsi prosedur ditampilkan pada gambar 4.21.

Deskripsi to_text:

```
 1  DECLARE tmptext VARCHAR(50);
 2  DECLARE fintext VARCHAR(100);
 3  DECLARE rettext VARCHAR(100);
 4  DECLARE m_etype SMALLINT;
 5  DECLARE last_fetched TINYINT;
 6
 7  DECLARE cur_geom CURSOR FOR SELECT
 8  CONCAT(IFNULL(x0,'NULL'),' ',IFNULL(y0,'NULL')
 9  ,',',IFNULL(x1,'NULL'),'
10  ',IFNULL(y1,'NULL'),',',IFNULL(x2,'NULL'),'
11  ',IFNULL(y2,'NULL'),',',IFNULL(x3,'NULL'),'
12  ',IFNULL(y3,'NULL'),',',IFNULL(x4,'NULL'),'
13  ',IFNULL(y4,'NULL'))
14  FROM geometri WHERE gid = gidIn;
15
16  DECLARE CONTINUE HANDLER FOR NOT FOUND SET last_fetched=1;
17
18  SET last_fetched = 0;
19  SET fintext = '';
20
21  SELECT etype INTO m_etype FROM geometri WHERE gid = gidIn;
22  SELECT geometry_type_nama INTO geoname FROM geometry_types
23  WHERE id_geometrY_type = etype;
24
25  OPEN cur_geom;
26
27  cursor_loop:LOOP
28        FETCH cur_geom INTO tmptext;
29        SET fintext = CONCAT(fintext,',',tmptext);
30        IF last_fetched = 1 THEN
31              LEAVE cursor_loop;
32        END IF;
33  END LOOP;
34
35  CLOSE cur_geom;
36
37  IF INSTR(fintext,'NULL') > 0 THEN
```

```
38        SET rettext = SUBSTR(fintext,2,INSTR(fintext,'NULL')-
39  3);
40  ELSE
41        SET rettext = SUBSTR(fintext,2,LENGTH(fintext));
42  END IF;
43
44  RETURN CONCAT(geoname,'('rettext,')');
45
46  END IF;
```

Gambar 4.21 Deskripsi prosedur to_text

Berikut ini adalah penjelasan dari prosedur to_text:

1. Baris 7 s.d. 14 adalah pengambilan data geometri (berupa kumpulan koordinat penyusun geometri) dari tabel geometri.

3 Baris 25 s.d. 35 adalah proses looping untuk penggabungan data geometri menjadi bentuk string tunggal. Looping akan berhenti jika tidak ditemui lagi record lain dengan id yang sama.

4 Baris 37 s.d. 41 adalah penyempurnaan isi string tunggal sebelum dikembalikan kepada pengguna.

5 Baris 44 adalah melakukan penggabungan string untuk mendapatkan format WKT geometri. Hasil penggabungan string tersebut dikembalikan kepada fungsi pemanggil.

4.5.2.4 Implementasi Operator Panjang

Operator panjang dirancang untuk bekerja dengan satu parameter dengan tipe data integer. Parameter yang dimaksud adalah nilai dari kolom geometri pada tabel feature. Operator panjang diimplementasikan dengan 2 cara atau 2 fungsi. Fungsi pertama akan berperan sebagai pengambil data spasial dari tabel geometri dan penampil hasil perhitungan panjang, sedangkan fungsi ke-dua berperan sebagai penghitung panjang.

Fungsi pertama diimplementasikan menggunakan PL/SQL berupa Stored Procedure dengan nama **panjang**. Fungsi ini akan mengambil data koordinat penyusun geometri yang terdapat pada tabel geometri kemudian mengoperkan data kumpulan koordinat tersebut kepada fungsi ke-dua untuk diproses. Kumpulan koordinat tersebut harus dibentuk sedemikian rupa sehingga dapat dioper ke fungsi ke-dua.

Fungsi ke-dua diimplementasikan menggunakan bahasa C dengan nama **udf_panjang**. Fungsi ini akan memproses parameter, merepresentasikan kumpulan koordinat ke dalam array struct, melakukan kalkulasi dengan formula penghitungan panjang, kemudian mengembalikan hasil perhitungan ke fungsi **panjang**. Secara keseluruhan, operator ini memberikan nilai kembalian berupa bilangan pecahan (double). Deskripsi operator ini ditampilkan pada gambar 4.22 dan 4.23.

Deskripsi panjang:

```
 1  DECLARE tmptext VARCHAR(50);
 2  DECLARE fintext VARCHAR(100);
 3  DECLARE rettext VARCHAR(100);
 4  DECLARE m_etype SMALLINT;
 5  DECLARE last_fetched TINYINT;
 6
 7  DECLARE cur_geom CURSOR FOR SELECT
 8  CONCAT(IFNULL(x0,'NULL'),' ',IFNULL(y0,'NULL')
 9  ,',',IFNULL(x1,'NULL'),'
10  ',IFNULL(y1,'NULL'),',',IFNULL(x2,'NULL'),'
11  ',IFNULL(y2,'NULL'),',',IFNULL(x3,'NULL'),'
12  ',IFNULL(y3,'NULL'),',',IFNULL(x4,'NULL'),'
13  ',IFNULL(y4,'NULL'))
14  FROM geometri WHERE gid = gidIn;
15
16  DECLARE CONTINUE HANDLER FOR NOT FOUND SET last_fetched=1;
17
18  SET last_fetched = 0;
19  SET fintext = '';
20
21  SELECT etype INTO m_etype FROM geometri WHERE gid = gidIn;
22  IF (m_etype != 3 || m_etype != 5) THEN
23        RETURN NULL;
24
25  ELSE
26  OPEN cur_geom;
27
28  cursor_loop:LOOP
29        FETCH cur_geom INTO tmptext;
30        SET fintext = CONCAT(fintext,',',tmptext);
31        IF last_fetched = 1 THEN
32              LEAVE cursor_loop;
33        END IF;
34  END LOOP;
35
36  CLOSE cur_geom;
37
38  IF INSTR(fintext,'NULL') > 0 THEN
39        SET rettext = SUBSTR(fintext,2,INSTR(fintext,'NULL')-
40  3);
41  ELSE
42        SET rettext = SUBSTR(fintext,2,LENGTH(fintext));
43  END IF;
```

```
44
45  RETURN udf_panjang(rettext);
46
47  END IF;
```

Gambar 4.22 Deskripsi fungsi panjang

Berikut ini adalah penjelasan dari fungsi panjang:

1. Baris 7 s.d. 14 adalah pengambilan data geometri (berupa kumpulan koordinat penyusun geometri) dari tabel geometri.
2. Baris 21 s.d. 22 adalah proses pemeriksaan tipe geometri dari objek geometri yang dioperasikan. Apabila objek geometri bukan Garis(3) atau Poligon(5), maka fungsi mengembalikan NULL, jika tidak proses berlanjut.

3 Baris 26 s.d. 34 adalah proses looping untuk penggabungan data geometri menjadi bentuk string tunggal. Looping akan berhenti jika tidak ditemui lagi record lain dengan id yang sama.

4 Baris 38 s.d. 43 adalah penyempurnaan isi string tunggal sebelum dioperkan ke fungsi udf_panjang.

5 Baris 45 adalah pengoperan string sebagai parameter pada fungsi udf_panjang.

Deskripsi udf_panjang:

```
 1  char *kopian, *p, *pend;
 2  char *inp = args->args[0];
 3  char **arrayku;
 4  int i=0, j, cont=0;
 5  XY *dxy;
 6  double hasil = 0;
 7
 8  kopian = (char*) malloc((strlen(args->args[0])) *
 9  sizeof(char));
10  strcpy(kopian,args->args[0]);
11
12  while(*inp){
13  if (isspace(*inp))
14        cont++;
15        inp++;
16  }
17
18  arrayku = (char**) malloc((cont+1)*sizeof(char[15]));
19  dxy = malloc(cont*sizeof(XY));
20  if(arrayku == NULL || dxy == NULL){
```

```
21        *error = 1;
22        exit(1);
23  }
24
25  p=strtok(kopian,",");
26  while (p!=NULL){
27        arrayku[i] = strdup(p);
28        p=strtok(NULL,",");
29        i++;
30  }
31
32  for(j=0;j<i;j++){
33        dxy[j].x = strtod(arrayku[j],&pend);
34        dxy[j].y = strtod(pend,NULL);
35  }
36
37  for(j=0;j<i-1;j++){
38        hasil += sqrt(pow((dxy[j].x - dxy[j+1].x),2) +
39  pow((dxy[j].y - dxy[j+1].y),2));
40  }
41
42  free(arrayku);
43  free(kopian);
44  free(p);
45  free(pend);
46  free(dxy);
47
48  return hasil;
```

