Network Noise

Volume 3, Issue 2 Terry Christie, Editor May 1998

NOTES FROM DYNCORP

Kevin Kiah, DynCorp

First of all, Charlie and I want to thank all the participants at the last CAIS conference held at LLNL. We received a lot of good suggestions and ideas for CAIS 2000. CAIS 3.5 will be released on June 15, 1998. The 1998 engineering costs will be included in the release along with the new CAIS/FIMS interface, the summary condition screen and 3 new reports to meet the deferred maintenance requirements. DynCorp has added a new programmer, Nick Stockton, to the CAIS contract. He has a BS in Management Information Systems from the Salisbury State University and is well on his way to attaining an MS in Telecommunications at the University of Maryland at University College. Nick's primary responsibility will be to design, code and distribute CAIS 2000. He will also help answer Hotline questions when needed.

OVERVIEW OF THE PROPOSED CHANGES TO SITE-CAIS

Jesse Oak, Parsons Brinkerhoff

The following is a list of items discussed at the DOE CAIS User's meeting held the week of March 30 through April 3, 1998, at Lawrence Livermore National Laboratory. The breakdown consists of taking out the trash items, possible enhancements, and other items discussed at the meeting. The plan is to release a version of Site-CAIS around June 15, 1998 to include the Summary Condition Screen, the enhanced FIMS/CAIS Interface, Deferred Maintenance Reporting, and the 1998 Cost followed by a new Site-CAIS version in the fall of 1999.

The following items have been identified for removal from the Site-CAIS system:
The Download module.
The Upload module.
The project Promote module.
The Routes module.
Delete the old FIMS Interface module.
Delete all FIMS reference data on the Site Asset Screen.
Delete all seismic references in the system such as on the Asset Dimension Screen.
Remove the Equipment Archiving Module.
Delete the Survey Equipment linked in the Inspection module.
Remove the field "Created by Upload" found on the Detail 1 tab of the Asset screens.
Delete all "Effective" and "Ineffective" dates that are not linked to costing or required for system operation.
Remove the Work Order tab from the system.

The following items were discussed at the User's conference:
Look at making the Database View Generator user friendly and effective.
Look at making the Data Analysis Packet user friendly.
The CAIS/FIMS Interface module to be easier to use and include the creation of the CAIS/FIMS relationship, selection of the data to transfer, and the creation of the output file.
Look at making the cost adders easy to see and use.
Incorporate Lawrence Livermore cost adders into the existing system.
Look at creating site defined cost multipliers and adders to the system.
Look at reworking the Equipment screens to

streamline data entry.
Look at reworking the Project module to make it user friendly and eliminate the need to Post and Unpost projects.
Change survey inspection units default to “in condition”.
Look at changing the inspection units out of condition only at the inspection unit level.
Look at making the Reports module easier to use and to view.
Look at creating “memory” for data input into select fields to avoid re-entry for duplication data.
Look at changing surveys to be virtual containers for the inspection units creating the ability to track deficiencies cradle to grave and place IU’s into multiple surveys.
Keep the Inquiries module but rework the reports to be user friendly.
Change the login procedure to something different then user “CAIS” and password “CAIS” for better system security.
Look at adding the deficiency sequence numbers automatically generated by Oracle to the reports for tracking purposes.
Look at reworking and simplifying the report module.
Look at reworking the Archive module to save existing survey data as read-only to freeze costs for review only.
Look at creating the ability for sites to build their own Project and Inspection Unit tags for sorting.
Revise system to be true 32-bit and user friendly.
Look at creating a generic import and export feature in the system.
Look at revising the system tables to give the sites the ability to easily modify, add, and delete the table data. Certain tables such as WBS, Component, Type, and Deficiency will remain locked to the sites for consistency.
Digital photographs will not be incorporated into the system. Sites with photos will look at using Darrell’s Web based program.
Look at reworking the Master Equipment List module to be easier to use and understand.
Look at creating the ability to send data to MS Excel, Access, and Word, etc. for site customization and reporting.
Look at developing an easy system for performing tracking and trending of collected deficiencies.
Look at reworking the Data Analysis Packet and Project modules to be simple easy to use processes.
Study how to rework the program to be intuitive and flexible.

