Query Processing and Optimization

- SQL queries are declarative
  - Users do not specify the procedure to answer it
  - DBMS needs to pick an algorithm to answer query
- Relevant SDBMS component
  - Query processing and optimization (QPO)
    - Picks algorithms to process a SQL query

What is Query Processing and Optimization (QPO)?

- Basic idea of QPO
  - In SQL, queries are expressed in high level declarative form
  - QPO translates a SQL query to an execution plan
    - over physical data model
    - using operations on file-structures, indices, etc.
  - Ideal execution plan answers Q in as little time as possible
  - Constraints: QPO overheads are small
    - Computation time for QPO steps << that for execution plan

Why QPO?

- Identify performance bottleneck for a query
  - Is it the physical data model or QPO ?
- How to help QPO speed up processing of a query ?
  - Providing hints, rewriting query, etc.
- How to enhance physical data model to speed up queries?
  - Add indices, change file- structures, …

Three Key Concepts in QPO

- Building blocks
  - Most DBMS have few building blocks:
    - select (point query, range query), join, sorting, ...
  - A SQL queries is decomposed in building blocks
- Query processing strategies for building blocks
  - DBMS keeps a few processing strategies for each building block
    - e.g. a point query can be answer via an index or via scanning data-file
- Query optimization
  - For each building block of a given query, DBMS QPO tries to choose
    - "Most efficient" strategy given database parameters
    - Parameter examples: Table size, available indices, …
    - Ex. Index search is chosen for a point query if the index is available

QPO Challenges

- Choice of building blocks
  - SQL Queries are based on relational algebra (RA)
  - Building blocks of RA are select, project, join
  - SQL3 adds new building blocks like transitive closure
- Choice of processing strategies for building blocks
  - Constraints: Too many strategies=> higher complexity
  - Commercial DBMS have a total of 10 to 30 strategies
    - 2 to 4 strategies for each building block
- How to choose the "best" strategy from among the applicable ones?
  - May use a fixed priority scheme
  - May use a simple cost model based on DBMS parameters

QPO Challenges in SDBMS

- Building Blocks for spatial queries
  - Rich set of spatial data types, operations
  - A consensus on "building blocks" is lacking
  - Current choices include spatial select, spatial join, nearest neighbor
- Choice of strategies
  - Limited choice for some building blocks, e.g. nearest neighbor
- Choosing best strategies
  - Cost models are more complex since
    - Spatial Queries are both CPU and I/O intensive
    - while Traditional queries are I/O intensive
  - Cost models of spatial strategies are *not mature*.

Building Blocks for Spatial Queries

- Challenges in choosing building blocks
  - Rich set of data types - point, line string, polygon, …
  - Rich set of operators - topological, euclidean, set-based, …
  - Large collection of computation geometric algorithms
    - for different spatial operations on different spatial data types
  - Desire to limit complexity of SDBMS
- How to simplify choice of data types and operators?
  - Reusing a Geographic Information System (GIS)
    - which already implements spatial data types and operations
    - however may have difficulties processing large data set on disk
  - SDBMS reduces set of objects to be processed by a GIS
  - SDBMS is used as a filter
  - This is filter and refinement approach

***Filter-Refine* Paradigm**

- Processing a spatial query Q
  - Filter step : find a superset S of object in answer to Q
    - Using approximate of spatial data type and operator
  - Refinement step : find exact answer to Q reusing a GIS to process S
    - Using exact spatial data type and operation

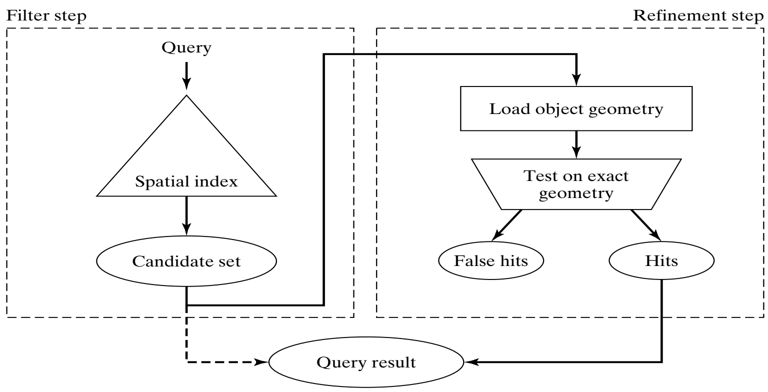
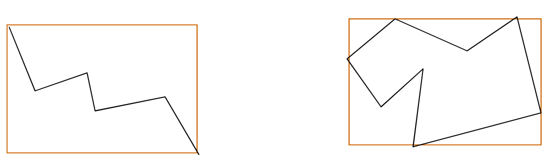


