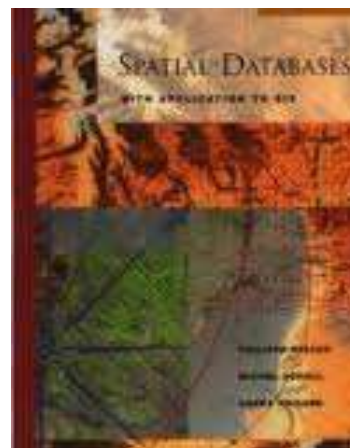
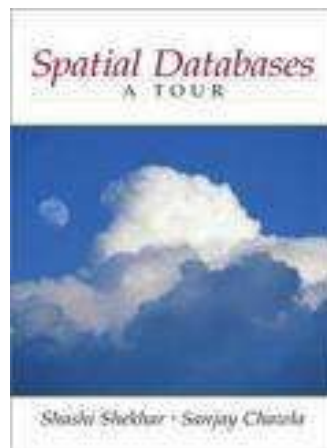


# Optimalisasi dan Pemrosesan Query



Database Spasial

# Topik

1. Evaluasi Operasi Spasial
2. Optimalisasi Query
3. Analisa Struktur Index Spasial
4. Sistem Basis Data Spasial Terdistribusi
5. Sistem Basis Data Spasial Parallel
6. Ringkasan

# Tujuan I

- Memahami konsep Optimalisasi dan Pemrosesan Query (QPO)
  - Apakah QPO itu?
  - Mengapa QPO dipelajari?

# Alasan

- Ukuran dan volume Record
- Kecepatan akses.
- Rancangan aplikasi perlu optimalisasi SQL
- Minimalkan pencarian trek
- Indeks untuk kecepatan akses data.

# Analogi Transmisi Otomatis pada Mobil

- Transmisi Manual : Otomatis  $\approx$  Java : SQL
- Program Java
  - Algoritma
  - Perubahan gigi manual pada mobil
- Query SQL
  - User tidak menentukan prosedur.
  - DBMS memilih algoritma
  - Analogi: transmisi otomatis memilih gigi
- Komponen SDBMS yang relevan
  - Optimalisasi Query (QPO)
    - Memilih algoritma
  - Model data fisik : QPO  $\approx$  mesin : transmisi otomatis

# Apakah Optimalisasi dan Pemrosesan Query (QPO) itu?

- Gagasan dasar QPO
  - SQL → query dengan format tingkat tinggi
  - QPO → ubah SQL jadi rencana eksekusi
    - Melalui model data fisik
    - Operasi pada struktur file, indeks, dll.
  - Rencana eksekusi untuk menjawab query
  - Biaya QPO kecil
    - Waktu komputasi untuk langkah-langkah QPO << rencana eksekusi

# Optimalisasi Query

- Optimalisasi Query adalah
  - prosedur
  - strategi evaluasi
  - evaluasi yang efektif.
- Tiga hal yang mempengaruhi optimalisasi query,
  - Ruang pencarian
  - Model biaya
  - Strategi pencarian

# Mengapa QPO Dipelajari?

- Mengapa transmisi otomatis pada mobil dipelajari?
  - Cari penyebab kekurangan tenaga pada mobil
    - Masalah mesin atau transmisi?
  - Pecahkan masalah kinerja dengan gigi manual
    - Kondisi menanjak, menurun → gigi rendah
- Mengapa QPO pada SDBMS dipelajari?
  - Tentukan *bottleneck* query
    - Masalah di model data fisik atau QPO ?
  - Cara membantu QPO mempercepat proses query ?
    - Menyediakan *hints*, menulis ulang query, dll.
  - Memperkokoh model data fisik untuk percepat queries?
    - Tambah indeks, mengubah struktur file, ...



# Tujuan Optimalisasi Query

- Waktu proses
- Waktu respon
- Akses termurah

# Kecepatan Akses Data

- Faktor lain yang mempengaruhi kecepatan akses data adalah:

## 1. Optimalisasi aplikasi

- Apakah akses datanya efisien?
- Bagaimana basis data dirancang?
  - Kadang-kadang normalisasi belum efisien.
  - Denormalisasi.
  - Contoh → tabel dengan relasi one-to-one yang sering diakses bisa jadi perlu digabung menjadi satu tabel.

## 2. Indeks

## 3. Pengklasteran

# Fakta

- Indeks tidak sesuai → tidak meningkatkan kecepatan akses.
- Contoh:
  - Tabel diindeks berdasarkan city, province, zip code dari tabel Employee
    - CREATE INDEX idx\_city\_prov\_zip\_code
    - ON employee(city, province, zip\_code)
    - TABLESPACE INDX;
  - User membuat query:
    - SELECT \* FROM employee WHERE province='West Java';
  - → indeks tidak dipakai karena kolom pertama (city) tidak ada dalam klausa WHERE
  - Jika user pernyataan ini sering digunakan → indeks harus diurutkan berdasarkan propinsi.

# Fakta

- Pencarian data lebih cepat → record berada di blok tabel yang sama/dekat.
- Contoh:
  - Pernyataan SQL ini:
    - `SELECT * FROM employee`
    - `WHERE id BETWEEN 1010 AND 2010;`
  - Query menscan lebih sedikit → data diindeks berdasarkan ID
  - Jika tidak diindeks berdasarkan ID, pilihannya → buat tabel lain yang diindeks berdasarkan atribut yang berbeda, seperti:
    - `CREATE TABLE indexed_employee`
    - `AS SELECT * FROM employee`
    - `ORDER BY id;`
  - SQL di atas berisi set data yang sama, tapi diindeks berdasarkan ID

# Tiga Konsep Kunci dalam QPO

- 1. *Building Block*
  - Mobil gerakannya sedikit, mis. Maju, mundur
  - DBMS punya sedikit *building block* :
    - select (point query, range query), join, sorting, ...
  - Query SQL dipecah ke dalam *building block*
- 2. Strategi pemrosesan Query *building block*
  - Mobil punya gigi untuk gerakan maju: 1, 2, 3, 4
  - DBMS menjaga strategi pemrosesan tiap *building block*.
    - Mis. Query titik bisa dijawab melalui indeks atau scan file data

# Tiga Konsep Kunci dalam QPO

- 3. Optimalisasi Query

- Transmisi otomatis → memilih gigi yang terbaik berdasarkan parameter gerakan.
- Pada *building block* dari query, QPO DBMS berusaha memilih:
  - Strategi “paling efisien” berdasarkan parameter basis data yang diberikan.
  - Contoh Parameter: ukuran tabel, indeks yang tersedia, ..
  - **Contoh** : pencarian indeks akan dipilih untuk sebuah query titik, jika indeksnya tersedia.

