AN OVERVIEW OF SPATIAL INDEXING WITHIN RDBMS

DAVID DEHAAN
SQL ANYWHERE QUERY PROCESSING TEAM
SYBASE

THURSDAY 9 FEB 2012
CHERITON SCHOOL OF COMPUTER SCIENCE, CS 448/648
OUTLINE

• Motivation
• GIS Primer
• Spatial Indexing
  – Naïve Approaches
  – Data-driven Strategies
  – Space-driven Strategies
• Spatial Support in SQL Anywhere 12
• Advanced GIS: Indexing Round-Earth Data
MOTIVATION

What’s the point?

• Common Application Domains
  - CAD/CAM
  - GIS
  - Multimedia
  - OLAP

*Low Dimensionality*

• Common Queries
  - Point Data:
    - Polygon Range
    - Nearest Neighbour
  - Polygon Data:
    - Point Stabbing
    - Polygon Range (Intersection or Containment)

*High Dimensionality*

*Low Dimensionality*
GIS PRIMER
DATA REPRESENTATION

Two fundamental representations used in GIS (and graphics in general).

• Raster
  – Data stored as (one or more) pixelated images
  – Granularity fixed by pre-defined grid
  – Single pixel blends information from multiple objects
  – Storage size depends upon canvas size & granularity

• Vector
  – Each spatial object stored/rendered separately
  – Fundamental feature types:
    ▪ Points (0-dim)
    ▪ Lines (1-dim)
    ▪ Polygons (2-dim)
  – Granularity limited only by precision of coordinates
  – Storage size depends upon number & complexity of objects
SQL/MM DATA MODEL

Also the Open Geospatial Consortium standard for SQL access to spatial data.
SQL/MM SAMPLE QUERY

SELECT geometry
FROM countries
WHERE geometry.ST_Intersects( new ST_Polygon( 'POLYGON((-90 0, -90 -45, 0 -45, 0 0, -90 0))') , 4326 ) = 1
OGC STANDARDS WITHIN WEB SERVICES

Modern GIS applications integrate spatial data from a rich collection of sources.

SPATIAL INDEXING

NAÏVE APPROACHES
COMPOSITE KEY B-TREES

**Main Idea:** Compose index keys from multi-dimensional points

- Point data only
- Index key formed by fixing an ordering of dimensions

- Query processing:
  - Scan entire key range between min/max key values touched by query object
    - Poor performance if leading attributes are not equality predicates

- Fundamental weakness:
  - Spatial co-locality substantially different from index co-locality
FIXED GRIDS

Main Idea: Divide space into even partitions

- Point or polygon data
- Partitions store lists of data objects that intersect the cell

Query processing:
  - Scan list for each partition touched by query object
    - Performance depends heavily on range size relative to grid granularity

Fundamental weakness:
  - Fixed granularity does not adapt to data distribution or query workload
SPATIAL INDEXING

DATA-DRIVEN STRATEGIES
DATA-DRIVEN INDEXING STRATEGIES

• **Distinguishing Characteristics:**
  - Index key *composed from* values of indexed attribute(s)
  - Index layout/organization adjusts to distribution of inserted index keys

• Representative Index Types:
  - Single-dimensional
    ▪ Balanced binary trees
    ▪ B-trees
  - Multi-dimensional
    ▪ *kd*-trees
    ▪ Point quadtrees
    ▪ R-trees
**KD-TREES**

*Main Idea:* Generalize BSTs by splitting on alternating dimensions

- Point data only
- Not designed for secondary storage
POINT QUADTREES

Main Idea: Variation of $kd$-trees that splits on all dimensions simultaneously

- Point data only
- Not designed for secondary storage
R-TREES

Main idea: Generalize B-trees to rectangular keys

• Polygon data
• Optimized for secondary storage
• Query processing:
  – Recursively descend into all subtrees that intersect the query object
• Insertion:
  – Recursively descend into and extend any subtree intersecting the object
  – Node splitting like B-tree
R-TREES CONT’D

*Main idea:* Generalize B-trees to rectangular keys

- Tree structure can vary widely depending upon
  - Insertion order
  - Heuristic for breaking ties
- Degree of overlap heavily impacts query performance
  - No worst-case guarantees
- Plethora of variations:
  - Various insert heuristics
  - Partition data objects to avoid overlap (R+ tree)
  - Total ordering of leaves (Hilbert R-tree)

- Widely implemented:
  - Oracle
  - IBM Informix
  - Ingres
  - Postgres (PostGIS)
  - MySQL
  - SQLite (SpatiaLite)
  - ...
SPATIAL INDEXING

SPACE-DRIVEN STRATEGIES
• **Distinguishing Characteristics:**
  – Index key *a function of* values of indexed attribute(s)
    ▪ Design of function pre-supposes knowledge of domain
  – Index layout dictated by structure of key domain

• Representative Index Types:
  – Single-dimensional
    ▪ Various hash-based indexes
  – Multi-dimensional
    ▪ Fixed grids
    ▪ Region quadtrees
    ▪ Linearized (region) quadtrees (hybrid of space-driven & data-driven)
REGION QUADTREES

Main Idea: Variation of Point Quadtree that always splits into uniform quadrants

• Point or polygon data
• Not designed for secondary storage
LINEARIZED QUADTREES

Main Idea: Logical region quadtree physically stored within a B+-tree

- Key domain logically corresponds to uniform recursive partition of space
- Keys physically stored in B-tree (presumes total order)

- Space-filling curve translates spatial co-locality into index co-locality
  - Logical subtrees form contiguous key ranges
LINEARIZED QUADTREES CONT’D