Fig 1

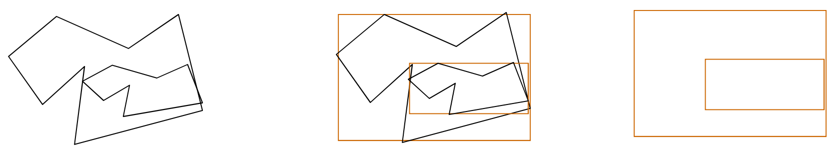
Approximate Spatial Data types

- Approximating spatial data types
  - Minimum orthogonal bounding rectangle (MOBR or MBR)
    - approximates line string, polygon, …
  - MBRs are used by spatial indexes, e.g. R-tree
  - Algorithms for spatial operations MBRs are simple



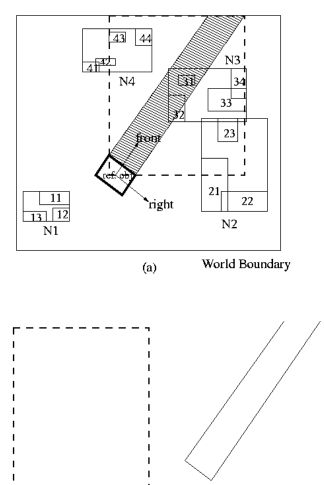
Approximate Spatial Operations

- Approximating spatial operations
  - SDBMS processes MBRs for refinement step
  - Overlap predicate used to approximate topological operations
  - Example: inside(A, B) replaced by
    - overlap(MBR(A), MBR(B)) in filter step
    - Refer picture below - Let A be outer polygon and B be the inner one
    - inside(A, B) is true only if overlap(MBR(A), MBR(B))
    - However overlap is only a filter for inside predicate needing refinement later



Filter Step Example

- Query:
  - List objects in front of a viewer V
- Equivalent overlap query
  - Direction region is a polygon
  - List objects overlapping with
    - polygon(front(V))
- Approximate query
  - List objects overlapping with
    - MBR(polygon (front (V)))



(a) World Boundary

(b) Range Query (c) Direction region

Approximate Spatial Operations - 2

- Exercise: Approximate following using overlap predicate
  - Cross(A, B), Touch(A, B), Disjoint(A, B)
- Given MBRs R and S, Provide conditions to test
  - Overlap(A, B)

Choice of building blocks

- Choice of building blocks
  - Varies across software vendors and products
  - Representative building blocks are listed here
- List of building blocks
  - Point Query- Name a highlighted city on a digital map.
    - Return one spatial object out of a table
  - Range Query- List all countries crossed by of the river Amazon.
    - Returns several objects within a spatial region from a table
  - Spatial Join: List all pairs of overlapping rivers and countries.
    - Return pairs from 2 tables satisfying a spatial predicate
  - Nearest Neighbor: Find the city closest to Mount Everest.
    - Return one spatial object from a collection

Strategies for Each Building Block

- Choice of strategies
  - Varies across software vendors and products
  - Representative strategies are provided
  - Some strategies need special file-structures or indices
- Description of strategies
  - There are multiple strategies for each building block!
  - *Focus on concepts rather than procedures*

Strategies for Point Queries

- Point Query Example
  - Name a highlighted city on a digital map.
  - Return one spatial object out of a table
- List of strategies
  - Scan all B disk sectors of the data file
  - If records are ordered using space filling curve (say Z-order)
    - then use binary search on the Z-order of search point
    - to examine about log(B, base = 2) disk sectors
  - If an index is available on spatial location of data objects,
    - then use find() operation on the index
    - number of disk sector examined = depth of index (typically 4 to 5)

