Rebuilding a Nameserver Database

In 24 Easy Steps

December 13, 1988

by Steven Dorner

Computing Services Office

University of Illinois at Urbana-Champaign

Introduction

At the beginning of each semester, it is necessary to rebuild the database used by the CCSO Nameserver. The purpose of this annual ritual is to add to the database students and staff members who have joined the University since the last database was installed, and also to remove those who have left.

The process could in theory be done on a running database through use of the Nameserver add, delete, and change commands. This approach has several drawbacks: due to indexing demands, it is slow; ph performance suffers tremendously during the process; deleted entries are only marked as deleted, not removed from the database. In order to avoid these things, rebuilding of the nameserver database is done by first dumping the contents of the database into ASCII files, then combining these files with files produced by reading the tapes supplied by AISS.

This method is not without its drawbacks. It takes a long time, it involves many steps, the nameserver database has to be locked throughout most of the process, and it takes quite a bit of disk space. On the positive side, the steps themselves are usually fairly simple, and, since the build is taking place separately from the installed database, it can be done on any convenient machine with lots of processor and disk space.

Overview

The process begins with three databases; the extant Nameserver database, the Staff directory tape, and the Student directory tape:

Figure 1. The Starting Point

The extant database is locked, and three sets of data are extracted from it; the extant students, the extant staff, and other entries:

Figure 2. Steps 7-11

Then, the Staff tape and the Extant Staff are merged, as are the Student tape and the Extant Students. During this merge, students or staff members appearing only in the extant database, and not on the tapes, are deleted.

Figure 3. Steps 1-6 and 12-13

The Staff and Student databases are now merged; this is to avoid duplicating entries for students who happen to be employees of the University—these students will appear on both the Staff and Student tapes.

Figure 4. Steps 14-16

Now, the resulting data is made into a Nameserver database, and the miscellaneous data taken from the old database is added into the new database by Nameserver add commands:

Figure 5. Steps 17-24

The new database is ready for use by the Nameserver.

Introduction to the Detailed Description

As complicated as the above description is, it leaves out many steps and details. The rest of this document will explain in detail everything that is involved in the creation of a new Nameserver database. A few things that will help you follow the discussion are:

- The process involves many temporary files. These files follow a distinct naming scheme. The prefix "**f" means the data pertains to staff; the prefix "s" means the data pertains to students. The prefix "sf" means the data is student and staff data combined. A postfix or infix "tape" means the data**

came from one of the directory tapes. A postfix or infix "old" means the data came from the extant database. A postfix or infix "new" means the data will be put in the new database.

- At many stages of the process, temporary files may be omitted in favor of pipelines. This will allow the build to take place in less disk space; you stand to lose more processing time if something goes wrong, however. The frequent use of temporary files allows automatic "checkpointing" of the build process; if a step fails, you need only go back as far as the last temporary file to restart. Let your confidence be your guide...

- Most data files are tab-separated lists of fields. Each field begins with its Nameserver field number followed by a colon. When files are not in this format, a note will be made. During the build process, it is convenient to have the nameserver prod.cnf file handy for ready reference.

The 24–Way Path to Nirvana

AISS produces 1600 bpi, unlabelled tapes in ebcdic, with the data elements in fixed-width fields, one record per person, and multiple records per block. The UNIX utility dd is used to read these tapes onto disk, doing unblocking and character set conversion. *Student.tape and staff.tape should be the names of devices on which the student and staff tapes are mounted.*

1. **dd ibs=12100 cbs=121 conv=ASCII,lcase <***student.tape >s.tape.raw*
2. **dd ibs=3500 cbs=350 conv=ASCII,lcase <***staff.tape >f.tape.raw*

Now, the tapes are converted from fixed-width lines into tab-separated lines, and the proper field numbers are prepended to each field. For students only, the AISS data element that encodes class and college is expanded into the Nameserver field "curriculum". The programs s.pb and f.pb perform these marvels.

3. **s.pb <s.tape.raw >s.tape.id**
4. **f.pb <f.tape.raw >f.tape.id**

Next, the University id's in the data files are converted by ssnid into random Nameserver id's. This is done with the help of the dbm database IdDB, which remembers the mappings from University to Nameserver id's. University id's which have been encountered before will be mapped into whatever they were mapped into last time; those not appearing in the database will be assigned at random, and the choice recorded in IdDB. The resulting files are sorted on their first fields, the Nameserver id's. IdDB should be read in from tape, the programs run, and then IdDB should be written back out to tape and removed from disk. This will assure privacy of University id's.

5. **ssnid <s.tape.id | sort >s.tape**
6. **ssnid <f.tape.id | sort >f.tape**

At this point, the running Nameserver database must be made read-only by placing a line beginning with "read" in the .sta file (/nameserv/db/prod.sta on our system).