OAK RIDGE NATIONAL LAB NOTES

Charles Lamb, Program Coordinator ORNL

The ORNL CAS Program is still alive and kicking. We anticipate a lot of changes in the coming months. As of April 1, 1998 a new sub-contractor has moved on site. Bechtel/Jacobs has assumed over ½ million square feet of ORNL floor space and many other site projects. We are anxious to begin a new relationship. We still don’t know exactly how the CAS Program will be used. Our inspectors just completed a requested inspection of the Tower Shielding Facility to transition this facility to Bechtel/Jacobs. Our goals are to continue the cutting edge technology, that ORNL is noted for, and the CAS Program is no exception. We are striving hard to be the leader in the information business. We have plans to increase the amount of information on our web site, we are looking into new software that will enable us to better communicate and report our findings. It is up to each individual person in the CAS network to make our program the best that it can be.

WHAT IS LYNX?

Terry Christie, Oak Ridge National Laboratory

LYNX is a new type of software product that incorporates the new state-of-the-art digital color cameras into a complete Visual Documenting System. These cameras do not require film or processing. They store their images on memory chips instead of film. The digital photos are transferred directly to the computer’s hard disk.

The LYNX Visual Documenting System provides all of the tools necessary to download the photos

from the camera, compress them, secure them from tampering, file them in a high speed database with attached notes, and link them to digital reports. LYNX can instantly send its digital photos and reports to other LYNX Systems via modem, floppy disk, the Internet, WAN, LAN, etc.
How does LYNX differ from regular photo software? Normal consumer photo software simply downloads individual images from digital cameras as individual graphic files, then lets the user manipulate the images to brighten, crop, enlarge, etc. In contrast, LYNX uses the photos as part of a complete, secure Visual Documenting System that can transfer its photos and reports between sites…no matter how far apart they are!

LYNX also performs the following unique functions:

- Automatically downloads digital photos directly from a wide variety of digital cameras including Kodak, Logitech, Dycam, and others.
- Compresses photos automatically. Just one floppy disk will hold an average of 50 compressed photos!
- Automatically adds date and time stamps to each photo. These date and time stamps cannot be changed.
- Archive the original photos in a high-speed, secure database, where they cannot be altered. Copies of the secure photo can be exported or placed in other documents (word processor, slide show program, etc).
- Attaches an unlimited number of keywords to each photo allowing them to be found instantly at any time.
- Attaches photos to schedule activities automatically linking them to Primavera Project Planner (P3), SureTrak or Microsoft Project.
- Attaches a permanent description (up to 5,000 characters) to each photo.
- Files the photos to user defined jobs, and links the photos to user defined reports in each job. Each job can contain an unlimited number of photos.
- Imports other images such as scanned photos, still frames from VCR’s, faxes, etc. and allows them to be filed to the jobs and attached to LYNX’s reports.
- Exports and transfers the photo and reports using the built-in file transfer system. The secure LYNX transfer packets can be sent from site to site using LYNX’s built in telecommunications software, diskette, the Internet, wide area network, e-mail, Remote Access Service, or any other type network or modem connection.

You can access more information on this system via: *http://www.trfsys.com/web/lynx/index.htm*

PANTEX CAS PROGRAM?

Bob Von Eschen, Pantex

- “Doesn’t exist anymore and is not funded in the next budget!” per the Pantex Plant Manager. However only the name has been changed, to “Facility Inspection”. Currently, Jerry and I continue the assessments and report the deficiencies to an ever-increasing clientele.
- A proposal has been presented to develop an “Inspection Management Group” which would combine the inspection forces of CAS and construction, with training to the CAS Program (Construction Standard Institute) Manuals.
- No official instruction has been received to proceed with the “Deferred Maintenance Cost” analysis, to the contrary our instructions are to remain on “hold”.
- Official word has been issued that Pantex is to downsize by about 390 persons by the end of March 1999. Even though most of the reduction is expected by attrition, some will undoubtedly be let go. So far our group has not been targeted, so plans are to see you at the next network meeting.

SIZING YOUR CAIS DATABASE

Robert Hampton, Los Alamos National Laboratory

One of the biggest problems I've had as a CAIS Administrator has been trying to figure out how much space the CAIS Database (any DB) is going to take up now and in the future. I recently read several articles about sizing database objects (see References). Probably the most important aspect of this topic is figuring out how big your tables and indexes are and how big they might get. Your Oracle Database Administrator's Guide will give you guidelines but it is written in Greek by geeks. The listings that I have provided (plagiarized from before mentioned articles) will help automatically size your tables and indexes and allow you to keep this info in your database. Not only can these scripts help you keep track of your current database info (like num_rows, pct_free, and pct_used) but you should be able to keep up with future growth trends.