Strategies for Range Queries

- Range Query Example-
  - List all countries crossed by of the river Amazon.
  - Returns several objects within a spatial region from a table
- List of strategies
  - Scan all B disk sectors of the data file
  - If records are ordered using space filling curve (say Z-order)
    - then determine range of Z-order values satisfying range query
    - Use binary search to get lowest Z-order within query answer
    - Scan forward in the data file till the highest z-order satisfying query
  - If an index is available on spatial location of data objects,
    - then use range-query operation on the index

Strategies for Spatial Joins

- Spatial Join Example:
  - List all pairs of overlapping rivers and countries.
  - Return pairs from 2 tables satisfying a spatial predicate
- List of strategies
  - Nested loop:
    - Test all possible pairs for spatial predicate
    - All rivers are paired with all countries
  - Space Partitioning:
    - Test pairs of objects from common spatial regions only
    - Rivers in Africa are tested with countries in Africa only!
  - Tree Matching
    - Hierarchical pairing of object groups from each table
  - Other, e.g. spatial-join-index based, external plane-sweep, …

Strategies for Nearest Neighbor Queries

- Nearest Neighbor Example
  - Find the city closest to Mount Everest.
  - Return one spatial object from city data file C
- List of strategies
  - Two phase approach
    - Fetch C's disk sector(s) containing the location of Mt. Everest
    - M = minimum distance(Mt. Everest, cities in fetched sectors)
    - Test all cities within distance M of Mt. Everest (Range Query)
  - Single phase approach
    - Recursive algorithm for R-tree
    - Eliminate candidates dominated by some other candidate

Query Processing and Optimizer process

- Overview
  - Start : A SQL Query
  - End: An execution plan
  - Intermediate Steps
    - query trees
    - logical tree transforms
    - strategy selection
- Next phase?
  - Execution plan is executed
  - Query answer returned

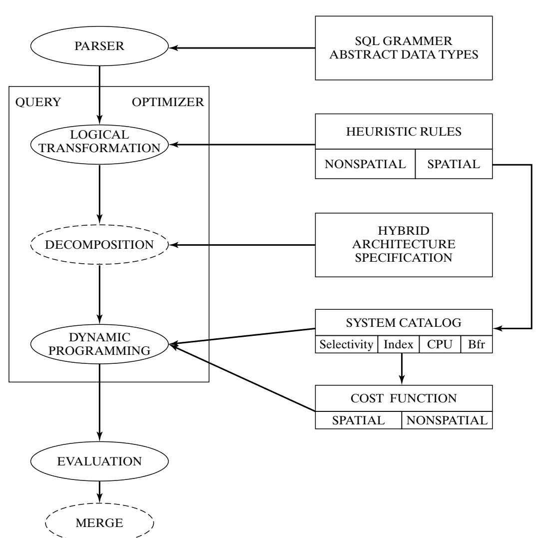


Fig 2

Query Trees

- Nodes = building blocks of (spatial) queries
- Children = inputs to a building block
- Leafs = Tables
- Example SQL query and its query tree follows:

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

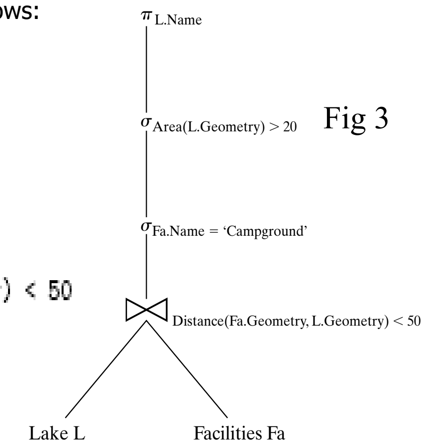


Fig 3

Logical Transformation of Query Trees

- Motivation
  - Transformation do not change the answer of the query
  - But can reduce computational cost by
    - reducing data produced by sub-queries
    - reducing computation needs of parent node
- Example Transformation
  - Push down select operation below join
  - Example: Fig. 4 (compare w/ Fig 3, last slide)
  - Reduces size of table for join operation
- Other common transformations
  - Push project down
  - Reorder join operations
  - ...