Main Idea: Logical region quadtree physically stored within a B+-tree

• Point or polygon data
• Optimized for secondary storage
• Relies on object tessellation
• Insertion:
  – Tesselate data into tiles
  – Insert entry for each tile
• Query processing:
  – Tesselate query into tiles
  – Retrieve corresponding key ranges

• Widely implemented:
  – Oracle
  – IBM DB2
  – Microsoft SQL Server
  – Teradata
  – Sybase SQL Anywhere
SUMMARY: RDBMS SPATIAL INDEXES

Comparison of two widely-implemented indexes in general-purpose RDBMS

**R-tree**
- Domain agnostic
- Objects approximated as single rectangle
  - More precise filtering for (nearly) rectangular data
  - Smaller index
- Index structure/quality depends on insertion order
  - Degrades under updates
- Single forking index traversal
  - Complicates locking
  - Parallelism opportunities revealed during traversal

**Linearized Quadtrees**
- Domain fixed at index creation
- Objects approximated as multiple tiles
  - More precise filtering for non-rectangular data
  - More expensive scanning
- Index structure/quality independent of insertion order
  - Predictable performance
- Set of B+-tree ranges
  - Well studied/tuned locking
  - Parallelism opportunities revealed during tessellation
SPATIAL SUPPORT IN SYBASE SQL ANYWHERE 12
WHAT IS SQL ANYWHERE?

RDBMS component of the Sybase iAnywhere product suite.

• SQL Anywhere
  – Full-function, small-footprint relational DBMS with support for triggers, stored procedures, materialized views, intra-query parallelism, hot failover, OLAP queries, multidatabase capability, spatial data, ...

• Mobilink/SQL Remote
  – Two-way data replication/synchronization technologies for replicating data through different mechanisms to support occasionally-connected devices

• Ultralite
  – “fingerprint” database supports ad-hoc SQL on very small devices

• UltraliteJ
  – 100% Java fingerprint database for Blackberry and iPhone
DESIGN GOALS OF SQL ANYWHERE

• Ease of administration
  – Comprehensive yet comprehensible tools
• Good out-of-the-box performance
  – “Embeddability” features → self-tuning
  – Many environments have no DBA’s
• Cross-platform support
  – 32- and 64-bit Windows (7, Vista, XP, Server, 2000, 9x),
    Windows CE/Pocket PC, Linux 32- and 64-bit, HP-UX, AIX,
    Solaris (SPARC and Intel), Mac OS/X, Compaq Tru-64
• Interoperability
LINEAR QUADTREE TUNING ISSUES

How do you configure indexes if you haven’t seen the data or workload?

• Performance of linear quadtrees greatly affected by tessellation granularity
• Other systems:
  – Same algorithm used for both data and query objects
  – Parameters specified at index creation
    ▪ Number of subdivisions constituting a “level” (2, 4, 8, 16, 32, 64)
    ▪ Maximum number of levels in logical quadtree
    ▪ Maximum number of levels to descend within an object
    ▪ Maximum number of tessellation blocks per object
• Not obvious:
  – How to choose these parameters
  – How DBA can know that index is mis-configured
DECOUPLING DATA/QUERY TESSELLATION

Why should data and query objects be tessellated by the same algorithm?

• Data and query objects have competing priorities for tessellation granularity
  • Data object tessellation
    – Optimal granularity depends upon **query window**
    – Small queries → granular data (minimize false positives)
    – Large queries → coarse data (minimize duplicates)
  • Query object tessellation
    – Optimal granularity depends upon **data density**
    – Dense data → granular queries (minimize false positives)
    – Sparse data → coarse queries (minimize B-tree probes)
DATA-DRIVEN QUERY TESSELLATION

Defer tessellation decisions until query execution time.

• Data objects not tessellated
  – Single index entry per geometry (smallest containing block)

• Query objects tessellated dynamically
  – Candidate tessellation \( \rightarrow \) index range plan
  – Plan cost estimated using DBMS cost model (histograms)
  – Top-down branch-and-bound algorithm finds tessellation with optimal cost
  – Cost-based fallback to sequential table scan
  – Run-time per-tuple (query geometry) plan optimization
ADVANCED GIS CONCEPTS

INDEXING ROUND-EARTH DATA
SPATIAL REFERENCE SYSTEMS

What is the length of this line?

\text{LINESTRING( 0 0, 1 1 )}

- Cartesian coordinate system: \textbf{1.4142 units}
- Polar coordinate system: \textbf{1 unit}
- World Geodetic System (WGS) 84: \textbf{156899.568 m}

- SRS provides semantic context:
  - Coordinate system unit of measure (degrees, meters, etc.)
  - Coordinate bounds
  - Linear unit of measure
  - Planar vs spheroid data
  - Projection information for transforming between SRSs
  - Specified tolerance (SQL Anywhere)
ROUND EARTH COORDINATE SYSTEMS

Unfortunately, “as the crow flies” depends upon the breed of crow...

Multiple interpretations of lines on ellipsoidal earth:
1. Geodesic
   - Shortest path along true surface (ellipsoidal earth)
   - Widely used, but complex to compute and reason about
   - Used by Oracle, DB2

2. Great Elliptic Arc
   - Shortest path along circular earth (great circular arc), projected down to true surface
   - Simpler to compute and reason about
   - Used by Microsoft, Sybase
GNOMONIC PROJECTIONS

Great circular arcs project as straight lines onto any plane.

PROJECTING ROUND EARTH

Limitations of a single gnomonic projection.

PROJECTING ROUND EARTH

Project globe onto a regular octahedral; cut/unfold along equator; flatten.

http://www.progonos.com/furuti/MapProj/Dither/ProjPoly/Foldout/Octahedron/octahedron.html