The first thing that needs to be done is to capture all of the internal variables that Oracle tracks. The query in ***Listing 1***. Will do that assuming a default of 1 for INITRANS (initial transactions?) and 20 for PCTFREE (20% free).

LISTING 1.

```
# newspace.sql
# show key values used in space calculations
set linesize 132
col db_block_size for a15
select (a.db_block_size-b.kcbh-c.ub4-d.ktbbh-f.kdbh
          -((j.ini_trans-1)*e.ktbit)) hsize,
       CEIL((a.db_block_size-b.kcbh-c.ub4-d.ktbbh-f.kdbh
          -((j.ini_trans-1)*e.ktbit))*(1-j.pct_free/100))- g.kdbt  availspace,
   a.db_block_size,b.kcbh,c.ub4,d.ktbbh,e.ktbit,f.kdbh,g.kdbt, h.ub1,i.sb2,j.ini_trans,j.pct_free
from
(select "NAME" ,"VALUE" db_block_size from v$parameter where    name = 'db_block_size' ) a,
(select "TYPE" ,"TYPE_SIZE"  kcbh         from v$TYPE_SIZE where    "TYPE" = 'KCBH' ) b,
(select "TYPE" ,"TYPE_SIZE"  ub4          from v$TYPE_SIZE where    "TYPE" = 'UB4'  ) c,
(select "TYPE" ,"TYPE_SIZE"  ktbbh        from v$TYPE_SIZE where    "TYPE" = 'KTBBH') d,
(select "TYPE" ,"TYPE_SIZE"  ktbit        from v$TYPE_SIZE where    "TYPE" = 'KTBIT') e,
(select "TYPE" ,"TYPE_SIZE"  kdbh         from v$TYPE_SIZE where    "TYPE" = 'KDBH' ) f,
(select decode(g.kdbt,0,c.ub4,g.kdbt) kdbt from
(select "TYPE" ,"TYPE_SIZE"  ub4 from v$TYPE_SIZE where     "TYPE" = 'UB4'  ) c,
(select "TYPE" ,decode("TYPE_SIZE",NULL,0,"TYPE_SIZE") kdbt      from v$TYPE_SIZE ,dual
    where "TYPE"(+) = 'KDBT' and dummy = "TYPE"(+) ) g    ) g,
(select "TYPE" ,"TYPE_SIZE"  ub1          from v$TYPE_SIZE where    "TYPE" = 'UB1'  ) h,
(select "TYPE" ,"TYPE_SIZE"  sb2          from v$TYPE_SIZE where    "TYPE" = 'SB2'  ) i,
(select 'INITRANS' ,1   ini_trans,
        'PCTFREE'  ,20  pct_free          from dual           ) j
/
```

Note: Use this newspace.sql query script to identify the internal variables in your Oracle database. (If you are using a version of the database prior to Oracle7 Release 7.3.4, change all instances of column TYPE_SIZE to SIZE for table v$type_size.)

Listing 1. should produce output like ***Table 1***.

TABLE 1.

HSIZE	AVAIL SPACE	DR_BLOCK_ SIZE	KCBH	UB4	KTBBH	KTBIT	KDBH	KDBT	UB1	SB21	N1_TRANS	PCT_ FREE
8106	6481	8192	20	4	48	24	14	4	1	2	1	20

Use this information to check current utilization. You must know the current database block size (db_block_size), how muchspace is left after Oracle allocates header space (HSIZE) and the big one how much of the block can be used for data (AVAILSPACE).
Next, use ***Listing 2***. To create sz_tables and sz_indexes tables to store all the critical sizing info.

LISTING 2.

```
create table sz_tables  create table sz_indexes
(       (
owner          varchar2(30) not null,   owner                  varchar2(30) not null,
table_name     varchar2(30) not null,   index_name      varchar2(30) not null,
tablespace_name varchar2(30),                   table_owner     varchar2(30) not null,
avg_row_len     number       default 1, table_name             varchar2(30) not null,
ini_trans       number       default 1,         tablespace_name varchar2(30),
pct_free        number       default 20,        ini_trans       number,
pct_used        number       default 80,        pct_free        number,
num_rows        number                          index_entry_size        number,
)                                               uniqueness              varchar2(9)
 /                              )
                                /
```

Use ***Listing 3***. to insert the index information (it can be interesting since you need to sum column lengths and still keep track of the individual column).