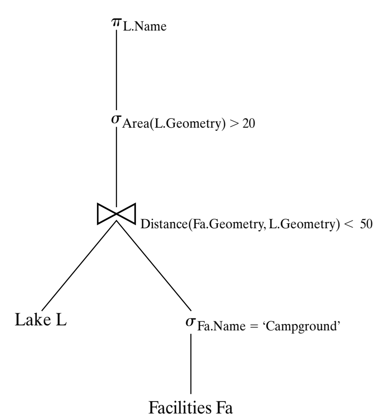


Fig 4

Logical Transformation and Spatial Queries

- Traditional logical transform rules
  - For relational queries with simple data types and operations
    - CPU costs are much smaller and I/O costs
  - Need to be reviewed for spatial queries
    - Complex data types, operations
    - CPU cost is higher
- Example:
  - Push down spatial selection below join
  - May not decrease cost if
    - area() is costlier than distance()

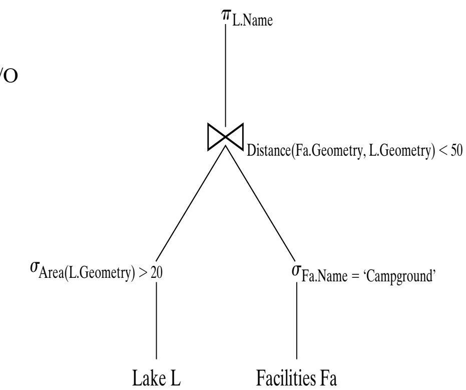
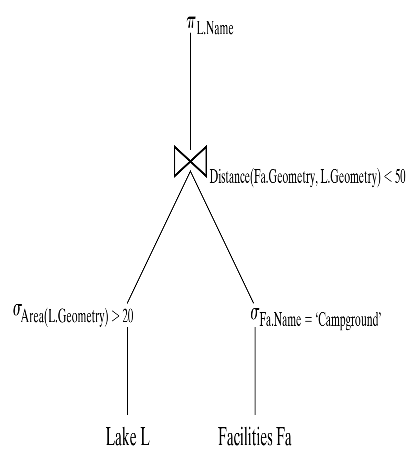


Fig 5

Execution Plans

- An execution plan has 3 components
  - A query tree
  - A strategy selected for each non-leaf node
  - An ordering of evaluation of non-leaf nodes
- Example
  - Strategies for Query tree in Fig
    - Use scan for Area(L.Geometry) > 20
    - Use index for Fa.Name = 'Campground'
    - Use space-partitioning join for
      - Distance(Fa, L) < 50
    - Use on-the-fly for projection
  - Ordering
    - As listed above



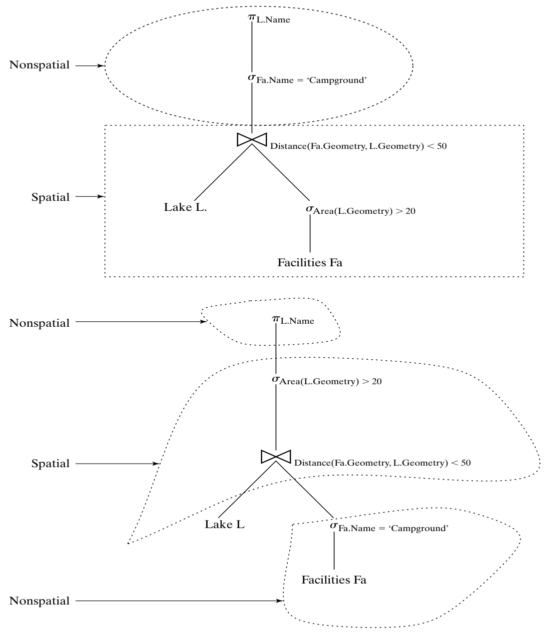
Choosing strategies for building blocks

- A priority scheme
  - Check applicability of each strategies given file-structures and indices
  - Choose highest priority strategy
  - This procedure is fast, Used for complex queries
- Rule based approach
  - System has a set of rules mapping situations to strategy choices
  - Example: Use scan for range query if result size > 10 % of data file
- Cost based approach

