Subject: Avoiding multiple or complex WHEREs?
Posted by cgguido on Mon, 21 Feb 2011 21:31:00 GMT
View Forum Message <> Reply to Message

;Say I have an array of point IDs and times where p[0,*] are IDs and
;p[1,*] are times, something like:

```
  p=[[1,1],[1,2],[2,3]]
```

;Typically, the array has many points, many times and many gaps
;I want to be able to get a subset of 'p' that contains a range
;[id0:idn] or a list [id0,id2,id7] of IDs and Ts the most efficient
;way possible.

;Right now, I build an array 'i' with one FOR loop where i[id,t] is
the row index to 'p'
;with the correct id and t. If that combinations does not exist, then
;i[id,t]=NaN. So:

```
  i[0,0]=NaN
  i[2,0]=NaN
  i[1,*]=[NaN,0,1,NaN]
```

;So if I want all instances of particle 1 in array p I do

```
 i1=i[1,where(i[1,*] ge 0)]
 p1=p[*,i1]
```

;This gets handy when I want particles 1,3,4 and times [23:100],
;because I don't have to do where(id eq 1 OR id eq 3 OR id EQ 4....
etc.)

```
   i_cool=i[1,where(i[[1,3,4],[23:100]] ge 0)]
   p_cool=p[*,i_cool]
```

;Now, I am betting that a histogram/histobin trick might be even
;better... Any ideas?

;PS: This does not deal nicely with IDs or Ts that are bigger than the
;biggest contained in 'p'

Subject: Re: Avoiding multiple or complex WHEREs?
Posted by cgguido on Tue, 22 Feb 2011 20:02:42 GMT
View Forum Message <> Reply to Message

On Feb 22, 10:29 am, Paulo Penteado <pp.pente...@gmail.com> wrote:

> On Feb 21, 6:55 pm, Gianguido Cianci <gianguido.cia...@gmail.com>
> wrote:
>
>> Yeah :-/ I guess I am trying to build something like a HASH with two
>> keys for each value... but I could be going about this completely
>> backwards!
>
> It depends on how you are going to use it, but this problems suggests
> to me making a class that overloaded the brackets, to search for what
> you want, and return !null if not found. Internally it could use
> hashes, lists or arrays (of values or pointer arrays). The better
> choice depends on things like the number of points and whether they
> form a grid in ID, whether Ts are integers (I am guessing that
> particle IDs are). That way it would be usable doing things like
>
> p_cool=p[[1,3,4],23:100]

Wow. I did a little object oriented programming a long time ago, in java, but never with IDL. DIdn't realize that you could overload even the brackets operators. That's really cool! Not sure exactly how this would work though, seen as in reality 'p' has a few more columns (x,y,z,mass,radius,T,ID) so I would want to overload the operator for the last two columns but still be able to use plot, p[0,*], p[1,*] to plot al the positions. So for columns 0:4, the brackets should not be overloaded. And since you mentioned it, ID and T are both integers, not all ids may be present at all times, there could be 10k to 100k ID-Time pairs in each set.

Thanks for your help Paulo!

Gianguido