LISTING 3.

```
insert into sz_indexes (
select z0.owner,
       z0.index_name,
       z0.table_owner,
       z0.table_name,
       z0.tablespace_name,
       z0.ini_trans,
       z0.pct_free,
       sum(decode(sign(z1.column_length-126),1,z1.column_
       length+2,z1.column_length+1)),
       z0.uniqueness
  from dba_indexes z0, dba_ind_columns z1
 where z0.owner        = z1.index_owner
   and z0.index_name   = z1.index_name
   and z0.table_owner  = z1.table_owner
   and z0.table_name   = z1.table_name
   and z0.owner        = '<some_owner>'
 group by z0.owner,z0.index_name,z0.table_owner,z0.table_name,
         z0.tablespace_name,z0.ini_trans,z0.pct_free,z0.uniqueness)
/
```

The table info is much easier, see ***Example 1***:

EXAMPLE 1.

```
insert into sz_tables (select owner,table_name,",avg_row_len,
      ini_trans,pct_free,pct_used,num_rows
                from dba_tables where owner = '<some_owner>')
```

If you are simulating this exercise (i.e., you do not have real objects but are using data from users and analysts) then create a flat file like ***Example 2***.

EXAMPLE 2.

```
file.lst
<some_owner>      Table01   50
<some_owner>      Table02   100000
.           .           .
.           .           .
<some_owner>      Table0n   13000
```

Example 3. shows a Unix script (Bourne Shell) to generate insert/update statements.

EXAMPLE 3.

```
# crszsql.sh
more $1 | while read LINE
do
   OWNER='echo $LINE | awk '{print $1}' - | tr a-z A-Z'
   TABLE_NAME='echo $LINE | awk '{print $2}' - | tr a-z A-Z'
   NUM_ROWS='echo $LINE | awk "{print $3}"
   NEW_LINE="insert into sz_tables(owner,table_name,num_rows) \
   values ("$OWNER","$TABLE_NAME","$NUM_ROWS");"'
   echo $NEW_LINE
done
```

Pipe that info into another flat file (in Unix - sorry you NT guys will have to figure out your own stuff) like in ***Example 4***.

EXAMPLE 4.

```
<unix prompt> crszsql.sh  file.lst > crszsql.sql
```

All you are trying to do here is get the initial info into the tables (sz_tables, sz_indexes).
Use ***Listing 4.*** to figure out estimated table sizes. The script generates Catalog numbers (from dba_tables) and

Analyzed numbers (from dba_tables and sz_tables).