Choosing strategies for building blocks - 2

- Cost model based approach
  - Single building block
    - Use formulas to estimate cost of each strategy, given table size etc.
    - Choose the strategy with least cost
  - A query tree
    - Least cost combination of strategy choices for non-leaf nodes
    - Dynamic programming algorithm
- Commercial practice
  - RDBMS use cost based approach for relational building blocks
  - But cost models for spatial strategies are not mature
  - Rule based approach is often used for spatial strategies

Query Decomposition



Fig

Trends in Query Processing and Optimization

- Motivation
  - SDBMS and GIS are invaluable to many organizations
  - Price of success is to get new requests from customers
    - to support new computing hardware and environment
    - to support new applications
- New computing environments
  - Distributed computing
  - Internet and web
  - Parallel computers
- New applications
  - Location based services, transportation
  - Data Mining
  - Raster data

Distributed Spatial Databases

- Distributed Environments
  - Collection of autonomous heterogeneous computers
  - Connected by networks
  - Client-server architectures
    - Server computer provides well-defined services
    - Client computers use the services
- New issues for SDBMS
  - Conceptual data model -
    - Translation between heterogeneous schemas
  - Logical data model
    - Naming and querying tables in other SDBMSs
    - Keeping copies of tables (in other SDBMs) consistent with original table
  - Query Processing and Optimization
    - Cost of data transfer over network may dominate CPU and I/O costs
    - New strategies to control data transfer costs

Distributed SDBMS - 2

- Data-transfer strategies for joining 2 table at different sites
  - Transfer one table to the other site
  - Semi-join strategy
    - Transfer join column of one table to the other site
    - Transfer back the matching rows of the other table back to first site
  - Semi-join often is cheaper than transferring a table to other site

FARM

FID (10 bytes)	OWNER_NAME (10 bytes)	FARM_BOUNDARY (2000 bytes)	FARM_MBR (16 bytes)

DISEASE_MAP

MAP-ID (10 bytes)	DISEASE_NAME (20 bytes)	DISEASE_BOUNDARY (2000 bytes)	D_MBR (16 bytes)

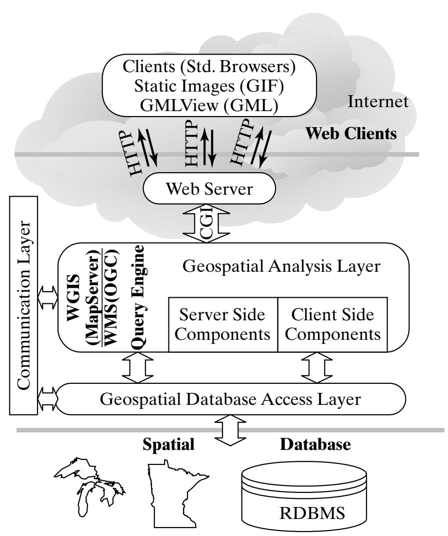
Fig 9: Two table at different sites to be joined on overlap of D_MBR overlap FARM_MBR

Internet and (World-wide-)web

- Internet and Web Environments
  - Very popular medium of information access in last few years
  - A distributed environment
  - Web servers, web clients
    - Common data formats (e.g. HTML, XML)
    - Common communication protocols (e.g. http)
    - Naming - uniform resource locator (url), e.g. www.cs.umn.edu
- New issues for SDBMS
  - Offer SDBMS service on web
  - Use Web data formats, communication protocols etc.
    - Example on next slide
  - Evaluate and improve web for SDBMS clients and servers

Web-based Spatial Database Systems

- SDBMS on web
  - MapServer case study
  - SDBMS talks to a web server
  - web server talks to web clients
- Commercial practice
  - Several web based products
  - Web data formats for spatial data
    - GML
    - WMS

