Multidimensional Indexes

CS157B

Chris Pollett

Feb 23, 2005.

Outline

- Applications
- Hash-like Structures

Application Needing Multidimensional Indexes

- Geographics Information Systems
  - Store maps. Might include objects like points, houses, roads, etc…Or instead of maps, might store circuit layouts…
  - Types of queries:
    - Partial match query - ask for all points matching a certain values in some specified dimensions
    - Range query - ask for all points between range of values
    - Nearest neighbour queries - closest point to a given point
    - Where am I query -- given a point, what object am I in?

More Applications

- Data Cube
  - These are often used by decision support applications where they are used to analyze information to better understand a companies operations.
    - Ex: A chain of stores might collect with each sale: the date and time, the store of purchase, the item, the color, and the size. This information could be viewed as a point in a mulitdimensional space and one might want to quickly answer aggregating queries like: number of red ties sold in each store in each month of 2005.

SQL for Multidimensional Queries

- Might have a table of Points with x and y attributes. A query might look for the closest point to (10,20):

```
SELECT * FROM Points p
   WHERE NOT EXISTS(
     SELECT * FROM Points q
     WHERE (q.x-10.0)*(q.x-10.0) + (q.y-20.0)*(q.y-20.0)
   < (p.x-10.0)*(p.x-10.0) + (p.y-20.0)*(p.y-20.0)
);
```

Another possible query: Have a table of Rectangle and ask for all tuples containing a point.

Using Conventional Indexes

To do a range query like find all points between $300<x<400$ and $500<y<600$, could have a B-tree index on x and y.

Suppose 1/10 of records satisfy the above condition on x

and 1/10 satisfy the condition on y. So 1/100 of points are in the rectangle. This is also around the typical number of records that can be held in a block.

We could do a range query on x to retrieve records pointers for x in desired range and do the same for y. We can then intersect these lists of pointers... And look up each record. Unfortunately, each record is likely to be on a different block. So we'd have to look up 1/100 *(# of records) blocks to do this. But the whole file has size (#of records) * Blocking factor = (# of records)/100 so haven't save anything.

Hash-Like Structures for Multidimensional Indexes

- Grid Files
- Partitioned hashing

Grid Files

- Split each dimension by a set of grids lines. For Points table example might have lines x= 100, 200, 300, 400, … and ys=100, 200, 300, 400.
- Index then has a bucket for each rectangle:$(x_i,x_{i+1}]$ x $(y_j,y_{j+1}]$. (Extended in a similar fashion if have more than two dimensions.) Each bucket has a set of pointers to records of Points in that rectangle.
- If a bucket is two full use overflow blocks.
- To insert just determine the rectangle the point to be inserted belongs to and add a pointer to the corresponding bucket.

Partitioned Hash Functions

- Idea: have a normal hash table on several attribute A1, A2, …; however, use a special kind of hash function such that the first k1 bits output determined by A1 only, the next k2 bits by A2 only, …