Gambar 4.23 Deskripsi fungsi udf_panjang

Berikut ini adalah penjelasan dari fungsi udf_panjang:

1. Baris 8 s.d. 30 adalah proses pemecahan parameter fungsi (string berisi kumpulan koordinat) berdasarkan tanda koma, dan menyimpan ke dalam array character 2 dimensi bernama arrayku.
2. Baris 31 s.d. 35 adalah proses penyimpanan data koordinat dari variable arrayku ke array struct XY bernama dxy.
3 Baris 37 s.d. 40 adalah proses looping untuk menghitung panjang. Looping digunakan dalam upaya mencari panjang dari setiap segmen yang ada pada Garis.
4 Baris 42 s.d. 46 adalah proses pembebasan memori yang semula digunakan oleh variable-variabel pointer.
5 Baris 48 adalah pengembalian nilai hasil perhitungan ke fungsi pemanggil (panjang)

4.5.2.5 Implementasi Operator Luas

Operator Luas diimplementasikan ke dalam 2 fungsi. Fungsi pertama bernama **Luas** dibuat menggunakan PL/SQL. Sedangkan fungsi kedua bernama **udf_luas** diimplementasikan menggunakan bahasa C. Seperti pada Operator Panjang, fungsi **Luas** digunakan sebagai pengambil data geometri dari database, dan sebagai pengembali hasil perhitungan. Formula perhitungan luas diimplementasikan pada fungsi **udf_luas**. Deskripsi fungsi luas dan udf_luas ditampilkan pada gambar 4.24 dan 4.25.

Deskripsi luas:

```
 1  DECLARE tmptext VARCHAR(50);
 2  DECLARE fintext VARCHAR(100);
 3  DECLARE rettext VARCHAR(100);
 4  DECLARE m_etype SMALLINT;
 5  DECLARE last_fetched TINYINT;
 6
 7  DECLARE cur_geom CURSOR FOR SELECT
 8  CONCAT(IFNULL(x0,'NULL'),' ',IFNULL(y0,'NULL')
 9  ,',',IFNULL(x1,'NULL'),'
10  ',IFNULL(y1,'NULL'),',',IFNULL(x2,'NULL'),'
11  ',IFNULL(y2,'NULL'),',',IFNULL(x3,'NULL'),'
12  ',IFNULL(yz3,'NULL'),',',IFNULL(x4,'NULL'),'
13  ',IFNULL(y4,'NULL'))
14  FROM geometri WHERE gid = gidIn;
15
16  DECLARE CONTINUE HANDLER FOR NOT FOUND SET last_fetched=1;
17
18  SET last_fetched = 0;
19  SET fintext = '';
20
21  SELECT etype INTO m_etype FROM geometri WHERE gid = gidIn;
22  IF (m_etype != 5) THEN
23        RETURN NULL;
24
25  ELSE
26  OPEN cur_geom;
27
28  cursor_loop:LOOP
29        FETCH cur_geom INTO tmptext;
30        SET fintext = CONCAT(fintext,',',tmptext);
31        IF last_fetched = 1 THEN
32              LEAVE cursor_loop;
33        END IF;
34  END LOOP;
35
36  CLOSE cur_geom;
37
38  IF INSTR(fintext,'NULL') > 0 THEN
39        SET rettext = SUBSTR(fintext,2,INSTR(fintext,'NULL')-
40  3);
41  ELSE
```

```
42        SET rettext = SUBSTR(fintext,2,LENGTH(fintext));
43  END IF;
44
45  RETURN udf_panjang(rettext);
46
47  END IF;
48
```

Gambar 4.24 Deskripsi fungsi luas

Berikut ini adalah penjelasan dari fungsi luas.

1. Baris 7 s.d. 14 adalah pengambilan data geometri (berupa kumpulan koordinat penyusun geometri) dari tabel geometri.
2. Baris 21 s.d. 22 adalah proses pemeriksaan tipe geometri dari objek geometri yang dioperasikan. Apabila objek geometri bukan Poligon(5), maka fungsi mengembalikan NULL, jika tidak proses berlanjut.
3 Baris 26 s.d. 36 adalah proses looping untuk penggabungan data geometri menjadi bentuk string tunggal. Looping akan berhenti jika tidak ditemui lagi record lain dengan id yang sama.
4 Baris 38 s.d. 43 adalah penyempurnaan isi string tunggal sebelum dioperkan ke fungsi udf_luas.
5 Baris 45 adalah pengoperan string sebagai parameter pada fungsi udf_luas.

Deskripsi udf_luas:

```
1  .
2  .
3  .
4  for(j=0;j<i-1;j++){
5        hasil += dxy[j].x * dxy[j+1].y - dxy[j+1].x *
6  dxy[j].y;
7  }
8
9  return(hasil < 0 ? -(hasil/2) : hasil/2);
```

Gambar 4.25 Deskripsi fungsi udf_luas

Berikut ini adalah penjelasan dari fungsi udf_luas.

1. Pada fungsi udf_luas terjadi juga seperti pada fungsi udf_panjang, yaitu proses pemecahan string parameter berisi kumpulan koordinat ke dalam struct koordinat
2 Baris 4 s.d. 6 adalah proses penghitungan luas daerah.
3. Baris 9 adalah proses pengembalian hasil penghitungan dengan terlebih dahulu membagi hasil penghitungan dengan 2. Kembaliannya adalah nilai absolute dari hasil penghitungan.

4.5.2.6 Implementasi Operator Batasan

Operator ini bertugas mengembalikan geometri yang menjadi pembatas dari geometri 2 dimensi, yaitu Poligon. Prosedur ini diimplementasikan menggunakan PL/SQL, berupa Stored Procedure dengan nama **batasan**. Prosedur ini memberikan nilai kembalian berupa format WKT dari geometri Garis pembatas Poligon. Deskripsi operator ini ditampilkan pada gambar 4.26.

```
Deskripsi batasan:
```

```
 1  DECLARE tmptext VARCHAR(50);
 2  DECLARE fintext VARCHAR(100);
 3  DECLARE rettext VARCHAR(100);
 4  DECLARE m_etype MEDIUMINT;
 5  DECLARE last_fetched TINYINT;
 6
 7  DECLARE cur_geom CURSOR FOR SELECT
 8  CONCAT(IFNULL(x0,'NULL'),' ',IFNULL(y0,'NULL')
 9  ,',',IFNULL(x1,'NULL'),'
10  ',IFNULL(y1,'NULL'),',',IFNULL(x2,'NULL'),'
11  ',IFNULL(y2,'NULL'),',',IFNULL(x3,'NULL'),'
12  ',IFNULL(y3,'NULL'),',',IFNULL(x4,'NULL'),'
13  ',IFNULL(y4,'NULL'))
14  FROM geometri WHERE gid = gidIn;
15
16  DECLARE CONTINUE HANDLER FOR NOT FOUND SET last_fetched=1;
17
18  SET last_fetched = 0;
19  SET fintext = '';
20
21  SELECT etype INTO m_etype FROM geometri WHERE gid = gidIn;
22  IF (m_etype != 5) THEN
23        RETURN NULL;
24  ELSE
25
26  OPEN cur_geom;
27
28  cursor_loop:LOOP
29        FETCH cur_geom INTO tmptext;
30        IF last_fetched = 1 THEN
31              LEAVE cursor_loop;
32        END IF;
33        SET fintext = CONCAT(fintext,',',tmptext);
34  END LOOP;
35  CLOSE cur_geom;
36
37  IF INSTR(fintext,'NULL') > 0 THEN
38  SET rettext = SUBSTR(fintext,2,INSTR(fintext,'NULL')-3);
39  ELSE
40  SET rettext = SUBSTR(fintext,2,LENGTH(fintext));
41  END IF;
42
43  RETURN CONCAT('GARIS(',rettext,')');
44  END IF;
```

Gambar 4.26 Deskripsi fungsi batasan

Berikut ini adalah penjelasan dari fungsi batasan.

1. Baris 7 s.d. 14 adalah pengambilan data geometri (berupa kumpulan koordinat penyusun geometri) dari tabel geometri.
2. Baris 21 s.d. 23 adalah proses pemeriksaan tipe geometri dari objek geometri yang dioperasikan. Apabila objek geometri bukan

Poligon(5), maka fungsi mengembalikan NULL, jika tidak proses berlanjut.