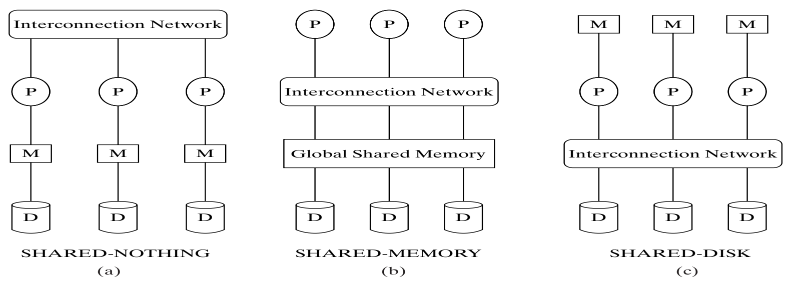


•Fig 10

Parallel Spatial Databases

- Parallel Environments
  - Computer with multiple CPUs, Disk drives (See Fig. 11 for examples)
  - All CPUs and disk available to a SDBMS
  - Can speed-up processing of spatial queries!

Fig 11



Parallel Spatial Databases - 2

- New issues for DBMS
  - Physical Data Model
    - Declustering: How to partition tables, indices across disk drives?
  - Query Processing and Optimization
    - Query partitioning: How to divide queries among CPUs?
    - Cost model of strategies on parallel computers
- Exmaple: Techniques for declustering (Fig. 12)
  - Simple technique: round robin based on an order (space filling curve)
  - Disk

Declustering for Data Partitioning

- Exmaple
  - A Simple Techniques for de-clustering (Fig. 12)
    - •1. Order the spatial objects using a space filling curve
    - •2. Allocate to disk drives in a round robin manner
  - Effective for point objects, e.g. pixels in an image
  - Many queries, e.g. large MBRs are parallelized well
    - •Ex. Consider a query to retrieve dat in bottom-left quarter of the space
      - •Two data points retrieved fromeach disk drive for Z-curve

3	4	5	6	7	0	1	2
6	7	0	1	2	3	4	5
1	2	3	4	5	6	7	0
4	5	6	7	0	1	2	3
7	0	1	2	3	4	5	6
2	3	4	5	6	7	0	1
5	6	7	0	1	2	3	4
0	1	2	3	4	5	6	7

Linear Method
disk-id =
(x + 5y) mod 8

7	0	1	2	3	4	5	6
6	7	0	1	2	3	4	5
5	6	7	0	1	2	3	4
4	5	6	7	0	1	2	3
3	4	5	6	7	0	1	2
2	3	4	5	6	7	0	1
1	2	3	4	5	6	7	0
0	1	2	3	4	5	6	7

CMD Method
disk-id =
(x + y) mod 8

42	43	46	47	58	59	62	63		2	3	6	7	2	3	6	7
40	41	44	45	56	57	60	61		0	1	4	5	0	1	4	5
34	35	38	39	50	51	54	55		2	3	6	7	2	3	6	7
32	33	36	37	48	49	52	53	→	0	1	4	5	0	1	4	5
10	11	14	15	26	27	30	31		2	3	6	7	2	3	6	7
8	9	12	13	24	25	28	29		0	1	4	5	0	1	4	5
2	3	6	7	18	19	22	23		2	3	6	7	2	3	6	7
0	1	4	5	16	17	20	21		0	1	4	5	0	1	4	5

Z-Curve Method -> disk-id = Z(x, y) mod 8

63	62	49	48	47	44	43	42		7	6	1	0	7	4	3	2
60	61	50	51	46	45	40	41		4	5	2	3	6	5	0	1
59	56	55	52	33	34	39	38		3	0	7	4	1	2	6	5
58	57	54	53	32	35	36	37		2	1	6	5	0	3	1	2
5	6	9	10	31	28	27	26	→	5	6	1	2	7	4	3	2
4	7	8	11	30	29	24	25		4	7	0	3	6	5	0	1
3	2	13	12	17	18	23	22		3	2	5	4	1	2	7	6
0	1	14	15	16	19	20	21		0	1	6	7	0	3	4	5

Hilbert Method -> disk-id = H(x, y) mod 8

A Case Study: High Performance GIS

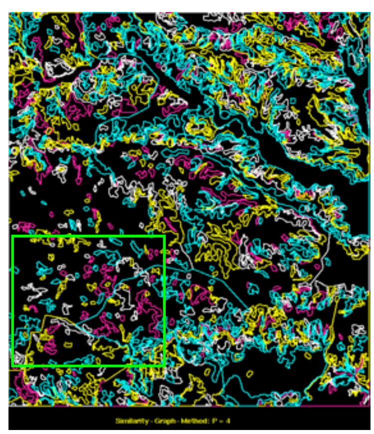
Goal: Meet the response time constraint for real time battlefield terrain visualization in flight simulator.