# Tantangan QPO

- Pilihan *building block*
  - Query SQL berdasarkan aljabar relasi(RA)
  - *Building block* RA adalah select, project, join
  - SQL3 menambah *building block* baru
- Pilihan strategi pemrosesan untuk *building block*
  - Hambatan: terlalu banyak strategi → lebih kompleks
  - DBMS komersial punya 10 sampai 30 strategi
    - 2 sampai 4 strategi tiap *building block*
- Bagaimana memilih strategi “terbaik” dari pilihan yang ada?
  - Gunakan skema prioritas
  - Atau gunakan model biaya sederhana berdasarkan parameter DBMS

# Operasi Dasar pada Aljabar Relasional

- Operasi Select

$\sigma \langle \text{selection condition} \rangle (R)$

- Operasi Project

$\pi \langle \text{List attribute} \rangle (R)$

- Operasi Join

$R \times S$  diikuti oleh  $\sigma \text{ PK=FK}$

$R \bowtie \langle \text{join condition} \rangle (S)$

$R * S$ , catatan: PK=FK

- Operasi Set

Union, Intersection, Difference



# Tantangan QPO dalam SDBMS

- *Building block* untuk query spasial
  - Kumpulan jenis data spasial dan operasi
  - Sedikitnya konsesus “*building block*”
  - Pilihan yang ada sekarang → spatial select, spatial join, nearest neighbor
- Pilihan strategi
  - Terbatasnya pilihan untuk beberapa *building block* mis. nearest neighbor
- Pemilihan strategi yang terbaik
  - Model biaya lebih rumit karena
    - Query spasial memakai CPU dan I/O dengan intensif
    - Sedangkan query non spasial hanya intensif pada I/O
  - Model biaya pada strategi spasial belum sempurna.

# Operasi Analisa GIS yang Umum

Operasi GIS	Fungsi
Search	Thematic Search, Search by region, (re-)classification
Location Analysis	Buffer, corridor, overlay
Terrain Analysis	Slope/aspect, Catchment, drainage network
Flow Analysis	Connectivity, shortest path
Distribution	Change detection, proximity, nearest neighbor
Spatial Analysis/Statistics	Pattern, centrality, autocorrelation, indices of similarity, topology: hole description
Measurements	Distance, perimeter, shape, adjacency, direction

# Tujuan 2

- Mempelajari alternatif algoritma untuk memproses query spasial
  - Apakah *building block* untuk query spasial?
  - Apa strategi umum untuk tiap *building block*?

# Cakupan



- Pilihan *building block* untuk query spasial
- Pilihan strategi proses untuk *building block*
- Bagaimana memilih strategi “terbaik” dari yang bisa diterapkan?

# Building Block untuk Query Spasial

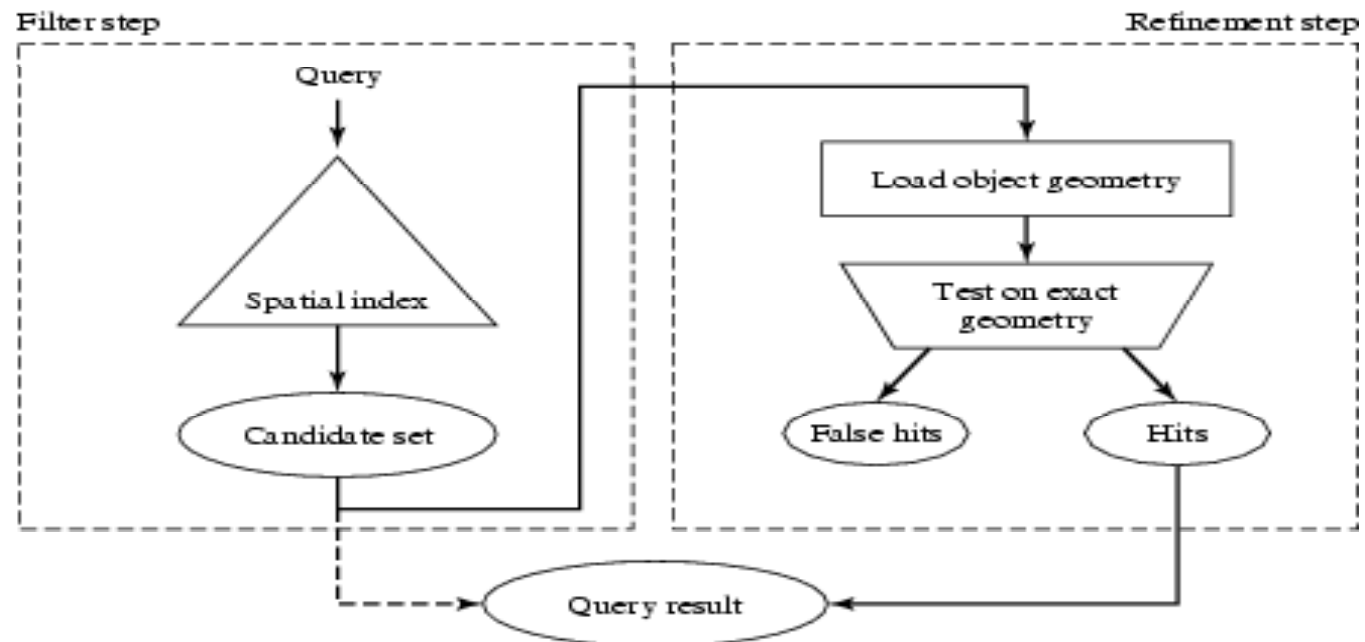
- Tantangan dalam memilih *building block*
  - Jenis data beragam - titik, garis, poligon, ...
  - Operator beragam - topological, euclidean, set-based, ...
  - Banyaknya kumpulan algoritma geometris komputasi
    - operasi spasial berbeda untuk jenis data yang berbeda
  - Keinginan untuk membatasi kerumitan DBMS spasial

# Building Block untuk Query Spasial

- Bagaimana menyederhanakan pilihan jenis data dan operator?
  - Penggunaan kembali Geographic Information System (GIS)
    - Yang sudah menerapkan operasi dan jenis data spasial
    - Tetapi mungkin punya kesulitan untuk memproses data berukuran besar pada disk
  - DBMS spasial mengurangi jumlah objek yang akan diproses GIS
  - DBMS spasial digunakan sebagai filter
  - Ini merupakan pendekatan *filter* dan *refinement*

# Paradigma Filter-Refine

- Pemrosesan query spasial Q
  - Tahap *Filter* : mencari kelompok objek K dalam menjawab Q
    - Gunakan perkiraan operator dan jenis data spasial
  - Tahap *Refinement* : menemukan jawaban yang tepat untuk Q
    - Gunakan GIS untuk memproses K
    - Gunakan operasi dan jenis data spasial yang tepat



# Paradigma Filter-Refine

- Tahap *Filter* → objek spasial ditampilkan dengan gunakan MBR
- Contoh:“ Cari semua sungai yang wilayah banjirnya overlap dengan daerah Pemukiman”. Dalam SQL, menjadi:
  - SELECT River.Name
  - FROM River
  - WHERE Overlap(River.Flood-Plain, : Pemukiman)
- Jika kita memperkirakan wilayah banjir semua sungai dengan MBR, lalu menentukan apakah titik di dalam MBR lebih murah dibanding memeriksa titik pada poligon yang tidak beraturan.
- Jawabannya adalah set besar jawaban yang real → set kandidat
- Dasar spasial bisa digantikan dengan perkiraan untuk menyederhanakan *query optimizer*.
  - Contoh : touch (river.flood-plain, : Pemukiman) → overlap(MBR(river.flood-plain, : Pemukiman))



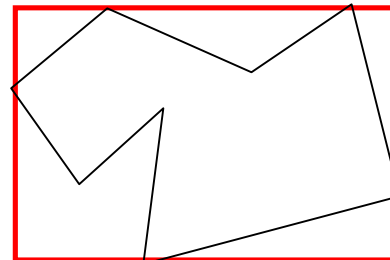
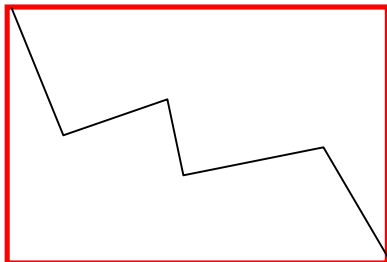
# Paradigma Filter-Refine

- *Refinement:*

- Geometri yang tepat untuk tiap elemen dari set kandidat
- Biasanya memerlukan penggunaan algoritma CPU secara intensif
- Kadang-kadang diproses diluar basis data spasial dalam program aplikasi seperti GIS, menggunakan set kandidat dari tahap *filter*.

