Scalability, from a database systems perspective

Dave Abel

Roadmap

- Scale of what?
- A case study: 2D to kd;
- Some algorithms for kd similarity joins;
- So …

Size matters (for joins)

- Number of sources;
- Number of points;
- Number of dimensions.

Let's use eAstronomy as an example.

Number of Sources

- Key issues:
  - Heterogeneity (despite standards);
  - The added sophistication of a more general solution.
- Optimisation typically flounders through inability to reliably estimate sizes of interim sets;
- But does it really matter?.

Number of points

- “massive” usually means that the data set is too large to fit in real memory;
- 10**7 seems to define “massive” in the database world;
- Usually target O(logN + k) for queries and O(NlogN + k) for joins, in disk I/O.

Number of dimensions

- Most database access methods are aimed at a single attribute/dimension. QEP deals with multiple atomic operations;
- Relatively recent interest in search and joins in high-dimensional space: data mining, image databases, complex objects.
- Surprises for the migrants from geospatial database. The curse of dimensionality (which the mathematicians have known all along).

Some simple algebra

$$N\varepsilon^d = n$$

<u>or</u>

$$\varepsilon = (n/N)^{1/d}$$

So, ε approaches 1 as d increases. The traditional approaches of restricting the search space fail.

But 2d is still interesting

Location is often significant:

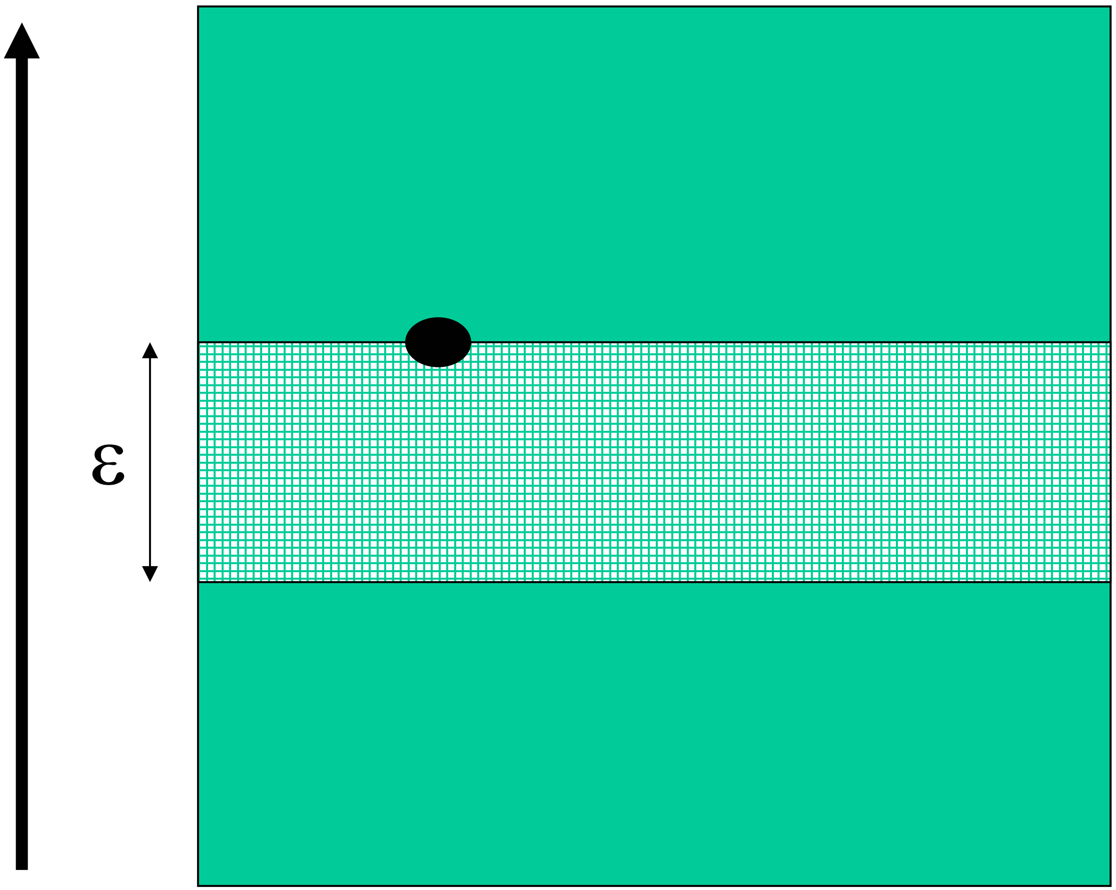
- Geospatial Information Systems (aka Geographic Information Systems) are well-established;
- Many Astronomy challenges deal with 2d databases (although the coordinate system has its tricks).

Issues of sheer size make it worthwhile to consider solutons specific to 2d.

The Sweep Algorithms for Key Operations

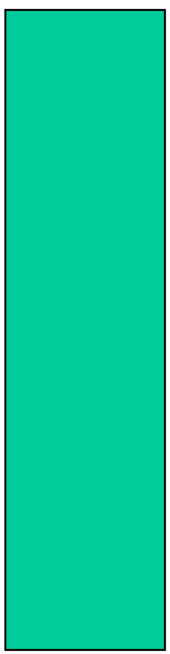
- Neighbour finding, aka fixed-radius all-neighbours, aka similarity join;
- Catalogue matching, aka fuzzy join;
- Nearest Neighbour;
- K-Nearest Neighbours.