LISTING 4.

```
--# tblrowsz.sql
--# use to look at table row sizes and rows per block
set echo off
set linesize 132
set pagesize 20
col owner            for a08      head 'Owner'
col table_name         for a30      head 'Table Name'
col cat_rowsize       for 99999    head 'Catalog|Rowsize'
col cat_rows_per_block for 999999   head 'Catalog|Rows|per|Block'
col anl_rowsize       for 99999    head 'Analyzed|Rowsize'
col anl_rows_per_block for 999999   head 'Analyzed|Rows|per|Block'
col anl_num_rows       for 999999999 head 'Analyzed|Number|of Rows'
col anl_blocks        for 999999   head 'Analyzed|Blocks|Needed'
col anl_space         for 999999   head 'Analyzed|Space|Needed|(Meg)'
col cat_space         for 999999   head 'Catalog|Space|Needed|(Meg)
col tbl_pct_free      for 999      head 'Pct|Free'
col tbl_pct_used       for 999      head 'Pct|Used'
select y.owner, y.table_name,
      (3*z.ub1)+y.rowsize                           cat_rowsize,
      FLOOR(availspace/((3*z.ub1)+y.rowsize))              cat_rows_per_block,
      (3*z.ub1)+avg_row_len                           anl_rowsize,
      FLOOR(availspace/((3*z.ub1)+avg_row_len))             anl_rows_per_block,
      num_rows                                  anl_num_rows,
      FLOOR(num_rows/(FLOOR(availspace/((3*z.ub1)+avg_row_len))))     anl_blocks,
      FLOOR((db_block_size*FLOOR(num_rows/(FLOOR(availspace/
              ((3*z.ub1)+avg_row_len)))))/1024/1024)        anl_space,
      FLOOR((db_block_size*FLOOR(num_rows/(FLOOR(availspace/
              ((3*z.ub1)+((3*z.ub1)+y.rowsize))))))/1024/1024) cat_space,
      tbl_pct_free,
      tbl_pct_used
from
(select owner,table_name,
      sum(decode(sign(data_length-249),1,data_length+3,data_length+1)) rowsize
  from sys.dba_tab_columns zz
 group by owner,table_name) y,
(select j.owner             owner,
       j.table_name         table_name,
       j.avg_row_len         avg_row_len,
       j.num_rows           num_rows,
       j.pct_free          tbl_pct_free,
       j.pct_used          tbl_pct_used,
       (a.db_block_size - b.kcbh - c.ub4 - d.ktbbh - f.kdbh - ((ini_trans-1)*e.ktbit)) hsize,
       CEIL((a.db_block_size - b.kcbh - c.ub4 - d.ktbbh - f.kdbh - ((ini_trans-1)*e.ktbit)) *
 ( 1 - pct_free  /  100 )) - g.kdbt availspace,
a.db_block_size,b.kcbh,c.ub4,d.ktbbh,e.ktbit,f.kdbh,g.kdbt,h.ub1,i.sb2,ini_trans,pct_free from
 (select "NAME" ,"VALUE" db_block_size from v$parameter where name = 'db_block_size' ) a,
 (select "TYPE" ,"TYPE_SIZE"  kcbh          from v$TYPE_SIZE where "TYPE" = 'KCBH' ) b,
 (select "TYPE" ,"TYPE_SIZE"  ub4           from v$TYPE_SIZE where "TYPE" = 'UB4'  ) c,
 (select "TYPE" ,"TYPE_SIZE"  ktbbh         from v$TYPE_SIZE where "TYPE" = 'KTBBH') d,
 (select "TYPE" ,"TYPE_SIZE"  ktbit         from v$TYPE_SIZE where "TYPE" = 'KTBIT') e,
 (select "TYPE" ,"TYPE_SIZE"  kdbh          from v$TYPE_SIZE where "TYPE" = 'KDBH' ) f,
 (select decode(gg.kdbt,0,c.ub4,gg.kdbt) kdbt from
 (select "TYPE" ,"TYPE_SIZE"  ub4 from v$TYPE_SIZE where "TYPE" = 'UB4'          ) c,
 (select "TYPE" ,decode("TYPE_SIZE",NULL,0,"TYPE_SIZE") kdbt from v$TYPE_SIZE ,dual
    where "TYPE"(+) = 'KDBT' and dummy = "TYPE"(+)                              ) gg ) g,
 (select "TYPE" ,"TYPE_SIZE"  ub1           from v$TYPE_SIZE where "TYPE" = 'UB1'  ) h,
 (select "TYPE" ,"TYPE_SIZE"  sb2           from v$TYPE_SIZE where "TYPE" = 'SB2'  ) i,
 (select owner,table_name,
        AVG_ROW_LEN avg_row_len,
        INI_TRANS   ini_trans,
        PCT_FREE    pct_free,
        PCT_USED    pct_used,
        NUM_ROWS    num_rows
from &dba_tables jj where jj.table_name like upper('&table_name%')         ) j ) z where y.owner(+)       =
z.owner and y.table_name(+) = z.table_name
/
undef table_name
undef dba_table
spool off
```

The info in ***Table 2***. will help determine the report output and how you got the data.

TABLE 2.

Column	**dba_indexes**	**sz_indexes**
Catalog Rowsize	The sum of column lengths from dba_tab_columns	Same as for dba_tables if table exists in catalog
Catalog Rows per Block	The number of rows, given the catalog rowsize, that will fit into a block	Same as for dba_tables if table exists in catalog
Analyzed Rowsize	The avg_row_len from dba_tables after an analyze has been done on the table	The avg_row_len from sz_tables
Analyzed Rows per block	The number of rows, given the analyzed avg_row_len, that will fit into a block	The number of rows, given the avg_row_len from sz_tables, that will fit into a block
Analyzed Number of Rows	The value of num_rows from dba_tables after an analyze has been done on the table	The num_rows from sz_tables

Analyzed Blocks Needed	The number of data blocks needed, given the analyzed rows per block and analyzed number of rows	The number of data blocks needed, given the rows per block and the number of rows from the sz_tables data
Analyzed Space Needed (megabytes)	Total space needed, determined by dba_block_ size, dba_tables, and analyzed blocks needed	Total space needed, determined by dba_block_ size, sz_tables, and analyzed blocks needed
Catalog Space Needed (megabytes)	Total space needed determined by catalog blocks needed (not in report) and analyzed num_rows from dba_tables	Total space needed determined by catalog blocks needed (not in report) and num_rows from sz_tables
Pct Free	The percentage free from dba_tables	The percentage free from sz_tables
Pct Used	The percentage used from dba_tables	The percentage used from sz_tables

The tblrowsz.sql script should give output like ***Table 3.***

TABLE 3.