# Perkiraan Jenis Data Spasial

- Memperkirakan jenis data spasial
  - *Minimum orthogonal bounding rectangle* (MOBR atau MBR)
    - Memperkirakan garis, poligon, ...
    - Lihat contoh di bawah (segi empat merah adalah MBR untuk objek yang berwarna hitam)
  - MBR digunakan oleh indeks spasial, mis. R-tree
  - Algoritma untuk operasi spasial MBR, cukup sederhana
- Q → Pernyataan mana pada tabel berikut yang memberikan hasil MBR?



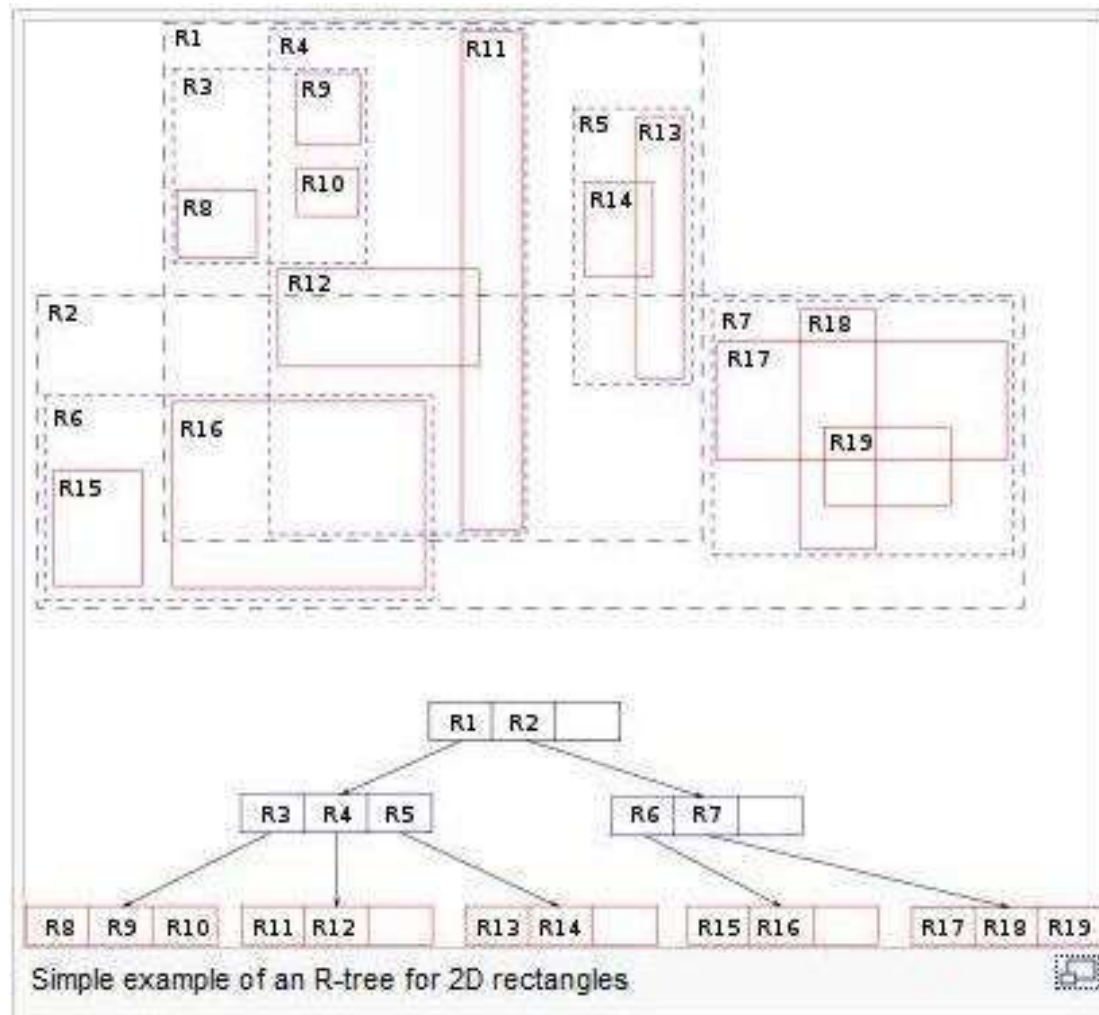
# Standar OGIS

Kategori	Pernyataan	Hasil
Basic Function	SpatialReference()	Returns the underlying coordinate systems of the geometry
	Envelope()	Returns the minimum orthogonal bounding rectangle of the geometry
	Export()	Returns the geometry in a different representation
	InEmpty()	Returns true if the geometry is a null set
	InSimple()	Returns true if the geometry is simple (no self-intersection)
	Boundary()	Returns the boundary of the geometry
Topological/Set Operators	Equal	Returns true if the interior and boundary of the two geometries are spatially equal
	Disjoint	Returns true if the boundaries and interior do not intersect
	Intersect	Returns true if the geometries are not disjoint
	Touch	Returns true if the boundaries of two surfaces intersect but the interiors do not
	Cross	Returns true if the interior of a surface intersects with a curve

# Standar OGIS

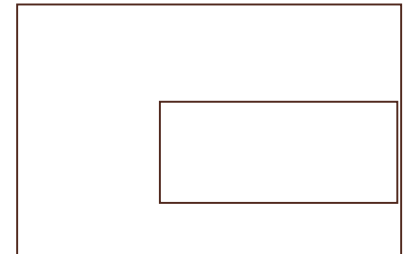
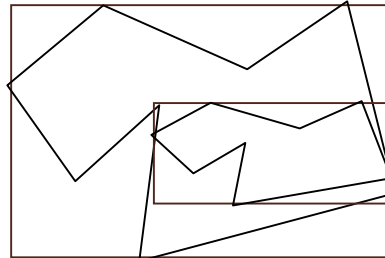
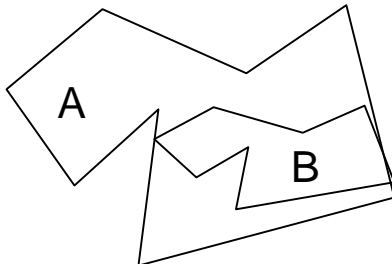
Kategori	Pernyataan	Hasil
Topological/Set Operators	Within	Returns true if the interior of the given geometry does not intersect with the exterior of another geometry
	Contains	Tests if the given geometry contains another given geometry
	Overlap	Returns true if the interiors of two geometries have nonempty intersection
Spatial Analysis	Distance	Returns the shortest distance between two geometries
	Buffer	Returns a geometry that consists of all points whose distance from the given geometry is less than or equal to the specified distance
	ConvexHull	Returns the smallest convex geometric set enclosing the geometry
	Intersection	Returns the geometric intersection of two geometries
	Union	Returns the geometric union of two geometries
	Difference	Returns the portion of a geometry that does not intersect with another given geometry
	SymmDiff	Returns the portions of two geometries that do not intersect with each other