3 Baris 26 s.d. 36 adalah proses looping untuk penggabungan data geometri menjadi bentuk string tunggal. Looping akan berhenti jika tidak ditemui lagi record lain dengan id yang sama.

4 Baris 38 s.d. 43 adalah penyempurnaan isi string tunggal sebelum dikembalikan kepada pengguna.

5 Baris 43 adalah melakukan penggabungan string untuk mendapatkan format WKT geometri pembatas Poligon, yaitu Garis. Hasil penggabungan string tersebut dikembalikan kepada pengguna.

4.5.2.7 Implementasi Operator Pusat

Operator Pusat diimplementasikan menggunakan 2 fungsi, yaitu fungsi **pusat** dan fungsi **udf_pusat**. Fungsi pusat dibangun menggunakan PL/SQL sedangkan fungsi udf_pusat menggunakan bahasa C. Seperti operator sebelumnya, fungsi pusat bertugas mengambil data geometri dari tabel geometri, manyusun data geometri ke dalam bentuk string tunggal, kemudian mengoperkannya ke fungsi udf_pusat. Pada fungsi udf_pusat, dilakukan kalkulasi untuk mencari pusat geometri. Deskripsi operator ini ditampilkan pada gambar 4.27 dan 4.28.

Deskripsi pusat:

```
 1  DECLARE tmptext VARCHAR(50);
 2  DECLARE fintext VARCHAR(100);
 3  DECLARE rettext VARCHAR(100);
 4  DECLARE m_etype SMALLINT;
 5  DECLARE last_fetched TINYINT;
 6
 7  DECLARE cur_geom CURSOR FOR SELECT
 8  CONCAT(IFNULL(x0,'NULL'),' ',IFNULL(y0,'NULL')
 9  ,',',IFNULL(x1,'NULL'),'
10  ',IFNULL(y1,'NULL'),',',IFNULL(x2,'NULL'),'
11  ',IFNULL(y2,'NULL'),',',IFNULL(x3,'NULL'),'
12  ',IFNULL(y3,'NULL'),',',IFNULL(x4,'NULL'),'
13  ',IFNULL(y4,'NULL'))
14  FROM geometri WHERE gid = gidIn;
15
16  DECLARE CONTINUE HANDLER FOR NOT FOUND SET last_fetched=1;
17
18  SET last_fetched = 0;
```

```
19  SET fintext = '';
20
21  SELECT etype INTO m_etype FROM geometri WHERE gid = gidIn;
22  IF (m_etype != 5) THEN
23        RETURN NULL;
24  ELSE
25
26  OPEN cur_geom;
27
28  cursor_loop:LOOP
29        FETCH cur_geom INTO tmptext;
30        SET fintext = CONCAT(fintext,tmptext);
31        IF last_fetched = 1 THEN
32              LEAVE cursor_loop;
33        END IF;
34  END LOOP;
35
36  CLOSE cur_geom;
37
38  IF INSTR(fintext,'NULL') > 0 THEN
39  SET rettext = SUBSTR(fintext,2,INSTR(fintext,'NULL')-3);
40  ELSE
41  SET rettext = SUBSTR(fintext,2,LENGTH(fintext));
42  END IF;
43
44  RETURN CONCAT('TITIK(',udf_pusat(rettext),')');
45  END IF;
46
47
```

Gambar 4.27 Deskripsi fungsi pusat

Berikut ini adalah penjelasan dari fungsi pusat.

1. Baris 7 s.d. 14 adalah pengambilan data geometri (berupa kumpulan koordinat penyusun geometri) dari tabel geometri.
2. Baris 21 s.d. 23 adalah proses pemeriksaan tipe geometri dari objek geometri yang dioperasikan. Apabila objek geometri bukan Poligon(5), maka fungsi mengembalikan NULL, jika tidak proses berlanjut.
3 Baris 26 s.d. 36 adalah proses looping untuk penggabungan data geometri menjadi bentuk string tunggal. Looping akan berhenti jika tidak ditemui lagi record lain dengan id yang sama.
4 Baris 38 s.d. 43 adalah penyempurnaan isi string tunggal sebelum dikembalikan kepada pengguna.

5 Baris 44 adalah melakukan pengoperan string tunggal sebagai parameter kepada fungsi udf_pusat. Hasil kembalian dari udf_pusat akan dibentuk menjadi format WKT yang sesusai.

Deskripsi udf_pusat:

```
 1  .
 2
 3  .
 4
 5  for(j=0;j<i-1;j++){
 6        res1 += (dxy[j].x + dxy[j+1].x) * (dxy[j].x *
 7  dxy[j+1].y - dxy[j+1].x * dxy[j].y);
 8        res2 += (dxy[j].y + dxy[j+1].y) * (dxy[j].x *
 9  dxy[j+1].y - dxy[j+1].x * dxy[j].y);
10  }
11
12  cx = res1/(6*luas);
13  cy = res2/(6*luas);
14
15  *length = sprintf(result,"%f %f",cx,cy);
16
17  return result;
```

Gambar 4.28 Deskripsi fungsi udf_pusat

Berikut ini adalah penjelasan dari fungsi udf_pusat.

1. Baris 5 s.d. 13 adalah proses penghitungan titik pusat geometri. Cx menampung absis dari titik pusat, sedangkan cy menampung ordinat dari titik pusat.
2. Baris 15 adalah proses penggabung cx dan cy menjadi bentuk string dan disimpan ke variable result untuk dikembalikan ke fungsi pusat.
3 Baris 17 adalah proses pengembalian nilai result ke fungsi pusat.

4.5.2.8 Implementasi Operator Bersilangan

Operator bersilangan membutuhkan 2 operator berupa objek geometri. Pada API ini, operator bersilangan dapat digunakan untuk pasangan objek geometri Garis-Garis. Operator ini akan mengembalikan nilai 1 bila hasilnya benar, dan 0 jika hasilnya salah.

Seperti operator sebelumnya, operator bersilangan diimplementasikan menggunakan 2 fungsi, yaitu fungsi **bersilangan** yang dibuat menggunakan PL/SQL, dan fungsi udf_bersilangan yang dibuat menggunakan bahasa C.

Deskripsi bersilangan:

```
 1  DECLARE cur_geom CURSOR FOR SELECT
 2  CONCAT(IFNULL(x0,'NULL'),' ',IFNULL(y0,'NULL')
 3  ,',',IFNULL(x1,'NULL'),'
 4  ',IFNULL(y1,'NULL'),',',IFNULL(x2,'NULL'),'
 5  ',IFNULL(y2,'NULL'),',',IFNULL(x3,'NULL'),'
 6  ',IFNULL(y3,'NULL'),',',IFNULL(x4,'NULL'),'
 7  ',IFNULL(y4,'NULL'))
 8  FROM geometri WHERE gid = gidIn;
 9
10  DECLARE cur_geom2 CURSOR FOR SELECT
11  CONCAT(IFNULL(x0,'NULL'),' ',IFNULL(y0,'NULL')
12  ,',',IFNULL(x1,'NULL'),'
13  ',IFNULL(y1,'NULL'),',',IFNULL(x2,'NULL'),'
14  ',IFNULL(y2,'NULL'),',',IFNULL(x3,'NULL'),'
15  ',IFNULL(y3,'NULL'),',',IFNULL(x4,'NULL'),'
16  ',IFNULL(y4,'NULL'))
17  FROM geometri WHERE gid = gidIn2;
18
19  DECLARE CONTINUE HANDLER FOR NOT FOUND SET last_fetched=1;
20
21  SET last_fetched = 0;
22  SET fintext = '';
23  SET fintext2 = '';
24
25  SELECT etype INTO m_etype FROM geometri WHERE gid = gidIn;
26  SELECT etype INTO m_etype2 FROM geometri WHERE gid =gidIn2;
27  IF (m_etype != m_etype2 && m_etype != 3) THEN
28        RETURN NULL;
29  ELSE
30
31  OPEN cur_geom;
32
33  cursor_loop:LOOP
34        FETCH cur_geom INTO tmptext;
35        SET fintext = CONCAT(fintext,tmptext);
36        IF last_fetched = 1 THEN
37              LEAVE cursor_loop;
38        END IF;
39  END LOOP;
40
41  CLOSE cur_geom;
42
43  SET last_fetched = 0;
44
45  OPEN cur_geom2;
46
47  cursor_loop:LOOP
48        FETCH cur_geom2 INTO tmptext;
49        SET fintext2 = CONCAT(fintext2,tmptext);
50        IF last_fetched = 1 THEN
```

```
51              LEAVE cursor_loop;
52          END IF;
53  END LOOP;
54
55  CLOSE cur_geom2;
56
57  IF INSTR(fintext,'NULL') > 0 THEN
58  SET rettext = SUBSTR(fintext,2,INSTR(fintext,'NULL')-3);
59  ELSE
60  SET rettext = SUBSTR(fintext,2,LENGTH(fintext));
61  END IF;
62
63  IF INSTR(fintext2,'NULL') > 0 THEN
64  SET rettext2 = SUBSTR(fintext2,2,INSTR(fintext2,'NULL')-3);
65  ELSE
66  SET rettext2 = SUBSTR(fintext2,2,LENGTH(fintext2));
67  END IF;
68
69  RETURN udf_bersilangan(rettext,rettext2);
70  END IF;
```

Gambar 4.29 Deskripsi fungsi bersilangan

Berikut ini adalah penjelasan dari fungsi bersilangan.