7. **echo read for database update >/nameserv/db/prod.sta**

Once the database is protected from modification, its contents should be dumped with mdump. This dumping is done into four different files; one for staff members, one for students, one for campus units, and one for other entries. Each dump may contain a different set of fields; for example, the "students" dump contains only fields that cannot be found on the student tape, whereas the "other" dump dumps all fields. In all cases, mdump outputs the "id" field first for each entry; mdump will manufacture a blank "id" field if none is present. The "other" dump is constructed to select those entries not selected by the other dumps.

8. **mdump students | sort >s.old**
9. **mdump staff | sort >f.old**
10. **mdump other | sort >other.old**
11. **mdump units >units.old**

It is now time to reconcile the old data with the new data; this is done with tmerge. The idea is twofold; to drop from the database persons who do not appear on the new tapes, and to bring along from the old

database any fields that are not found on the tapes themselves (e.g., email addresses).

Tmerge takes four arguments; the name of the file with data from the tape, the name of the file with data from the extant database, the name of the file for the merged data, and (optionally) the name of a file into which to put entries from the old database that are not going into the new database. This last argument we do not use; such entries slip quietly into oblivion.

12. tmerge s.tape s.old s.new

13. tmerge f.tape f.old f.new

Now, the stastu program is used to merge the staff and student files. For staff members who are also students, some fields will appear in each file (e.g., address). In such cases, the field from the staff file is given preference.

14. stastu f.new s.new >sf.new

It is necessary to compute Nameserver aliases for the new database. To do this, we extract the current alias (if there is no alias, we use the magic cookie "{none}") and the base name for an assigned alias (the last name and the first letter of the first name). These two items are prepended to each entry from sf.new, and the whole is sorted into reverse order.

15. aliasprepare <sf.new | sort >sf.prealias

Awk handily if slowly assigns our aliases.

16. awk −f alias.awk sf.prealias >sf.alias

We're getting close now. Use credb to create a new, empty database. The integer argument is the number of slots to use in the hash table; we use approximately 5 slots per entry.

17. credb prod 300007

Maked takes our tab-separated ASCII file and makes it into Nameserver .dir and .dov files.

18. maked prod <sf.alias

It is now time to build the index for the database. This step takes many hours (six the last time I did it, on our dual-processor Gould super-mini, in the dead of night). It is very disk-intensive.

19. makei prod

Now we make an index to the index, to facilitate wildcard searches.

20. build −s prod

This step is optional. If you have been doing the build on a machine with normal byteorder, and intend to use the database on a machine with reversed byteorder (like a VAX or 80x86), you must reverse the appropriate bytes in the database. The border program does this; it may be run on either the normal or the byte-perversed machine.

21. border prod (Optional)

It is now time to move the database into place; turn off the Nameserver completely, move the files into place, and let 'er rip.

22. echo stop installing new database... >/nameserv/db/prod.sta

23. mv prod.* /nameserv/db

24. rm /nameserv/db/prod.sta

Now that the database is up and running, we use normal Nameserver commands (via qi) to add in the entries from units.old and anything interesting from other.old. We then hope that the next semester won't come for a long time...

Appendix A

Data Tape Formats and Fields

Students

The student tape consists of blocks of 100 records, each record 121 bytes long. The tape is in ebcdic, and the layout is as follows:

	Start	**Stop**		**Nameserver**
Col.	Col.	Length	Description	Field
1	9	9	University id	id
10	29	20	name	name
30	47	18	street address	address
48	65	18	city	address
66	70	5	zip code	
71	77	7	telephone number	phone
78	118	41	home address	
119	121	3	class and college	curriculum

Staff

The staff tape consists of blocks of 10 records, each record 350 bytes long. The tape is in ebcdic, and the layout is as follows:

	Start	**Stop**		**Nameserver**	
Col.	Col.	Length	Description	Field	
1	9	9	University id	id	
	10	10	1 padding		
11	33	23	last name	name	
34	48	15	first name	name	
49	63	15	middle name	name	
64	78	15	spouse name		
79	153	75	job title	title	
154	178	25	employing department		department
179	203	25	office number	address	
204	228	25	office street	address	
229	244	16	office city		
245	246	2	office state		
247	251	5	office zip code		
252	254	3	office mailcode	address	
255	264	10	office phone number		phone
265	265	1	suppress home address		
266	315	50	home street address		address
316	330	15	home city	address	
331	332	2	home state		
333	337	5	home zip code		
338	338	1	suppress home phone number		

339	348	10	home phone number	phone
349	350	2	padding	