# R(rectangle)-tree



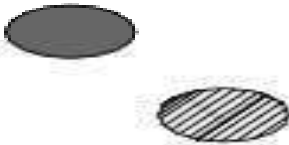







# Perkiraan Operasi Spasial

- Memperkirakan operasi spasial
  - DBMS spasial memproses MBR untuk tahap *refinement*
  - Predikat yang *overlap* digunakan untuk memperkirakan operasi topologi
  - Contoh:
    - $\text{inside}(A, B)$  digantikan dengan  $\text{overlap}(\text{MBR}(A), \text{MBR}(B))$  dalam tahap *filter*
    - Lihat gambar di bawah - A adalah poligon luar dan B adalah poligon dalam
    - $\text{inside}(A, B)$  benar jika  $\text{overlap}(\text{MBR}(A), \text{MBR}(B))$
    - Tapi *overlap* hanyalah filter untuk *inside predicate* yang selanjutnya memerlukan *refinement*



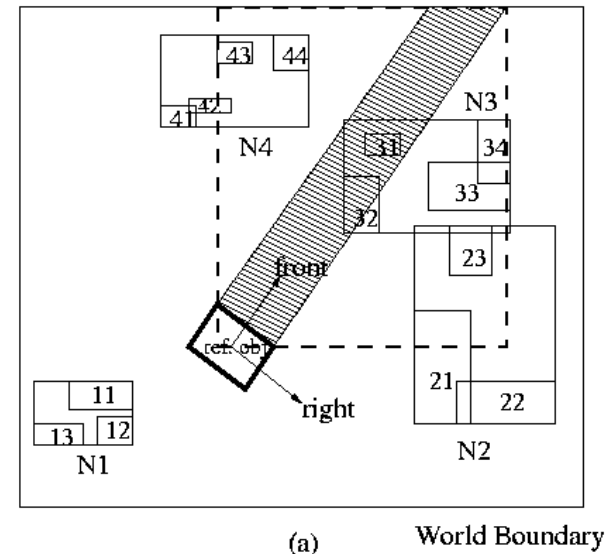
# Perkiraan Operasi Spasial

- Operasi topologi

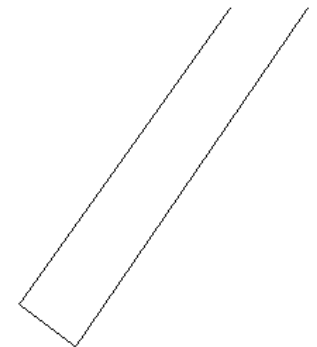
			
$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ <p>disjoint</p>	$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ <p>contains</p>	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$ <p>inside</p>	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ <p>equal</p>
			
$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ <p>meet</p>	$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ <p>covers</p>	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$ <p>coveredBy</p>	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ <p>overlap</p>

# Contoh Tahap Filter

- Query:
  - Tuliskan objek di depan pengamat V
- Query overlap yang sama
  - Wilayah arah adalah poligon
  - Tulis obyek yang overlap dengan
    - $\text{poligon}(\text{front}(V))$
- Perkiraan query
  - Tulis obyek yang overlap dengan
    - $\text{MBR}(\text{polygon}(\text{front}(V)))$



(b) Range Query




(c) Direction region



# Pilihan *building blocks*

- Pilihan *building blocks*
  - Beragam sesuai dengan produk dan pembuat software
- Daftar *building blocks*
  - Point Query- Tuliskan nama kota yang disorot pada peta digital
    - Menghasilkan satu objek spasial
  - Range Query- Tuliskan semua negara yang dilalui sungai Amazon.
    - Menghasilkan sejumlah objek dalam sebuah wilayah spasial
  - Spatial Join: Tuliskan semua pasangan sungai dan negara yang overlap
    - Menghasilkan pasangan dari dua tabel yang memenuhi operasi spasial
  - Nearest Neighbor: Carilah kota terdekat dengan Mount Everest.
    - Menghasilkan satu objek spasial

# Cakupan

- 
- Pilihan *building block* untuk query spasial
  - Pilihan strategi proses untuk *building block*
  - Bagaimana memilih strategi “terbaik” dari yang bisa diterapkan?

# Strategi untuk tiap *Building Block*

- Pilihan strategi
  - Beragam, bergantung pada produk dan pembuat software
  - Sejumlah strategi perlu struktur file atau indeks khusus
- Gambaran strategi
  - Ingat : ada banyak strategi untuk tiap *building block*!
  - Fokus pada konsep, bukan prosedur

# Strategi untuk Point Queries

- Contoh Point Query
  - Tuliskan nama kota yang disorot pada peta digital.
  - Menghasilkan satu objek spasial
- Daftar strategi
  - Scan semua sektor B disk dari file data
  - Jika record diurutkan dengan space filling curve (misal Z-order)
    - Maka gunakan pencarian biner pada Z-order untuk titik yang dicari
    - Memeriksa log sektor disk
  - Jika pada lokasi spasial objek data tersedia indeks,
    - Maka gunakan operasi find() terhadap indeks

# Strategi untuk Range Queries

- Contoh Range Query
  - Tuliskan semua negara yang dilalui sungai Amazon.
  - Menghasilkan sejumlah objek dalam sebuah wilayah spasial
- Daftar strategi
  - Scan semua sektor B disk pada file data
  - Jika record diurutkan dengan space filling curve (misal Z-order)
    - Maka tentukan range nilai Z-order yang memenuhi range query
    - Gunakan pencarian biner untuk mendapatkan Z-order terendah dalam jawaban query
    - Scan forward dalam file data sampai z-order tertinggi memenuhi query
  - Jika pada lokasi spasial objek data tersedia indeks
    - Maka gunakan operasi range-query terhadap indeks

# Strategi untuk Spatial Joins

- Contoh Spatial Join :
  - Tuliskan semua pasangan sungai dan negara yang overlap
  - Menghasilkan pasangan dari dua tabel yang memenuhi operasi spasial
- Daftar strategi
  - Nested loop:
    - Uji semua kemungkinan pasangan untuk operasi spasial
    - Semua sungai dipasangkan dengan semua negara
  - Partisi ruang :
    - Uji pasangan objek dari wilayah spasial yang umum saja
    - Sungai sungai di Afrika hanya diuji dengan negara-negara Afrika saja
  - Tree Matching
    - Pemasangan hirarki kelompok objek dari tiap tabel
  - Lainnya, mis. spatial-join-index based, external plane-sweep, ...