1. Baris 1 s.d. 17 adalah proses pengambilan data geometri (berupa kumpulan koordinat penyusun geometri) dari tabel geometri untuk kedua geometri yang ingin diperiksa.
2 Baris 25 s.d.28 adalah pemeriksaan tipe geometri, untuk memastikan yang dioperasikan adalah tipe geometri dengan pasangan objek Garis-Garis.
3. Baris 31 s.d. 53 adalah proses proses looping untuk penggabungan data geometri menjadi bentuk string tunggal. Looping akan berhenti jika tidak ditemui lagi record lain dengan id yang sama.
4 Baris 57 s.d. 67 adalah penyempurnaan isi string tunggal untuk masing-masing objek geometri sebelum dioper kepada fungsi udf_bersilangan.
5 Baris 69 adalah proses pengoperan 2 buah variable string hasil gabungan koordinat penyusun geometri ke fungsi udf_bersilangan, kemudian mengembalikan hasilnya langsung ke pengguna.

```
Deskripsi udf_bersilangan:
```

```
 1  .
 2  .
 3  .
 4  for (k=0; k<i1-1; k++){.
 5  for (j=0; j<i2-1; j++){.
 6
 7  numera=(dxy2[j+1]->x - dxy2[j]->x)(dxy1[k]->y - dxy2[j]->y)
 8  – (dxy2[j+1]->y - dxy2[j]->y)(dxy1[k]->x - dxy2[j]->x)
 9
10  numerb=(dxy1[k+1]->x – dxy1[k]->x)(dxy1[k]->y - dxy2[j]->y)
11  – (dxy1[k+1]->y – dxy1[k]->y)(dxy1[k]->x - dxy2[j]->x)
12
13  denom=(dxy2[j+1]->y - dxy2[j]->y)(dxy1[k+1]->x – dxy1[k]-
14  >x) – (dxy2[j+1]->x - dxy2[j]->x)(dxy1[k+1]->y – dxy1[k]-
15  >y)
16
17  mua = numera/denom;
18  mub = numerb/denom;
19
20  if((abs(numera)<EPS && abs(numerb)<EPS && abs(denom)<EPS)
21  || ((!abs(denom)<EPS) && (!mua<0 || !mua>1 || !mub<0 ||
22  !mub>1)))
23  return 1;
24
25  }
26  }
27
```

Gambar 4.30 Deskripsi fungsi udf_bersilangan

Berikut ini adalah penjelasan dari fungsi udf_ bersilangan.

1. Sebelum dihitung, dibentuk terlebih dahulu array struct dxy1 dan dxy2 untuk merepresentasikan kumpulan pasangan koordinat yang membangun masing-masing geometri.
2. Baris 4 s.d. 23 adalah proses pemeriksaan apakah tiap segmen dari geometri Garis ada yang berpotongan atau tidak. Syaratnya adalah kedua garis tidak coincident, tidak parallel, dan intersection tidak di luar segmen garis.

3 Baris 37 s.d. 40 adalah proses looping untuk menghitung panjang. Looping digunakan dalam upaya mencari panjang dari setiap segmen yang ada pada Garis.

4 Jika baris 20 s.d 22 bernilai true, maka kedua garis dinyatakan berpotongan, dan fungsi mengembalikan nilai 1.

5 Jika tidak, maka perulangan dilakukan kembali hingga seluruh kombinasi segment selesai diperiksa.

4.5.2.9 Implementasi Operator Berpotongan

Operator bersilangan membutuhkan 2 operator berupa objek geometri. Pada API ini, operator berpotongan dapat digunakan untuk pasangan objek geometri Poligon-Poligon. Operator ini akan mengembalikan nilai 1 bila hasilnya benar, dan 0 jika hasilnya salah.

Seperti operator sebelumnya, operator bersilangan diimplementasikan menggunakan 2 fungsi, yaitu fungsi **berpotongan** yang dibuat menggunakan PL/SQL, dan fungsi **udf_ berpotongan** yang dibuat menggunakan bahasa C.

Deskripsi berpotongan:

```
 1  DECLARE cur_geom CURSOR FOR SELECT
 2  CONCAT(IFNULL(x0,'NULL'),' ',IFNULL(y0,'NULL')
 3  ,',',IFNULL(x1,'NULL'),'
 4  ',IFNULL(y1,'NULL'),',',IFNULL(x2,'NULL'),'
 5  ',IFNULL(y2,'NULL'),',',IFNULL(x3,'NULL'),'
 6  ',IFNULL(y3,'NULL'),',',IFNULL(x4,'NULL'),'
 7  ',IFNULL(y4,'NULL'))
 8  FROM geometri WHERE gid = gidIn;
 9
10  DECLARE cur_geom2 CURSOR FOR SELECT
11  CONCAT(IFNULL(x0,'NULL'),' ',IFNULL(y0,'NULL')
12  ,',',IFNULL(x1,'NULL'),'
13  ',IFNULL(y1,'NULL'),',',IFNULL(x2,'NULL'),'
14  ',IFNULL(y2,'NULL'),',',IFNULL(x3,'NULL'),'
15  ',IFNULL(y3,'NULL'),',',IFNULL(x4,'NULL'),'
16  ',IFNULL(y4,'NULL'))
17  FROM geometri WHERE gid = gidIn2;
18
19  DECLARE CONTINUE HANDLER FOR NOT FOUND SET last_fetched=1;
20
21  SET last_fetched = 0;
22  SET fintext = '';
23  SET fintext2 = '';
24
25  SELECT etype INTO m_etype FROM geometri WHERE gid = gidIn;
26  SELECT etype INTO m_etype2 FROM geometri WHERE gid =gidIn2;
27  IF (m_etype != m_etype2 && m_etype != 5) THEN
28        RETURN NULL;
29  ELSE
30
31  OPEN cur_geom;
32
33  cursor_loop:LOOP
34        FETCH cur_geom INTO tmptext;
35        SET fintext = CONCAT(fintext,tmptext);
36        IF last_fetched = 1 THEN
37              LEAVE cursor_loop;
38        END IF;
39  END LOOP;
```

```
40
41  CLOSE cur_geom;
42
43  SET last_fetched = 0;
44
45  OPEN cur_geom2;
46
47  cursor_loop:LOOP
48        FETCH cur_geom2 INTO tmptext;
49        SET fintext2 = CONCAT(fintext2,tmptext);
50        IF last_fetched = 1 THEN
51              LEAVE cursor_loop;
52        END IF;
53  END LOOP;
54
55  CLOSE cur_geom2;
56
57  IF INSTR(fintext,'NULL') > 0 THEN
58  SET rettext = SUBSTR(fintext,2,INSTR(fintext,'NULL')-3);
59  ELSE
60  SET rettext = SUBSTR(fintext,2,LENGTH(fintext));
61  END IF;
62
63  IF INSTR(fintext2,'NULL') > 0 THEN
64  SET rettext2 = SUBSTR(fintext2,2,INSTR(fintext2,'NULL')-3);
65  ELSE
66  SET rettext2 = SUBSTR(fintext2,2,LENGTH(fintext2));
67  END IF;
68
69  RETURN udf_berpotongan(rettext,rettext2);
70  END IF;
```

Gambar 4.31 Deskripsi fungsi berpotongan

Berikut ini adalah penjelasan dari fungsi berpotongan.