The sweep algorithm for neighbour finding/similarity join



Active
List

Extend to kNN



1. Find an upper bound on dist to NN
2. Determine lower bounds on active list
3. Determine the NNs

WIP: preliminaries

- SDSS/Personal: 155K points, 12 seconds;
- Tycho2: 2.4M points; k = 10, 1000 seconds; k = 4, 700 seconds.

?? For large data sets. High dependence on density of points.

But it will be dismal for high-dimensional problems.

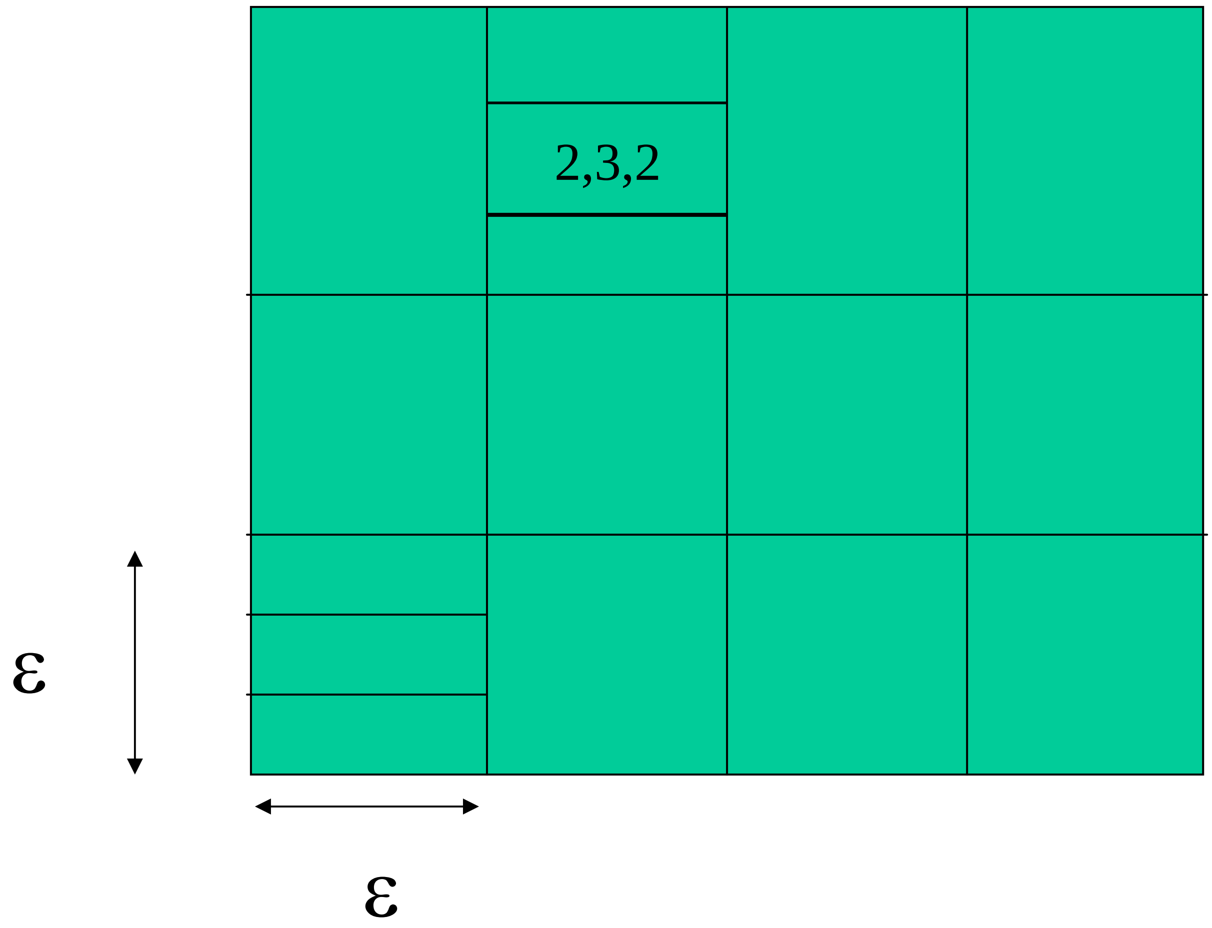
Why Dismal?

- The active list is a (d-1)-dimensional data set;
- The epsilon for the active list is high, so the list is large;
- We have reduced a join to a nasty nested-loop with a query innermost.

kD Similarity Joins & KNN

- bounding boxes (bad news after d = 8!);
- Quadtree techniques;
- Epsilon Grid Order;
- Gorder: EGO + dimensionality reduction + some tweaks on selectivity.

Epsilon Grid Order



The lessons

- Disk I/O optimisation is almost separate from CP optimisation;
- Selectivity is critical (ie avoidance of distance computations);
- High data dependence: reliance on the non-uniform distributions of ‘real’ data sets;
- How generally applicable are the results?

Best Practice?

G-order from Nat Univ Singapore:

- 0.58M points, d= 10; t =1800 seconds; S = 0.07;
- 30K points, d = 64; t = 150; S = 0.3;
- Probably about 10x better than a brute force nested loops;
- Effects of dimensionality are low.

Final Thoughts

- Where is the split between the memory-resident and disk-based families?
- Does the pure form of the problem ignore the Physics or other underlying models?
- kNN is inherently expensive. Is it a ‘classical’ problem?
- Parallelisation (with fresh approaches)?
- Are we near a plateau for similarity join and kNN with large data sets?