# Strategi untuk Query Nearest Neighbor

- Contoh Nearest Neighbor
  - Carilah kota terdekat dengan Mount Everest.
  - Menghasilkan satu objek spasial dari file data city C
- Pilihan strategi
  - Pendekatan dua tahap
    - Ambil sektor disk C yang berisi lokasi Mt. Everest
    - $M$  = jarak minimum ( Mt. Everest, kota-kota dalam sektor yang diambil)
    - Menguji semua kota dalam jarak  $M$  dari Mt. Everest (Range Query)
  - Pendekatan tahap tunggal
    - Algoritma berulang untuk R-tree
    - Menghilangkan kandidat yang dikalahkan kandidat lain

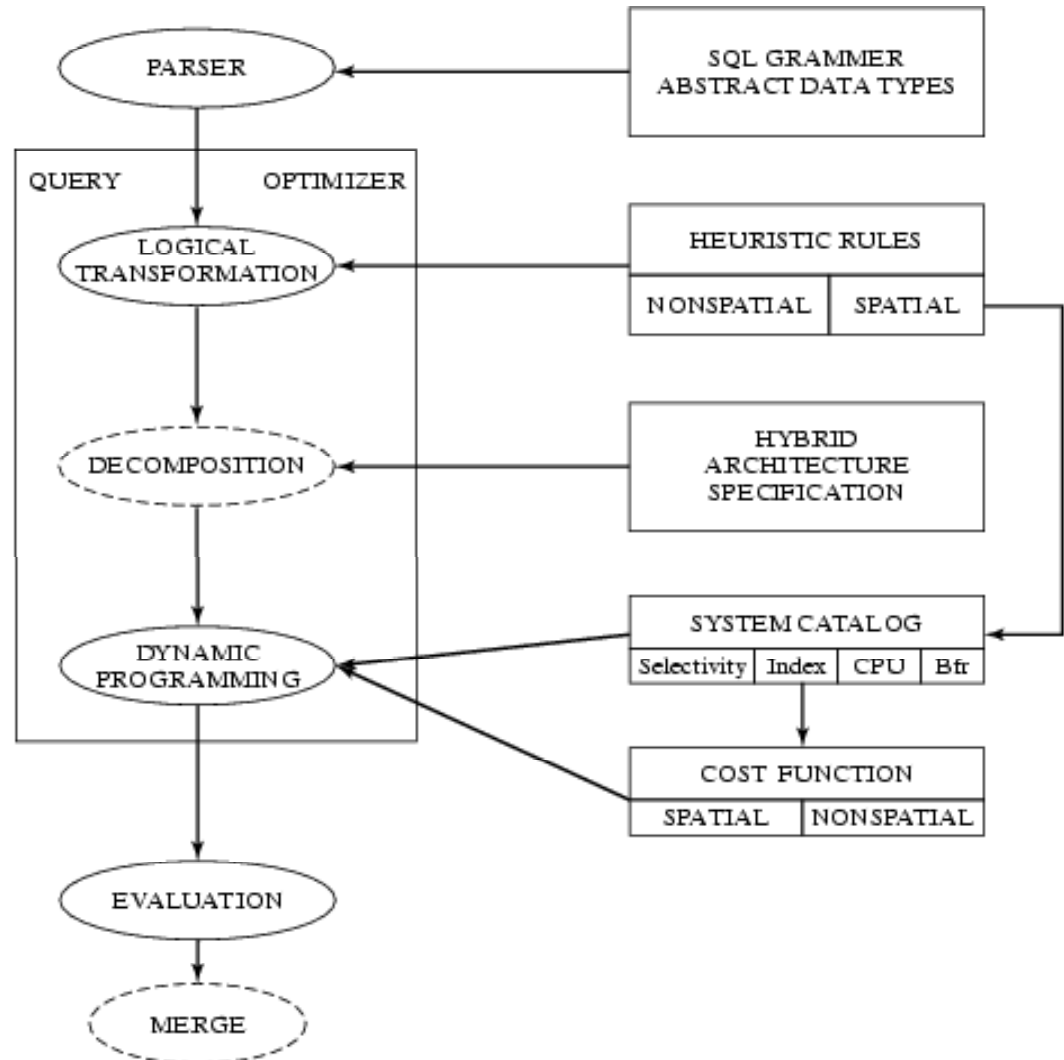
# Tujuan ke 3

- Mempelajari *query optimizers* (QO)
  - Langkah-langkah QPO
  - Bagaimana membandingkan strategi untuk *building block*?



# Proses Query dan Proses Optimalisasi

- Seperti sebuah perjalanan
  - Mulai : Query SQL
  - Akhir: Rencana eksekusi
  - Perhentian antara
    - Pohon query
    - Transformasi pohon logis
    - Pilihan strategi
- Apa yang terjadi setelah perjalanan?
  - Rencana eksekusi dijalankan
  - Jawaban Query diperoleh



# Proses QPO

- Idenya → hindari rencana buruk dan pilih rencana baik
- Dua bagian yang berbeda:

## 1. Transformasi Logis

- Pernyataan logis harus di *scan* melalui *parser*
- Periksa sintaks dan ubah pernyataan → pohon query
- Proses Query → dimulai pada daun ke atas pohon
- Nodes = *building blocks* dari query spasial
- Children = input pada *building block*
- Leafs = Tabel
- Contoh query SQL dan pohon querynya :

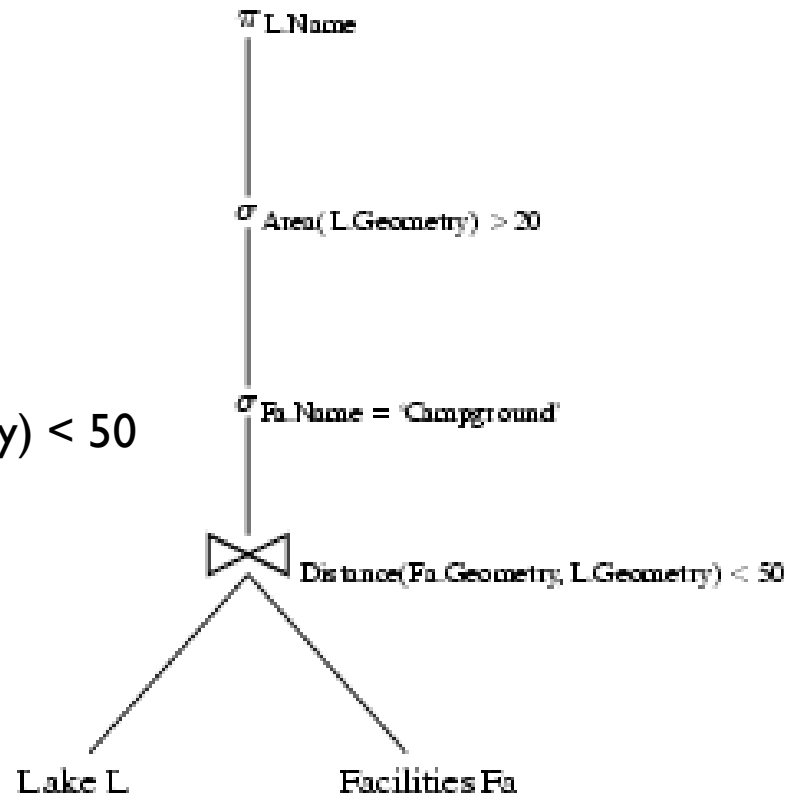
## 2. Pemrograman Dinamis

# Pohon Query

- Contoh: Carilah nama danau yang luasnya lebih dari 20 hektar dan jaraknya kurang dari 50 km dari campground