1. Baris 1 s.d. 17 adalah proses pengambilan data geometri (berupa kumpulan koordinat penyusun geometri) dari tabel geometri untuk kedua geometri yang ingin diperiksa.
2 Baris 25 s.d.28 adalah pemeriksaan tipe geometri, untuk memastikan yang dioperasikan adalah tipe geometri dengan pasangan objek Poligon-Poligon.
3. Baris 31 s.d. 53 adalah proses proses looping untuk penggabungan data geometri menjadi bentuk string tunggal. Looping akan berhenti jika tidak ditemui lagi record lain dengan id yang sama.
4 Baris 57 s.d. 67 adalah penyempurnaan isi string tunggal untuk masing-masing objek geometri sebelum dioper kepada fungsi udf_bersilangan.

5 Baris 69 adalah proses pengoperan 2 buah variable string hasil gabungan koordinat penyusun geometri ke fungsi **udf_berpotongan**, kemudian mengembalikan hasilnya langsung ke pengguna.

Deskripsi udf_berpotongan:

```
 1  .
 2  .
 3  .
 4  for (k=0; k<i1-1; k++){.
 5  for (j=0; j<i2-1; j++){.
 6
 7  numera=(dxy2[j+1]->x - dxy2[j]->x)(dxy1[k]->y - dxy2[j]->y)
 8  – (dxy2[j+1]->y - dxy2[j]->y)(dxy1[k]->x - dxy2[j]->x)
 9
10  numerb=(dxy1[k+1]->x – dxy1[k]->x)(dxy1[k]->y - dxy2[j]->y)
11  – (dxy1[k+1]->y – dxy1[k]->y)(dxy1[k]->x - dxy2[j]->x)
12
13  denom=(dxy2[j+1]->y - dxy2[j]->y)(dxy1[k+1]->x – dxy1[k]-
14  >x) – (dxy2[j+1]->x - dxy2[j]->x)(dxy1[k+1]->y – dxy1[k]-
15  >y)
16
17  mua = numera/denom;
18  mub = numerb/denom;
19
20  if((abs(numera)<EPS && abs(numerb)<EPS && abs(denom)<EPS)
21  || ((!abs(denom)<EPS) && (!mua<0 || !mua>1 || !mub<0 ||
22  !mub>1)))
23  return 1;
24
25  }
26  }
```

Gambar 4.32 Deskripsi fungsi udf_berpotongan

Berikut ini adalah penjelasan dari fungsi udf_ berpotongan.

1. Sebelum dihitung, dibentuk terlebih dahulu array struct dxy1 dan dxy2 untuk merepresentasikan kumpulan pasangan koordinat yang membangun masing-masing Poligon.
2. Baris 4 s.d. 23 adalah proses pemeriksaan apakah tiap segmen dari geometri Garis ada yang berpotongan atau tidak. Syaratnya adalah kedua garis tidak coincident, tidak parallel, dan intersection tidak di luar segmen garis pembentuk Poligon.

3 Baris 37 s.d. 40 adalah proses looping untuk menghitung panjang. Looping digunakan dalam upaya mencari panjang dari setiap segmen yang ada pada Poligon.

4 Jika baris 20 s.d 22 bernilai true, maka kedua garis dinyatakan berpotongan, dan fungsi mengembalikan nilai 1.

5 Jika tidak, maka perulangan dilakukan kembali hingga seluruh kombinasi segment selesai diperiksa.

4.5.2.10 Implementasi Operator Jarak

Operator jarak membutuhkan 2 operator berupa objek geometri. Pada API ini, operator jarak dapat digunakan untuk pasangan objek geometri Poligon-Poligon atau Titik-Poligon.

Seperti operator sebelumnya, operator jarak diimplementasikan menggunakan 2 fungsi, yaitu fungsi **jarak** yang dibuat menggunakan PL/SQL, dan fungsi **udf_ jarak** yang dibuat menggunakan bahasa C.

Deskripsi jarak:

```
 1     DECLARE cur_geom CURSOR FOR SELECT
 2     CONCAT(IFNULL(x0,'NULL'),' ',IFNULL(y0,'NULL')
 3     ,',',IFNULL(x1,'NULL'),'
 4     ',IFNULL(y1,'NULL'),',',IFNULL(x2,'NULL'),'
 5     ',IFNULL(y2,'NULL'),',',IFNULL(x3,'NULL'),'
 6     ',IFNULL(y3,'NULL'),',',IFNULL(x4,'NULL'),'
 7     ',IFNULL(y4,'NULL'))
 8     FROM geometri WHERE gid = gidIn;
 9
10     DECLARE cur_geom2 CURSOR FOR SELECT
11     CONCAT(IFNULL(x0,'NULL'),' ',IFNULL(y0,'NULL')
12     ,',',IFNULL(x1,'NULL'),'
13     ',IFNULL(y1,'NULL'),',',IFNULL(x2,'NULL'),'
14     ',IFNULL(y2,'NULL'),',',IFNULL(x3,'NULL'),'
15     ',IFNULL(y3,'NULL'),',',IFNULL(x4,'NULL'),'
16     ',IFNULL(y4,'NULL'))
17     FROM geometri WHERE gid = gidIn2;
18
19     DECLARE CONTINUE HANDLER FOR NOT FOUND SET last_fetched=1
20
21     SET last_fetched = 0;
22     SET fintext = ‘’;
23     SET fintext2 = ‘’;
24
25     SELECT etype INTO m_etype FROM geometri WHERE gid = gidIn
26     SELECT etype INTO m_etype2 FROM geometri WHERE gid =gidIn
27     IF (m_etype != 1 || m_etype != 5) && (m_etype != 5 ||
28     m_etype != 1) THEN
```

```
29          RETURN NULL;
30      ELSE
31
32      OPEN cur_geom;
33
34      cursor_loop:LOOP
35          FETCH cur_geom INTO tmptext;
36          SET fintext = CONCAT(fintext,tmptext);
37          IF last_fetched = 1 THEN
38              LEAVE cursor_loop;
39          END IF;
40      END LOOP;
41
42      CLOSE cur_geom;
43
44      SET last_fetched = 0;
45
46      OPEN cur_geom2;
47
48      cursor_loop:LOOP
49          FETCH cur_geom2 INTO tmptext;
50          SET fintext2 = CONCAT(fintext2,tmptext);
51          IF last_fetched = 1 THEN
52              LEAVE cursor_loop;
53          END IF;
54      END LOOP;
55
56      CLOSE cur_geom2;
57
58      IF INSTR(fintext,'NULL') > 0 THEN
59      SET rettext = SUBSTR(fintext,2,INSTR(fintext,'NULL')-3);
60      ELSE
61      SET rettext = SUBSTR(fintext,2,LENGTH(fintext));
62      END IF;
63
64      IF INSTR(fintext2,'NULL') > 0 THEN
65      SET rettext2 = SUBSTR(fintext2,2,INSTR(fintext2,'NULL')-3);
66      ELSE
67      SET rettext2 = SUBSTR(fintext2,2,LENGTH(fintext2));
68      END IF;
69
70      RETURN udf_jarak(rettext,m_etype,rettext2, m_etype2);
71      END IF;
```

Gambar 4.33 Deskripsi fungsi jarak

Berikut ini adalah penjelasan dari fungsi jarak.

1. Baris 1 s.d. 17 adalah proses pengambilan data geometri (berupa kumpulan koordinat penyusun geometri) dari tabel geometri untuk kedua geometri yang ingin dihitung jaraknya.
2 Baris 25 s.d.29 adalah pemeriksaan tipe geometri, untuk memastikan yang dioperasikan adalah tipe geometri dengan pasangan objek Poligon-Poligon atau Titik-Titik atau Poligon-Titik

3. Baris 32 s.d. 56 adalah proses proses looping untuk penggabungan data geometri menjadi bentuk string tunggal. Looping akan berhenti jika tidak ditemui lagi record lain dengan id yang sama.
4 Baris 58 s.d. 68 adalah penyempurnaan isi string tunggal untuk masing-masing objek geometri sebelum dioper kepada fungsi udf_bersilangan.
5 Baris 70 adalah proses pengoperan 2 buah variable string hasil gabungan koordinat penyusun geometri ke fungsi **udf_jarak**, kemudian mengembalikan hasilnya langsung ke pengguna.

