## Chapter 3.6 Tree Structure of Multi-Dimension Data

Considering range search and the nearest neighbor search of multi-dimensional data, there are four types, which includes:

* Multi-key index
* kd – tree
* quad tree
* r – tree

The former three are used in the point collection while R-tree is used to present the range collection and also can be used to present the point collection.

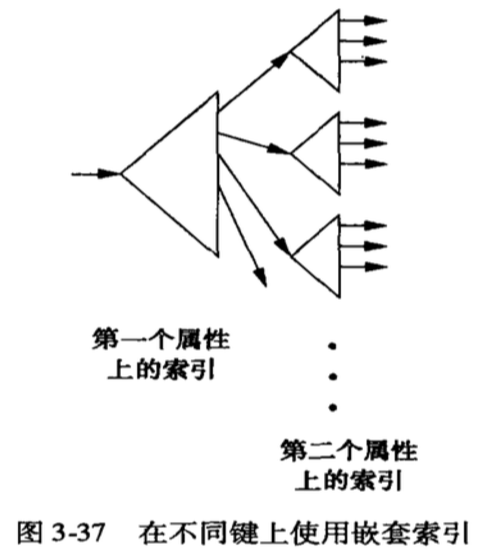
### Chapter 3.6.1 Multi-key index

Several properties are used to present the dimension of data points, the range search and the nearest neighbor search should be support. Multi-key index is a tree, points of each level is indexes of one property.

***Principles:***

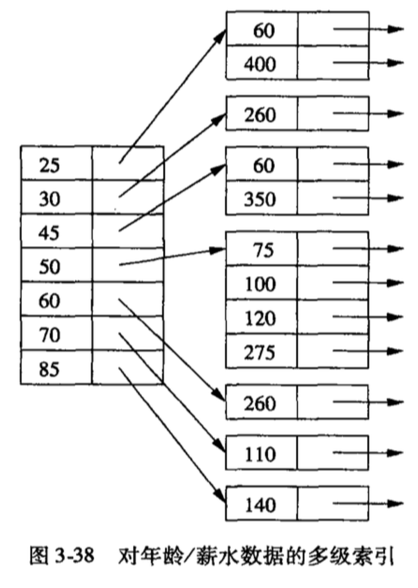
Two properties’ situation:

* The root is the first index of attribute among two attributes. The root can be any type, such as B-tree or hash list. (This means that each index in this structure can be a type of structure, either be a B-tree or hash list.)
* The index of each key relates to the pointer of another index. The first attribute should be V, while the second attribute can be a random value.



***Examples:***

As the image below shown, the first attribute is age while the second one is salary. The root is index of age, while in the right, seven indexes are supported to access other indexes.



Look from the image above, the root contains 7 indexes which represents age and in the right, it represents salary.

* Age = 25, then it relates to two salary pointers, including 60k and 400k.
* Age = 30, then it relates to one salary pointer, 260k. (one key-value pair index)
* Age = 45, then it relates to two salary pointers, including 60k and 350k.
* Age = 50, then it relates to four salary pointers, including 75k, 100k, 120k and 275k.
* Age = 60, then it relates to one salary pointer, 260k. (one key-value pair index)
* Age = 70, then it relates to one salary pointer, 110k. (one key-value pair index)
* Age = 85, then it relates to one salary pointer, 140k. (one key-value pair index)

### Chapter 3.6.2 The performance of multi-key index

***Principles:***

The performance of multi-key index, mainly considering two properties.

* ***Partial matching search:***
* Assign the first attribute, then search one of child index through root index and the index will lead to the wanting point. This process could be very efficient.

*(The place can be misleading, since each index of multi-key index is a structure, as introduced before, it can be a hash list or Btree structure.)*

* If the first attribute is unknown, then needs to search every child index, this is a time-consuming procedure.
* **Ranging search:**
* If single index does support range searching, such as B-tree or index sequence file, then multi-key index for ranging search works great.
* Use root index and the range of first attribute can find all sub-indexes that including all answers. Then using the second attribute to search each child-index.
* **Nearest ranging search:**
* This has been discussed before, just as the chapter 3.4.3.

### Chapter 3.6.3 kd – tree

kd – tree (k dimensional searching tree) is a main memory data structure which popularizes the binary searching tree to multi-dimensional data structure.