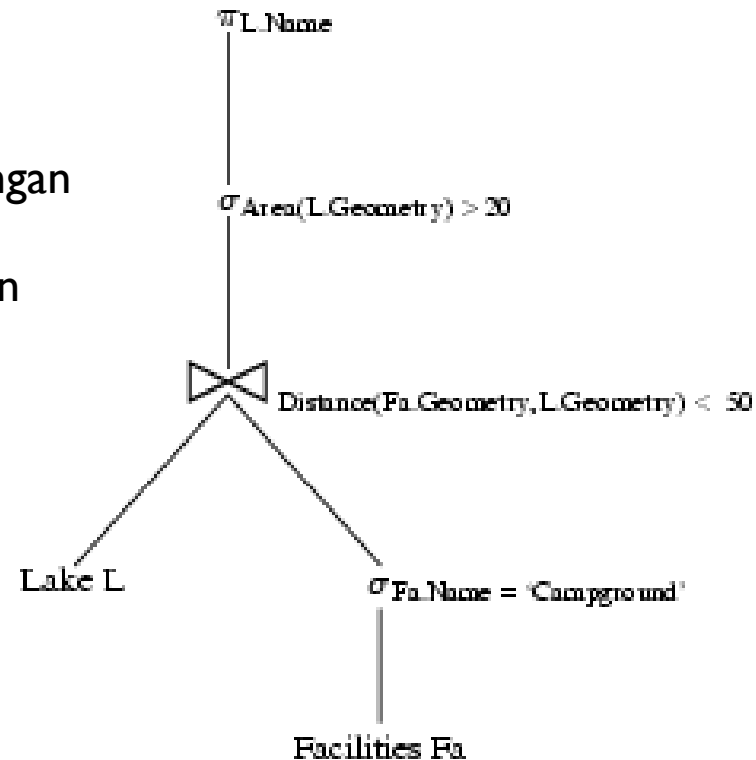
```

SELECT  L.Name
FROM    Lake L, Facilities Fa
WHERE   Area (L.Geometry) > 20 AND
        Fa.Name = 'campground' AND
        Distance (Fa.Geometry, L.Geometry) < 50
  
```



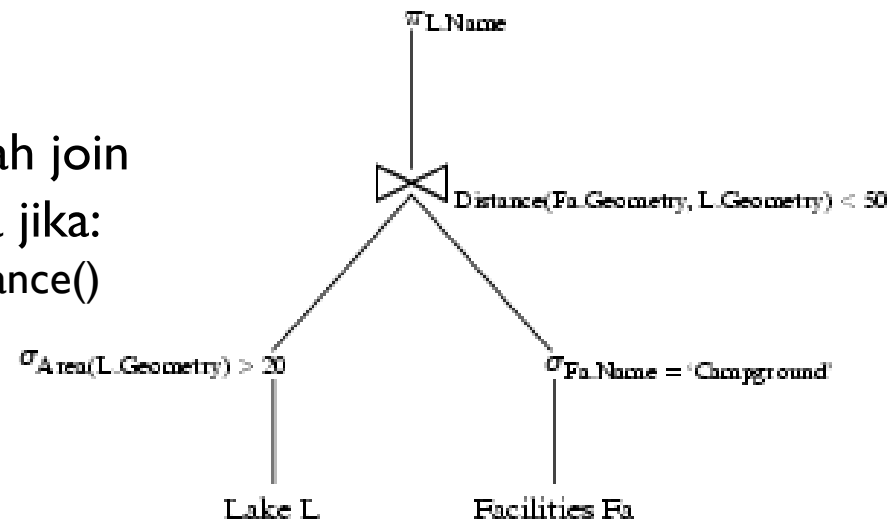
# Transformasi Logis dari Pohon Query

- Alasan
  - Transformasi tidak mengubah jawaban query
  - Tapi dapat mengurangi biaya komputasi dengan:
    - Mengurangi data yang dihasilkan sub-query
    - Mengurangi kebutuhan komputasi node induk
- Contoh transformasi
  - Menekan operasi select di bawah join
  - Contoh gambar disamping, bandingkan dengan gambar sebelumnya
  - Mengurangi ukuran tabel untuk operasi join
- Transformasi umum lain
  - Menekan project ke bawah
  - Mengatur urutan operasi join
  - ...



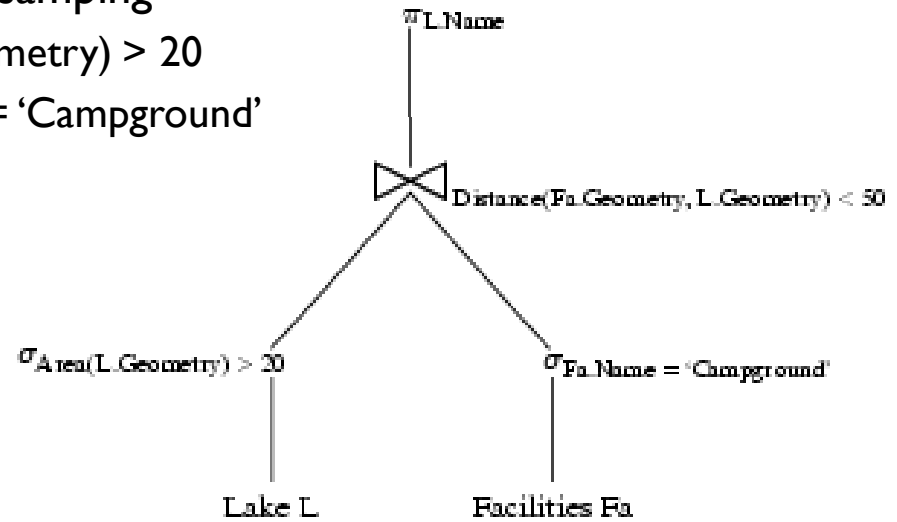
# Transformasi Logis dan Query Spasial

- Aturan transformasi logis tradisional
  - Untuk Query relasi dengan operasi dan jenis data sederhana
    - Biaya CPU jauh lebih kecil dibanding biaya I/O
  - Perlu ditinjau ulang untuk query spasial, melibatkan jenis data kompleks dan operasi
    - Biaya CPU lebih tinggi
- Contoh:
  - Menekan seleksi spasial di bawah join
  - Bisa jadi tidak mengurangi biaya jika:
    - Area() lebih mahal dari pada Distance()



# Rencana Eksekusi

- Rencana eksekusi memiliki 3 komponen
  - Pohon query
  - Strategi yang dipilih untuk tiap node bukan *leaf*
  - Evaluasi terurut untuk node bukan *leaf*
- Contoh
  - Strategi untuk pohon Query di samping
    - Gunakan *scan* untuk  $\text{Area}(\text{L.Geometry}) > 20$
    - Gunakan indeks untuk  $\text{Fa.Name} = \text{'Campground'}$
    - Gunakan join partisi ruang untuk
      - $\text{Distance}(\text{Fa}, \text{L}) < 50$