Deskripsi udf_jarak:

```
 1  .
 2  .
 3  .
 4  if (etype == 1){
 5        xy1->x = dxy1[0]->x;
 6        xy1->y = dxy1[0]->y;
 7  }else{
 8        for(j=0;j<i-1;j++){
 9        luas += dxy1[j].x * dxy1[j+1].y – dxy1[j+1].x *
10  dxy1[j].y;
11  }
12  luas = abs(luas/2);
13
14        for(j=0;j<i-1;j++){
15        res1 += (dxy1[j].x + dxy1[j+1].x) * (dxy1[j].x *
16  dxy1[j+1].y – dxy1[j+1].x * dxy1[j].y);
17        res2 += (dxy1[j].y + dxy1[j+1].y) * (dxy1[j].x *
18  dxy1[j+1].y – dxy1[j+1].x * dxy1[j].y);
19  }
20
21  xy1->x = res1/(6*luas);
22  xy1->y = res2/(6*luas);
23  }
24
25  if (etype2 == 1){
26        xy2->x = dxy2[0]->x;
27        xy2->y = dxy2[0]->y;
28  }else{
29        for(j=0;j<i-1;j++){
30        luas += dxy2[j].x * dxy2[j+1].y – dxy2[j+1].x *
31  dxy2[j].y;
32  }
33  luas = abs(luas/2);
34
35        for(j=0;j<i-1;j++){
36        res1 += (dxy2[j].x + dxy2[j+1].x) * (dxy2[j].x *
37  dxy2[j+1].y – dxy2[j+1].x * dxy2[j].y);
```

```
38        res2 += (dxy2[j].y + dxy2[j+1].y) * (dxy2[j].x *
39  dxy2[j+1].y – dxy2[j+1].x * dxy2[j].y);
40  }
41
42  xy2->x = res1/(6*luas);
43  xy2->y = res2/(6*luas);
44  }
45
46  jarak = sqrt(pow(xy2->x – xy1->x,2)+ pow(xy2->y – xy1-
47  >y,2));
48  return jarak;
```

Gambar 4.34 Deskripsi fungsi udf_jarak

Berikut ini adalah penjelasan dari fungsi udf_ jarak.

1. Baris 4 s.d. 23 adalah proses mencari titik tengah untuk objek geometri pertama. Jika objek geometri pertama adalah titik, maka koordinat titik tersebut adalah titik tengah, namun jika polygon, dicari terlebih dahulu titik tengahnya.
2. Baris 25 s.d. 44 adalah proses mencari titik tengah untuk objek geometri kedua. Jika objek geometri kedua adalah titik, maka koordinat titik tersebut adalah titik tengah, namun jika polygon, dicari terlebih dahulu titik tengahnya.
3. Baris 46 adalah proses pencarian jarak setelah diketahui titik tengah masing-masing objek geometri.
4. Baris 48 adalah pengembalian hasil penghitungan jarak ke fungsi pemanggil.

4.5.2.11 Implementasi Operator Perpotongan

Operator perpotongan digunakan untuk mencari daerah perpotongan antara 2 Poligon. Oleh karena itu pasangan objek geometri yang diperbolehkan hanya Poligon-Poligon. Operator ini diimplementasikan dalam 2 bentuk, yaitu fungsi **perpotongan** yang dibangun dengan menggunakan bahasa PL/SQL, dan juga udf_perpotongan yang dibangun dengan menggunakan bahasa C.

```
Deskripsi perpotongan:
```

```
 1  DECLARE cur_geom CURSOR FOR SELECT
 2  CONCAT(IFNULL(x0,'NULL'),' ',IFNULL(y0,'NULL')
 3  ,',',IFNULL(x1,'NULL'),'
 4  ',IFNULL(y1,'NULL'),',',IFNULL(x2,'NULL'),'
 5  ',IFNULL(y2,'NULL'),',',IFNULL(x3,'NULL'),'
 6  ',IFNULL(y3,'NULL'),',',IFNULL(x4,'NULL'),'
 7  ',IFNULL(y4,'NULL'))
 8  FROM geometri WHERE gid = gidIn;
 9
10  DECLARE cur_geom2 CURSOR FOR SELECT
11  CONCAT(IFNULL(x0,'NULL'),' ',IFNULL(y0,'NULL')
12  ,',',IFNULL(x1,'NULL'),'
13  ',IFNULL(y1,'NULL'),',',IFNULL(x2,'NULL'),'
14  ',IFNULL(y2,'NULL'),',',IFNULL(x3,'NULL'),'
15  ',IFNULL(y3,'NULL'),',',IFNULL(x4,'NULL'),'
16  ',IFNULL(y4,'NULL'))
17  FROM geometri WHERE gid = gidIn2;
18
19  DECLARE CONTINUE HANDLER FOR NOT FOUND SET last_fetched=1;
20
21  SET last_fetched = 0;
22  SET fintext = ‘’;
23  SET fintext2 = ‘’;
24
25  SELECT etype INTO m_etype FROM geometri WHERE gid = gidIn;
26  SELECT etype INTO m_etype2 FROM geometri WHERE gid =gidIn2;
27  IF (m_etype != m_etype2 && m_etype != 5) THEN
28        RETURN NULL;
29  ELSE
30
31  OPEN cur_geom;
32
33  cursor_loop:LOOP
34        FETCH cur_geom INTO tmptext;
35        SET fintext = CONCAT(fintext,tmptext);
36        IF last_fetched = 1 THEN
37              LEAVE cursor_loop;
38        END IF;
39  END LOOP;
40
41  CLOSE cur_geom;
42
43  SET last_fetched = 0;
44
45  OPEN cur_geom2;
46
47  cursor_loop:LOOP
48        FETCH cur_geom2 INTO tmptext;
49        SET fintext2 = CONCAT(fintext2,tmptext);
50        IF last_fetched = 1 THEN
51              LEAVE cursor_loop;
52        END IF;
53  END LOOP;
54
55  CLOSE cur_geom2;
56
57  IF INSTR(fintext,'NULL') > 0 THEN
58  SET rettext = SUBSTR(fintext,2,INSTR(fintext,'NULL')-3);
59  ELSE
```

```
60  SET rettext = SUBSTR(fintext,2,LENGTH(fintext));
61  END IF;
62
63  IF INSTR(fintext2,'NULL') > 0 THEN
64  SET rettext2 = SUBSTR(fintext2,2,INSTR(fintext2,'NULL')-3);
65  ELSE
66  SET rettext2 = SUBSTR(fintext2,2,LENGTH(fintext2));
67  END IF;
68
69  RETURN udf_perpotongan(rettext,rettext2);
70  END IF;
```

Gambar 4.35 Deskripsi fungsi perpotongan

Berikut ini adalah penjelasan dari fungsi perpotongan.

1. Baris 1 s.d. 17 adalah proses pengambilan data geometri (berupa kumpulan koordinat penyusun geometri) dari tabel geometri untuk kedua geometri yang ingin diperiksa.
2 Baris 25 s.d.28 adalah pemeriksaan tipe geometri, untuk memastikan yang dioperasikan adalah tipe geometri dengan pasangan objek Poligon-Poligon.
3. Baris 31 s.d. 53 adalah proses proses looping untuk penggabungan data geometri menjadi bentuk string tunggal. Looping akan berhenti jika tidak ditemui lagi record lain dengan id yang sama.
4 Baris 57 s.d. 67 adalah penyempurnaan isi string tunggal untuk masing-masing objek geometri sebelum dioper kepada fungsi udf_bersilangan.
5 Baris 69 adalah proses pengoperan 2 buah variable string hasil gabungan koordinat penyusun geometri ke fungsi **udf_perpotongan**, kemudian mengembalikan hasilnya langsung ke pengguna.

