Test Results: Pseudo Array Table Structure Versus Single Value per Row Table Structure[1]

October 16, 2006

1.0 Background

This past summer more than one RFC has made comments about the structure of the RFC Archive DB pecrsep table. This table has a pseudo-array structure and stores data at 15-minute interval, i.e. 96 values per row for a given station and date. The comments have been about how difficult it is to query this table and the fact that data at less than 15-minute interval does not always fit the time interval of the table and/or that not all data reported is stored.

2.0 Purpose

The purpose of this test was to look into the simplest possible change in table structure without making a major change in the original design philosophy of the RFC Archive database. The new table structure proposed is a single value per row structure, and the table shall be called peirsep. A single value per row structure would address the aforementioned comments. This test will test query performance only. Compatible queries will be run on both the pecrsep and peirsep tables. Even with this simplest of changes to a single table, preliminary review indicates that 10 RAX applications would be impacted.

3.0 Test System, ax2-nhdr

The test system, ax2-nhdr has the following hardware:

- Dedicated system, Rack mounted
- Intel Xeon 2.4GHz/400MHz
- 2 - 512MB PC2100 CL2.5 ECC DDR SDRAM RDIMM
- Ultra 320, ServeRAID-5i SCSI Controller (single channel)
- Six 73.4GB 10K rpm Ultra160 SCSI HS
- 10/100/1000 Port Ethernet Server Adapter
- Tape drive - 40/80GB DLTVS HH Int. SCSI Drive (Half-High) and Ultra 160 PCI Adapter (required for Tape device when using ServeRAID5i

[1] Report written by A. Juliann Meyer, RAXUM Team Leader, Sr. Hydrologist – Data Systems, Missouri Basin River Forecast Center, Pleasant Hill, MO

- DVD Drive/Recorder - DVR-A04 Pioneer DVR (4.7gb)

The ax2-nhdr system has the following for operating system and RDMS:

RDBMS: Postgres versions 7.4.8

OS: Red Hat Enterprise 4.0 u4

4.0 Data for the Test

For the pecrsep table the data currently residing in ax2-nhdr in the test database adb_ob7krf will be used. For the peirsep table, a set of scripts were written that unloaded all data in the adb_ob7krf pecrsep table and reformatted it to single value per row load files. Missings (-9999.0) were eliminated at the time of the unload.

The end result is that for this test, table pecrsep (pseudo array) has 12,047,050 rows and table peirsep (single value per row) has 158,834,736 rows. The tables have similar primary keys and both have an index on the obstime column. In addition, vacuum analyze was performed before the testing started.

5.0 Test Conditions

User oper's cron was shut off when tests scripts against both tables were run. Both shefdecoders (raw and processed) were shut off when the test script for the pseudo array table was run so as not to alter the number of rows in the pecrsep table. However, the shefdecoders were sometimes running when the test script for the single value per row table was run (the shefdecoders do not feed data into this table). The test scripts were started late in the day and when possible run over weekends so to minimize impact on HL programmers who might be using ax2-nhdr system.

6.0 Test Queries

Five queries were used for the test on the pecrsep table. These were:

query 1

```
select * from pecrsep where lid='KCDM7' and pe1='H' and pe2='G'
order by obstime;
```

query 2

```
select * from pecrsep where lid='SSCN1' and pe1='H' and pe2='G'
and obstime between '2000-01-01' and '2000-12-31'
order by obstime;
```

query 3

```
select pecrsep.lid,lat,lon,z1200 from pecrsep,location
where pecrsep.lid=location.lid and sed is NULL
and pe1='T' and pe2='A' and dur='I' where obstime='2006-07-04';
```

query 4

```
select pecrsep.lid, obstime, z1200 from pecrsep,rivercrit
where pecrsep.pe1='H' and pecrsep.pe2='G' and
(obstime >= '1993-06-01' and obstime < '1993-09-01')
and pecrsep.lid=rivercrit.lid and pecrsep.pe1=rivercrit.pe1
and pecrsep.pe2=rivercrit.pe2 and flood > 0.0 and z1200 > flood
order by obstime;
```

query 5

```
select pecrsep.lid, obstime, z1200 from pecrsep,rivercrit
where pecrsep.pe1='H' and pecrsep.pe2='G' and
(obstime >= '2006-04-01' and obstime < '2006-07-01')
and pecrsep.lid=rivercrit.lid and pecrsep.pe1=rivercrit.pe1
and pecrsep.pe2=rivercrit.pe2 and modflood > 0.0 and z1200 > modflood
order by obstime;
```