# Pemilihan strategi untuk *building blocks*

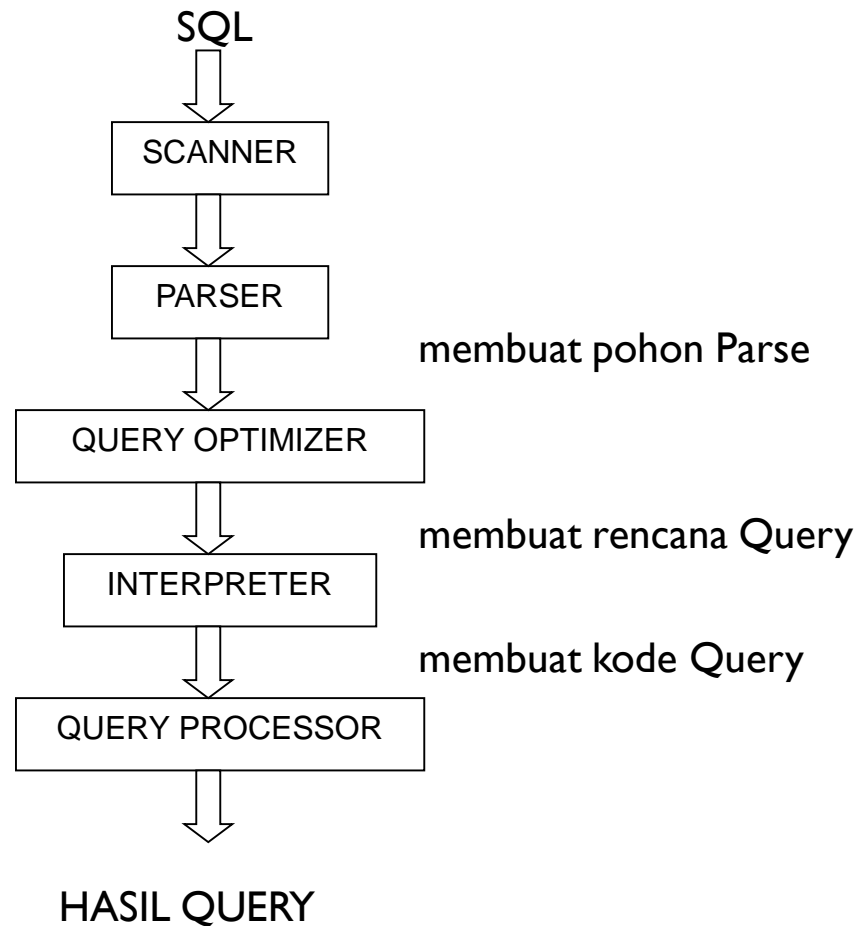
- Skema Prioritas
  - Periksa bisa dijalankan atau tidaknya tiap strategi berdasar struktur file dan indeks yang diberikan.
  - Strategi pemilihan prioritas tertinggi
  - Prosedur ini cepat, digunakan untuk query yang kompleks.
- Pendekatan berbasis aturan
  - Sistem memiliki sekumpulan aturan yang memetakan situasi ke pilihan strategi
  - Contoh: penggunaan *scan* untuk query range jika hasilnya > 10% dari file data
- Pendekatan berbasis biaya

# Pemilihan strategi untuk *building blocks* - 2

- Pendekatan berbasis model biaya
  - *Building block* tunggal
    - Memakai formula untuk memperkirakan biaya dari tiap strategi, berdasarkan ukuran tabel dll.
    - Pilih strategi dengan biaya paling sedikit
  - Pohon query
    - Kombinasi biaya termurah dari pilihan strategi untuk node bukan daun
    - Algoritma pemrograman dinamis
- Praktek komersial
  - DBMS relasional menggunakan pendekatan berbasis biaya untuk *building block* relasional
  - Tetapi model biaya untuk strategi spasial belumlah matang
  - Pendekatan berbasis aturan sering digunakan untuk strategi spasial



# Langkah dalam Optimalisasi Query



# Tujuan ke 4

- Mempelajari Tren
  - Pengaruh dari lingkungan terdistribusi, berbasis web, dan komputasi paralel

# Tren dalam QPO

- Alasan
  - GIS dan DBMS Spasial sangat berharga bagi banyak organisasi
  - Biaya keberhasilan adalah mendapat permintaan baru dari konsumen:
    - Untuk mendukung lingkungan dan hardware komputasi yang baru
    - Untuk mendukung aplikasi baru
- Lingkungan Komputasi yang Baru
  - Komputasi terdistribusi
  - Internet dan web
  - Komputer paralel
- Aplikasi yang Baru
  - Layanan berbasis lokasi, transportasi
  - Data mining
  - Data Raster

# Basis Data Spasial Terdistribusi

- Lingkungan terdistribusi
  - Kumpulan beragam komputer tak dikenal
  - Tersambung melalui jaringan
  - Arsitektur klien-server
    - Komputer server menyediakan layanan yang sudah didefinisikan dengan baik
    - Komputer klien menggunakan layanan itu
- Isu baru untuk DBMS Spasial
  - Model data konsep -
    - Translasi antara skema yang beragam
  - Model data logis
    - Penamaan dan query tabel di DBMS spasial lainnya
    - Menjaga salinan tabel (pada DBMS spasial lain) konsisten dengan tabel aslinya
  - QPO
    - Biaya transfer data melalui jaringan mungkin melebihi biaya CPU dan I/O
    - Strategi baru untuk mengontrol biaya transfer data

# Basis Data Spasial Terdistribusi - 2

- Strategi transfer data untuk menggabungkan dua tabel pada lokasi berbeda
  - Mentransfer satu tabel ke lokasi lainnya
  - Strategi semi-join
    - Mentransfer kolom gabungan pada satu tabel ke lokasi lain
    - Mentransfer kembali baris yang sesuai pada tabel lainnya ke lokasi pertama
  - Semi join seringkali lebih murah dibanding mentransfer satu tabel ke lokasi lain

Dua tabel di lokasi berbeda akan digabung dengan overlap dari D\_MBR overlap FARM\_MBR

FARM

<u>FID</u> (10 bytes)	OWNER_NAME (10 bytes)	FARM_BOUNDARY (2000 bytes)	FARM_MBR (16 bytes)
--------------------------	--------------------------	-------------------------------	------------------------

DISEASE\_MAP

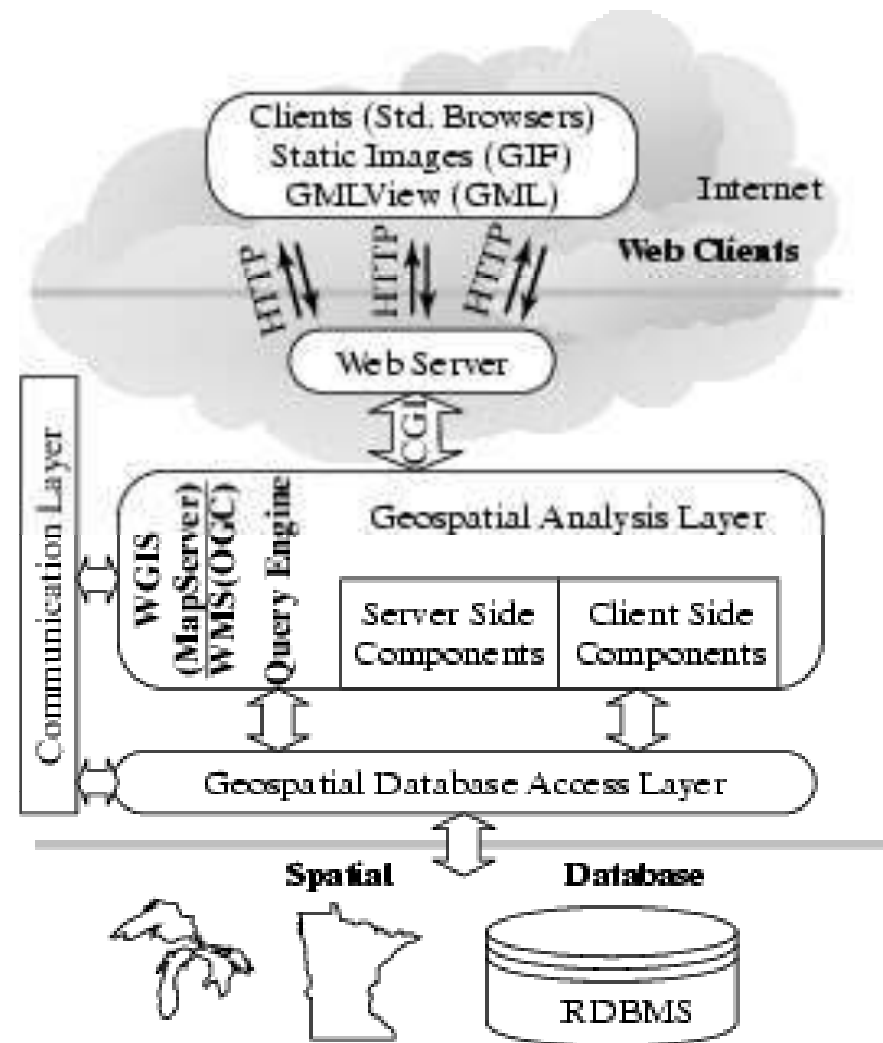
<u>MAP-ID</u> (10 bytes)	DISEASE_NAME (20 bytes)	DISEASE_BOUNDARY (2000 bytes)	D_MBR (16 bytes)
-----------------------------	----------------------------	----------------------------------	---------------------