Deskripsi udf_perpotongan:

```
1  .
2  .
3  .
4  for (k=0; k<i1-1; k++){.
5  for (j=0; j<i2-1; j++){.
6
7  numera=(dxy2[j+1]->x - dxy2[j]->x)(dxy1[k]->y - dxy2[j]->y)
8  – (dxy2[j+1]->y - dxy2[j]->y)(dxy1[k]->x - dxy2[j]->x)
9
```

```
10  numerb=(dxy1[k+1]->x – dxy1[k]->x)(dxy1[k]->y - dxy2[j]->y)
11  – (dxy1[k+1]->y – dxy1[k]->y)(dxy1[k]->x - dxy2[j]->x)
12
13  denom=(dxy2[j+1]->y - dxy2[j]->y)(dxy1[k+1]->x – dxy1[k]-
14  >x) – (dxy2[j+1]->x - dxy2[j]->x)(dxy1[k+1]->y – dxy1[k]-
15  >y)
16
17  mua = numera/denom;
18  mub = numerb/denom;
19
20  if((abs(numera)<EPS && abs(numerb)<EPS && abs(denom)<EPS)
21  || ((!abs(denom)<EPS) && (!mua<0 || !mua>1 || !mub<0 ||
22  !mub>1))){
23
24        dxyres[n]->x = dxy1[k]->x + mua * dxy1[k+1]->x –
25  dxy1[k]->x;
26        dxyres[n]->y = dxy1[k]->y + mub * dxy1[k+1]->y –
27  dxy1[k]->y;
28  }
29  }
30  return to_text(dxyres);
31  }
```

Gambar 4.36 Deskripsi fungsi udf_perpotongan

Berikut ini adalah penjelasan dari fungsi udf_ perpotongan.

1. Sebelum dihitung, dibentuk terlebih dahulu array struct dxy1 dan dxy2 untuk merepresentasikan kumpulan pasangan koordinat yang membangun masing-masing Poligon.
2. Baris 4 s.d. 23 adalah proses pemeriksaan apakah tiap segmen dari geometri Garis ada yang berpotongan atau tidak. Syaratnya adalah kedua garis tidak coincident, tidak parallel, dan intersection tidak di luar segmen garis pembentuk Poligon.
3 Baris 37 s.d. 40 adalah proses looping untuk menghitung panjang. Looping digunakan dalam upaya mencari panjang dari setiap segmen yang ada pada Poligon.
4 Jika baris 20 s.d 22 bernilai true, maka kedua garis dinyatakan berpotongan. Jika demikian, maka dilakukan kalkulasi untuk mencari nilai absis dan ordinat titik perpotongan dan dimasukkan ke dalam variable struct dxyres (Baris 24 s.d 27)..
5 Perulangan dilakukan kembali hingga seluruh kombinasi segment selesai diperiksa.

6 Baris 30 adalah proses membentuk isi array struct dxyres ke dalam bentuk string. Kemudian hasil string dikembalikan ke fungsi yang memanggi.

BAB V
PENGUJIAN

5.1 Pengujian Skema Basis Data Spasial

5.1.1 Pengujian Pembuatan Kolom Geometri

Pengujian ini bertujuan untuk mengetahui bahwa dapat dilakukan operasi create table spasial, dengan kolom data spasial. Parameter keberhasilan pengujian ini adalah tabel spasial dapat terbentuk , terwujud hubungan referensi dari tabel feature ke tabel geometri, dan juga tercatat metadata tentang tabel feature pada kolom geometry_columns.

Query yang diberikan :

No	Kasus Uji	Hasil
1	“CREATE TABLE jalan(id_jalan mediumint(9) NOT NULL, nama_jalan varchar(20) NOT NULL, lokasi int(10) unsigned NOT NULL, PRIMARY KEY (id_jalan);” “CALL tambah_kolom_geometri('jalan', jalan_geom', 'GARIS', 4326, 1);”	Berhasil
2	“CREATE TABLE gedung(id_gedung mediumint(9) NOT NULL, nama_gedung varchar(20) NOT NULL, lokasi int(10) unsigned NOT NULL, PRIMARY KEY (id_gedung);” “CALL tambah_kolom_geometri('gedung', 'gedung_geom', 'POLIGON', 4326, 2);”	Berhasil
3	“CREATE TABLE hydrant(id_hydrant mediumint(9) NOT NULL, kode_hydrant varchar(20) NOT NULL,	Berhasil

	hydrant_geom int(10) unsigned NOT NULL, PRIMARY KEY (id_gedung);” “CALL tambah_kolom_geometri('hydrant', 'hydrant_geom', 'TITIK', 4326, 0);”	

Untuk uji pertama, hasilnya terbentuk tabel bernama jalan, memiliki referensi ke tabel geometri dari kolom lokasi. Pada Tabel geometry_columns, masuk entry baru sebagai berikut :

Field	Values
f_table_name	jalan
f_geometry_columns	geom._jalan
g_table_name	geometri
Geometry_type	3
Coord_dimension	2
Srid	4326

5.1.2 Pengujian Penghapusan Kolom Geometri

Pengujian ini bertujuan untuk mengetahui bahwa dapat dilakukan operasi hapus kolom spasial. Parameter keberhasilan pengujian ini adalah kolom spasial dapat terhapus dan hubungan referensi dari tabel feature ke tabel geometri ikut terhapus pula.

Query yang diberikan :

No	Kasus Uji	Hasil
1	“CALL hapus_kolom_geometri('jalan', ‘jalan_geom');”	Berhasil
2	“CALL hapus_kolom_geometri('gedung', 'gedung_geom');”	Berhasil
3	“CALL hapus_kolom_geometri('hydrant', 'hydrant_geom');”	Berhasil

5.2 Pengujian Insert dan Select Basis Data Spasial

5.2.1 Pengujian Insert Data Spasial

Pengujian ini bertujuan untuk mengetahui bahwa dapat dilakukan operasi insert data spasial pada tabel spasial menggunakan tipe data Titik, Garis, dan Poligon.

No	Kasus Uji	Hasil
1	"INSERT INTO geometri VALUES (1,3,1,10,20,30,40,60,80,NULL,NULL,NULL, NULL)" "INSERT INTO jalan VALUES (1,'Jl. Hati-hati',1)"	Berhasil
2	"INSERT INTO geometri VALUES (2,3,1,20,10,10,60,NULL,NULL,NULL,NULL, NULL,NULL)" "INSERT INTO jalan VALUES (2,'Jl. Di Jalan',2)"	Berhasil
3	"INSERT INTO geometri VALUES (4,5,1,40,-20,70,-20,70,-50,40,-20)" "INSERT INTO gedung VALUES (1,'Gedung Pancasila',4)"	Berhasil
4	"INSERT INTO geometri VALUES (5,5,1,0,0,30,0,30,20,0, 20,0,0)" "INSERT INTO gedung VALUES (2,'Gedung Merdeka',5)"	Berhasil
5	"INSERT INTO geometri VALUES (6,1,1,50,60,NULL, NULL, NULL, NULL, NULL,NULL, NULL, NULL)" "INSERT INTO hydrant VALUES (1,'K113',6)"	Berhasil

5.2.2 Pengujian Select Data Spasial

Pengujian ini bertujuan untuk mengetahui bahwa dapat dilakukan operasi select data spasial pada tabel spasial. Parameter keberhasilan pengujian ini adalah hasil seleksi data spasial dapat diwujudkan dalam format WKT.

No	Kasus Uji
1	SELECT nama_jalan,to_text(jalan_geom)FROM jalan
2	SELECT nama_gedung, to_text(gedung_geom) FROM gedung
3	SELECT kode_hydrant, to_text(hydrant_geom) FROM hydrant

No	Hasil Uji
1	Jl. Hati-hati | GARIS(10 20,30 40,60 80) Jl. Di Jalan | GARIS(20 10,10 60)
2	Gedung Pancasila | POLIGON(40 -20,70 -20,70 -50,40 -20) Gedung Merdeka | POLIGON(0 0,30 0,30 20,0 20,0 0)
3	K113 | TITIK(50 60)

5.3 Pengujian Query Spasial

5.3.1 Pengujian Operator Dasar

5.3.1.1 Operator Panjang

Pengujian ini bertujuan untuk mengetahui bahwa dapat dilakukan operasi panjang() pada objek geometri. Parameter keberhasilan pengujian ini adalah hasil eksekusi operator menghasilkan nilai panjang garis atau keliling polygon. Sedangkan untuk titik, tidak menghasilkan apa-apa atau NULL.

No	Kasus Uji
1	SELECT nama_jalan, panjang(jalan_geom) from jalan
2	SELECT nama_gedung, panjang(gedung_geom) from gedung
3	SELECT kode_hydrant, panjang(hydrant_geom) from gedung

No	Hasil Uji
1	Jl. Hati-hati | 78.2842712474619 Jl. Di Jalan | 50.9901951359278
2	Gedung Pancasila | 240 Gedung Merdeka | 200
3	K113 | NULL

5.3.1.2 Operator Luas

Pengujian ini bertujuan untuk mengetahui bahwa dapat dilakukan operasi luas() pada objek geometri. Parameter keberhasilan pengujian ini adalah hasil eksekusi operator menghasilkan nilai luas pada polygon, sedangkan pada titik dan garis, tidak menghasilkan apa-apa atau NULL.

