Spatial Indexing

Sessions 1A and 1B
A Reminder...

- Project 3 is due next **Friday, 13th**!

- Please submit one only archive `project3.zip` containing:
  - `project3-indexer.zip`
  - `project3-searcher.zip`
  - `project2.zip`

- If you **plan to change your team status**, please let me know ASAP, so that you can submit your project 3 on CCLE.

- We have set up project 3 submission with the same teams as project 2.
Latent Semantic Indexing

\[ W (\text{words}) \]

\[ W \]

\[ (\text{docs}) \]

\[ V^T \]

\[ n \times p \]

\[ p \times p \]

\[ n \times k \]

\[ k \times k \]

\[ n \times p \]

\[ k \times p \]

\[ \text{Rank} \]

\[ \text{Reduction to } k \]
• Similarity between a query vector and the corpus

Each value in the vector represents the similarity between \( q \) and \( d_i \)
Spatial Indexing: Motivation

- DB Querying: nearest neighbor, range, where-am-I?, partial match.
- Efficient collision detection in 2 or 3D – Videogames!
- Computational fluid dynamics.
Grid Files

- Requires three data structures:
  - One directory \textit{DIR} – 2D array that references pages associated with cells.
  - Two scales, $S_x$ and $S_y$ – 1D arrays to store axes’ intervals.
• Insertions in a File Grid, for $M = 4$

Any problem with this method?
Quadtree

- An extended version of a Binary tree, but with (usually) four children.
- We can have at most $M$ objects per quadrant – otherwise, split into four children.

Increasing granularity
For $M = 3$

How do we build the Quadtree?
Watch a JavaScript demo at:
http://www.mikechambers.com/blog/2011/03/21/javascript-quadtree-implementation/

Large-Scale Liquid Simulation on Adaptive Hexahedral Grids

Florian Ferstl, Rüdiger Westermann and Christian Dick
R-Tree

- A data-driven approach – it is a **depth-balanced tree**.
- Organize points according to a containment relationship.
- Each node has **directory rectangle** (dr) which is **minimal bounding box** (mbb) of its children.
- Each node is mapped onto a disk page:

**Non-leaf node**

\[
\begin{array}{c}
e_1 \\
e_2 \\
\vdots \\
e_M
\end{array}
\]

- Non-leaf entries \((dr_i, addr_i)\) where \([0, M/2] \leq i \leq M\)

**Leaf node**

\[
\begin{array}{c}
o_1 \\
o_2 \\
\vdots \\
o_M
\end{array}
\]

- Leaf entries \((x_i, y_i)\) where \([0, M/2] \leq i \leq M\), all leaf nodes are at the same level
How do you search for a point? Like, point 12...
How do we insert points in the R-Tree?

Let’s try with the 15th point...

How does the adjusted R-Tree look like?

Traverse tree from root to leaf node where 15 will live. Then, adjust the dr up to the root.

Data space

Adjusted data space
- What about inserting the 16th point?

Data space (0)

- Are we done adjusting the data space?

Data space (1)
- $R$ has overflowed too! We need to split it...

We will create one internal node, $f$, adjust node $R$, and create a new root, $R'$, for the data space.

- How does the new R-Tree look like?
Spatial Indexing in MySQL

- First, make sure your table has a **NOT NULL** column of type geometry, `g`

```
CREATE TABLE particles (g GEOMETRY NOT NULL) ENGINE=MyISAM;
```

- Given a string, in **WKT (Well Known Text)** format, representing a point: **(1.5, -2.6)**, populate your table as follows:

```
INSERT INTO particles VALUES (GeomFromText('POINT(1.5 -2.6)'));
```

- With the column `g` ready, create an **R-Tree** – MySQL will do it for you!!

```
CREATE SPATIAL INDEX sp_index ON particles (g);
```