Six queries were used for the test on the peirsep table. These were:

query 1

```
select * from peirsep where lid='KCDM7' and pe1='H' and pe2='G'
order by obstime;
```

query 2

```
select * from peirsep where lid='SSCN1' and pe1='H' and pe2='G'
and obstime between '2000-01-01 00:00:00' and '2000-12-31 23:59:59'
order by obstime;
```

query 3

```
select peirsep.lid,lat,lon,datavalue from peirsep,location
where peirsep.lid=location.lid and sed is NULL
and pe1='T' and pe2='A' and dur='I' and obstime='2006-07-04 12:00:00';
```

query 4

```
select peirsep.lid, obstime, datavalue from peirsep,rivercrit
where peirsep.pe1='H' and peirsep.pe2='G' and
(obstime >= '1993-06-01 00:00:00' and obstime < '1993-09-01 23:59:59')
and to_char(obstime,'HHMI')='1200'
and peirsep.lid=rivercrit.lid and peirsep.pe1=rivercrit.pe1
and peirsep.pe2=rivercrit.pe2 and flood > 0.0 and datavalue > flood
order by obstime;
```

query 5

```
select peirsep.lid, obstime, datavalue from peirsep,rivercrit
where peirsep.pe1='H' and peirsep.pe2='G' and
(obstime >= '2006-04-01 00:00:00' and obstime < '2006-07-01 23:59:59')
and to_char(obstime,'HHMI')='1200'
and peirsep.lid=rivercrit.lid and peirsep.pe1=rivercrit.pe1
and peirsep.pe2=rivercrit.pe2 and modflood > 0.0 and datavalue > modflood
order by obstime;
```

query 6

```
select peirsep.lid, obstime, datavalue from peirsep,rivercrit
where peirsep.pe1='H' and peirsep.pe2='G' and
(obstime >= '1993-06-01 00:00:00’ and obstime < '1993-09-01 23:59:59')
and peirsep.lid=rivercrit.lid and peirsep.pe1=rivercrit.pe1
and peirsep.pe2=rivercrit.pe2 and flood > 0.0 and datavalue > flood
order by obstime;
```

7.0 Test Results

The test scripts were designed to cycle through the queries multiple times so that average run times could be computed. After trying several times to run the 6 queries on the single value per row table just once, and failing to complete them even once in 3 days, the test script for the single value per row table was broken up into 4 scripts. Queries 1 thru 3 were put in one script and queries 4, 5 and 6 into their own scripts. The single test script for queries 1 thru 3 cycled through each query multiple times so that average run times could be computed. For the remaining 3 queries, an attempt was made to run each script once. If the script had not completed in 3 days, then it was terminated. Scripts were modified by dropping the “order by” phrase to see if simplifying the query would allow it to complete in less than 3 days. The following tables summarize the test results.

query #	table pecrsep, pseudo array Number of Rows Returned	table peirsep, single value per row Number of Rows Returned
1	12,544	137,697
2	730	15,327
3	109	109
4	2079	2079
5	0	0
6	NA	19,523

query #	table pecrsep, pseudo array average runtime (secs)	table peirsep, single value per row Average runtime (secs)
1	36.6	878.1
2	26.2	729.8
3	0.9	2.4
4$	15.3	232,469
5#	180.2	> 3 days
6*	~17,299	217,611

Notes:

$ - For the single value per row table, could not get the query to complete after 3 days with the “order by” on it. This is the time it took to complete just once without “order by” phrase.

# - For the single value per row table, query 5 was still running after 3 days and was cancelled

* - 1) For the pseudo array equivalent of query 6, the user would run query 5 for each of the 96 data value columns.
2) For the single value per row table, could not get the query to complete after 3 days with the “order by” on it. This is the time it took to complete just once without “order by” phrase.