No	Kasus Uji
1	SELECT nama_jalan, luas(jalan_geom) from jalan

2	SELECT nama_gedung, luas(gedung_geom) from gedung
3	SELECT kode_hydrant, luas(hydrant_geom) from gedung

No	Hasil Uji
1	Jl. Hati-hati | NULL Jl. Di Jalan | NULL
2	Gedung Pancasila | 900 Gedung Merdeka | 600
3	K113 | NULL

5.3.1.3 Operator Batasan

Pengujian ini bertujuan untuk mengetahui bahwa dapat dilakukan operasi batasan() pada objek geometri. Parameter keberhasilan pengujian ini adalah hasil eksekusi operator menghasilkan nilai batasan polygon dalam format WKT, sedangkan pada titik dan garis, tidak menghasilkan apa-apa atau NULL.

No	Kasus Uji
1	SELECT nama_jalan, batasan(jalan_geom) from jalan
2	SELECT nama_gedung, batasan(gedung_geom) from gedung
3	SELECT kode_hydrant, batasan(hydrant_geom) from gedung

No	Hasil Uji
1	Jl. Hati-hati | NULL Jl. Di Jalan | NULL
2	Gedung Pancasila | GARIS(40 -20,70 -20,70 -50,40 -50,40 -20) Gedung Merdeka | GARIS(0 0,30 0,30 20,0 20,0 0)
3	K113 | NULL

5.3.1.4 Operator Pusat

Pengujian ini bertujuan untuk mengetahui bahwa dapat dilakukan operasi batasan() pada objek geometri. Parameter keberhasilan pengujian ini adalah hasil eksekusi operator menghasilkan nilai pusat polygon atau pusat titik dalam format WKT, sedangkan pada garis tidak menghasilkan apa-apa atau NULL.

No	Kasus Uji
1	SELECT nama_jalan, pusat(jalan_geom) from jalan
2	SELECT nama_gedung, pusat(gedung_geom) from gedung
3	SELECT kode_hydrant, pusat(hydrant_geom) from gedung

No	Hasil Uji
1	Jl. Hati-hati | NULL Jl. Di Jalan | NULL
2	Gedung Pancasila | TITIK (55 -35) Gedung Merdeka | TITIK(15 10)
3	K113 | NULL

5.3.2 Pengujian Operator Topologi

5.3.2.1 Operator Bersilangan

Pengujian ini bertujuan untuk mengetahui bahwa dapat dilakukan operasi bersilangan pada objek geometri Garis. Parameter keberhasilan pengujian adalah hasil eksekusi operator menghasilkan nilai 1 atau 0 untuk pasangan objek geometri Garis, sedangkan untuk pasangan objek geometri lain tidak menghasilkan apa-apa atau NULL.

No	Kasus Uji
1	SELECT bersilangan(SELECT jalan_geom from jalan WHERE id_jalan = 1, SELECT jalan_geom from jalan WHERE id_jalan = 2) as hasil

No	Hasil Uji
1	1

5.3.2.2 Operator Berpotongan

Pengujian ini bertujuan untuk mengetahui bahwa dapat dilakukan operasi berpotongan pada pasangan objek geometri Poligon. Parameter keberhasilan pengujian ini adalah hasil ekseskusi operator menghasilkan nilai 1 atau 0 untuk pasangan objek geometri Poligon, sedangkan untuk pasangan objek geometri lain tidak menghasilkan apa-apa atau NULL.

No	Kasus Uji
1	SELECT berpotongan(SELECT gedung_geom from gedung WHERE id_gedung = 1, SELECT taman_geom from

	tamanWHERE id_taman = 1) as hasil

No	Hasil Uji
1	1

5.3.3 Pengujian Operator Analisis Spasial

5.3.3.1 Operator Jarak

Pengujian ini bertujuan untuk mengetahui bahwa dapat dilakukan operasi jarak pada pasangan objek geometri Titik-Poligon, Poligon-Poligon, atau Titik-Poligon. Parameter keberhasilan pengujian ini adalah hasil eksekusi operator menghasilkan nilai tertentu (hasil kalkulasi jarak) untuk pasangan objek geometri Titik-Poligon, Poligon-Poligon, atau Titik-Poligon. Sedangkan untuk pasangan objek geometri lain, tidak menghasilkan apa-apa atau NULL.

No	Kasus Uji
1	SELECT jarak(SELECT gedung_geom FROM gedung WHERE id_gedung = 2, SELECT taman_geom FROM taman WHERE id_taman = 1)
2	SELECT jarak(SELECT gedung_geom FROM gedung WHERE id_gedung = 2, SELECT hydrant_geom FROM hydrant WHERE id_hydrant = 1)

No	Hasil Uji
1	16.74066904
2	63.72009102

5.3.3.2 Operator Perpotongan

Pengujian ini bertujuan untuk mengetahui bahwa dapat dilakukan operasi perpotongan pada pasangan objek geometri Poligon-Poligon. Parameter keberhasilan pengujian ini adalah hasil eksekusi operator menghasilkan area perpotongan dalam format WKT jika memang ada perpotongan, atau NULL jika tidak ada perpotongan untuk pasangan objek geometri Poligon-Poligon. Sedangkan untuk pasangan objek geometri lain, tidak menghasilkan apa-apa atau NULL.

No	Kasus Uji
1	SELECT perpotongan(SELECT gedung_geom FROM gedung WHERE id_gedung = 2, SELECT taman_geom FROM taman WHERE id_taman = 1)

No	Hasil Uji
1	POLIGON(20 0,20 10,30 10,30 0)

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut:

1. Setiap objek geometri/tipe data geometri (misal : Titik, Garis, Poligon) dapat disimpan dalam sebuah tabel tersendiri, yaitu tabel geometri. Tabel geometri ini berisi data koordinat penyusun objek geometri dan jenis objek geometri tersebut. Sedangkan objek *feature*, yang menggambarkan dunia nyata (misal : Kota, Jalan, Gedung) disimpan dalam tabel *feature*. Semua tabel *feature* akan mereferensi tabel geometri sesuai dengan tipe data geometri yang diperlukan.
2. Operator spasial yang disediakan untuk sistem basis data spasial adalah operator dasar, operator topologi, dan operator analisis spasial.
3. Operator spasial diimplementasikan dalam bentuk fungsi yang disusun dalam sebuah berkas *Dynamic Linked Library*, yang dibuat dalam bahasa C. File .dll tersebut kemudian dikaitkan pada MySQL sebagai plugin.
4. Agar operator spasial dapat dikenali dan dimanfaatkan oleh pengguna, maka harus dibuatkan *stored function* berdasarkan .dll yang telah dikaitkan.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan API ini antara lain :

1. Pada API ini, belum diimplementasikan penggunaan indeks. Nantinya, untuk eksekusi query yang lebih cepat, perlu dibuat implementasi indeks, misalkan menggunakan B-Tree.
2. API ini dapat dikembangkan dengan menambahkan operator spasial lain yang belum diimplementasikan.
3. API ini dapat dikembangkan dengan menambahkan fungsi untuk *mengeksport* data spasial ke format dokumen yang dapat diwujudkan menjadi gambar, misalnya KML.

DAFTAR PUSTAKA

[YEU-07] Yeung, Albert K.W., Hall G. Brent. 2007. *Spatial Database Systems: Design, Implementation, and Project Management.*Springer

[REG-02] Regaux Phillipe, Scholl Michel, Voisard Agnes. 2002. *Spatial Databases with Application to GIS*. Morgan Kaufmann.

[MAN-05] Manolopoulos, Yannis & Papadopoulos, A.N. & Vassilakopoulos M.G. 2005. *Spatial Databases : Technologies, Techniques, and Trends*. Idea Group.

[DUB-09] DuBois, Paul. 2009.*MySQL Developer's Library 4th Edition*. Addison-Wesley.

[CLE-93] Clementini, Eliseo et. al. 1993. *A Small Set of Formal Topological Relationship Suitable for End-User Interaction.*

[OGC-10] OGC. 2010. *OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 2*.

[ANO-11] Anonymous. 2011. *Module Bb, Dynamic Link Library – Dll.* http://www.tenouk.com/ModuleBB.html. Diakses tanggal 10 Desember 2011.

[ANO-12] Anonymous. 2012. *Application Programming Interface*. *http://en.wikipedia.org/wiki/Application_programming_interface*. Diakses 7 Januari 2012