# Internet dan (World-wide-)web

- Lingkungan Internet dan Web
  - Medium akses informasi yang sangat populer beberapa tahun belakangan ini
  - Lingkungan terdistribusi
  - Server Web, klien web:
    - Format data umum (mis. Html, xml)
    - Protokol komunikasi yang umum (mis. http)
    - Penamaan-Uniform resource locator (URL) – mis. [www.uinjkt.ac.id](http://www.uinjkt.ac.id)
- Isu baru untuk DBMS Spasial
  - Penawaran layanan DBMS spasial pada web
  - Penggunaan format data web, protokol komunikasi dll
  - Mengevaluasi dan meningkatkan web untuk server dan klien DBMS spasial

# Sistem Basis Data Spasial Berbasis-Web

- DBMS spasial pada web
  - MapServer
  - DBMS spasial berkomunikasi dengan webserver
  - Webserver berkomunikasi dengan web klien
- Praktek komersial
  - Beberapa produk berbasis web
  - Format data web untuk data spasial
    - GML (Geographic Markup Language)
    - WMS (Web Map Server)



# Sistem Basis Data Spasial Berbasis Web

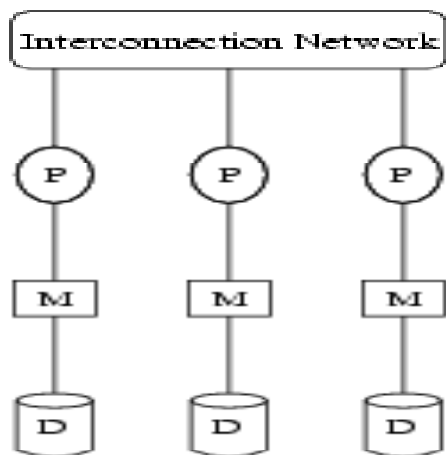
- Arsitektur WGIS (Map Server) – dikembangkan Univ. of Minnesota
  - Dirancang dan diimplementasikan menggunakan standar arsitektur 3-tier
  - Tier 1 → klien, standar browser
  - Tier 2 → server aplikasi:
    - Layer 1: Modul interface gateway yang umum → merespon permintaan http
    - Layer 2 : Sistem analisa Geospasial
    - Layer 3 : Sistem komunikasi → mengidentifikasi set data yang disyaratkan dan mengirimkan kembali ke sistem analisa geospasial
  - Tier 3 → Sistem akses basis data geospasial → terdiri dari antarmuka yang disesuaikan untuk mengakses format standar *image*



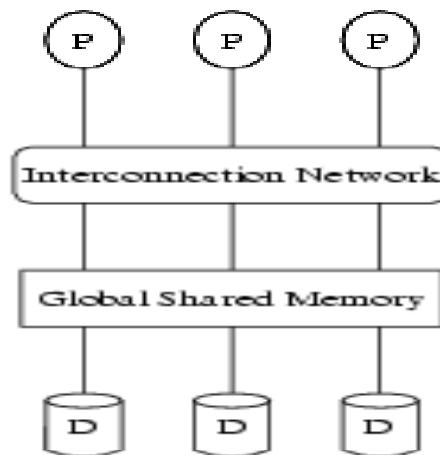
# Basis Data Spasial Paralel

- Lingkungan Paralel

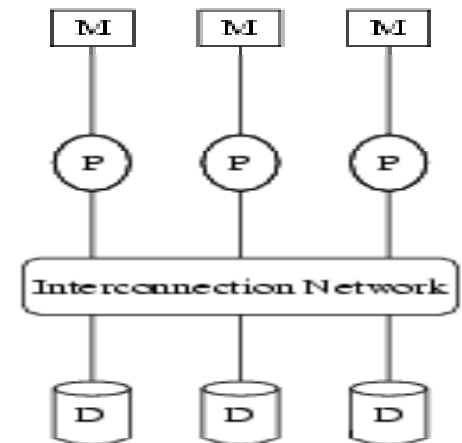
- Diperlukan dengan pertumbuhan data yang cepat membutuhkan waktu respon yang cepat
- Komputer dengan banyak CPU
- Dua ukuran utama untuk mengevaluasi → percepatan linier dan pembesaran linier
- Semua CPU dan Disk tersedia untuk DBMS spasial
- Bisa mempercepat proses query spasial



SHARED-NOTHING  
(a)



SHARED-MEMORY  
(b)



SHARED-DISK  
(c)

# Basis Data Spasial Paralel - 2

- Tiga arsitektur utama adalah Berbagi Memori, Berbagi Disk, dan Tidak Berbagi
  - Tidak Berbagi:
    - Meminimalkan interferensi antar prosesor yang berbeda
    - Kemampuan percepatan dan perbesaran linear
    - Menyeimbangkan masalah beban kerja
    - Kemungkinan masalah ketersediaan data ketika satu prosesor gagal
  - Berbagi Disk:
    - Tiap prosesor memiliki memori sendiri
    - Semua prosesor bisa mengakses semua disk pada sistem
  - Berbagi Memori:
    - Banyak CPU di hubungkan pada jaringan bisa mengakses memori sistem dan semua disk
    - Kemungkinan terjadi *bottleneck* dengan bertambahnya jumlah processor

# Ringkasan

- Optimalisasi dan Pemrosesan Query (QPO)
  - Mengubah query SQL menjadi rencana eksekusi
- Langkah proses QPO termasuk:
  - Pembuatan pohon query untuk query SQL
  - Pemilihan strategi untuk memproses setiap *node* dalam pohon query
  - Mengurutkan *node* untuk eksekusi
- Gagasan kunci untuk DBMS Spasial termasuk:
  - Paradigma *filter-refine* untuk mengurangi kerumitan
  - Strategi dan *bulding block* yang baru untuk query spasial