Methodology:

- Data-partitioning approach
- Evaluation on parallel computers,
  - e.g. Cray T3D, SGI Challenge.

Significance:

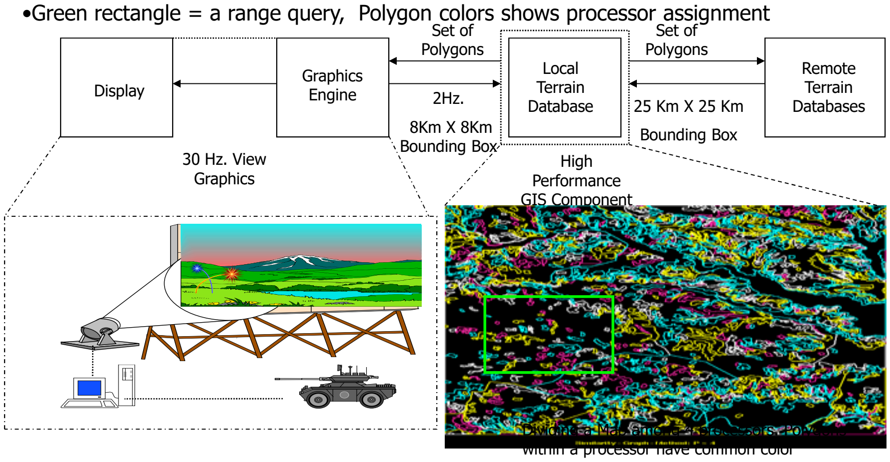
- A major improvement in capability of geographic information systems for determining the subset of terrain polygons within the view point (Range Query) of a soldier in a flight simulator using real geographic terrain data set.



Dividing a Map among 4 processors. Polygons within a processor have common color

A Case Study: High Performance GIS

- (1/30) second Response time constraint on Range Query
- Parallel processing necessary since best sequential computer cannot meet requirement
- Green rectangle = a range query, Polygon colors shows processor assignment



Real Time Visualization: A Case Study

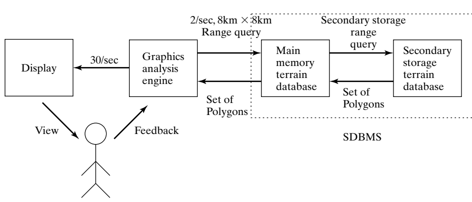


Fig 13

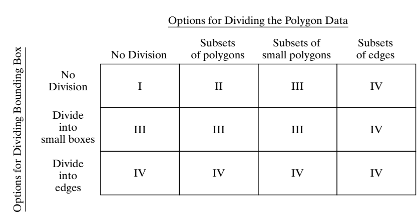


Fig 16

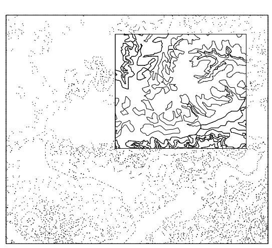


Fig 14 Range Query

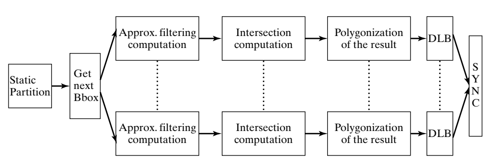


Fig 15

Summary

- Query processing and optimization (QPO)
  - translates SQL Queries to execution plan
- QPO process steps include
  - Creation of a query tree for the SQL query
  - Choice of strategies to process each node in query tree
  - Ordering the nodes for execution
- Key ideas for SDBMS include
  - Filter-Refine paradigm to reduce complexity
  - New building blocks and strategies for spatial queries