Owner	**Table Name**	**Catalog Rowsize**	**Catalog Rows per Block**	**Analyzed Rowsize**	**Analyzed Rows per Block**	**Analyzed Number of Rows**	**Analyzed Blocks Needed**	**Analyzed Space Needed (Meg)**	**Catalog Space Needed (Meg)**	**Pct Free**	**Pct_ Used**
<OWNER>	<TABLE_01>	87	83	61	119	15963	134	1	1	10	60
<OWNER>	<TABLE_02>	841	7	267	24	1958100	81587	637	2185	20	60
<OWNER>	<TABLE_03>	1060	6	425	15	4220403	281360	2198	5495	20	40
<OWNER>	<TABLE_04>	264	27	139	52	46322	890	6	13	10	60
<OWNER>	<TABLE_05>	76	95	71	102	44200	433	3	3	10	60

Listing 5. will calculate your index sizes.

LISTING 5.

```
--# ixrowsz.sql
set linesize 132
col owner                for a10       head 'Owner'
col index_name           for a30       head 'Index Name'
col cat_index_entry_size for 9999990   head 'Catalog|Index|Entry|Size'
col cat_blocks           for 99999990  head 'Catalog|Blocks|for|Index'
col cat_num_rows         for 999999999 head 'Catalog|Number|of Rows'
col cat_space            for 9999990   head 'Catalog|Space|Needed|(Meg)'
col sz_index_entry_size  for 9999990   head 'Sizing|Index|Entry|Size'col sz_blocks           for 99999990  head
'Sizing|Blocks|for|Index'
col sz_num_rows          for 999999999 head 'Sizing|Number|of Rows'
col sz_space             for 9999990   head 'Sizing|Space|Needed|(Meg)'
select y.owner,y.index_name,
       decode(y.uniqueness,'NONUNIQUE',1,0)+2+6+y.dba_index_entry_size cat_index_entry_size,
       1.05*((num_rows*(decode(y.uniqueness,'NONUNIQUE',1,0)+2+6+y.dba_index_entry_size)) /
            (availspace)) cat_blocks,
       num_rows cat_num_rows,
       (1.05*((num_rows*(decode(y.uniqueness,'NONUNIQUE',1,0)+2+6+y.dba_index_entry_size)) /
            (availspace)))*db_block_size/1024/1024 cat_space,
       decode(y.uniqueness,'NONUNIQUE',1,0)+2+6+y.sz_index_entry_size sz_index_entry_size,
       1.05*((sz_num_rows*(decode(y.uniqueness,'NONUNIQUE',1,0)+2+6+y.sz_index_entry_size)) /
            (availspace)) sz_blocks,
       sz_num_rows sz_num_rows,
       (1.05*((sz_num_rows*(decode(y.uniqueness,'NONUNIQUE',1,0)+2+6+y.sz_index_entry_size)) /
            (availspace)))*db_block_size/1024/1024 sz_space
  from
(select z0.owner,z0.index_name,
       sum(decode(sign(column_length-126),1,column_length+2,column_length+1)) dba_index_entry_size,
       z2.index_entry_size sz_index_entry_size,
       z0.uniqueness
  from &&dba_indexes z0, dba_ind_columns z1, sz_indexes z2
where z0.owner        = z1.index_owner(+)
   and z0.index_name   = z1.index_name(+)
   and z0.table_owner  = z1.table_owner(+)
   and z0.table_name   = z1.table_name(+)
   and z0.owner        = z2.owner(+)
   and z0.index_name   = z2.index_name(+)
   and z0.table_owner  = z2.table_owner(+)
   and z0.table_name   = z2.table_name(+)
 group by z0.owner,z0.index_name,z0.uniqueness,z2.index_entry_size) y,
(select j.owner,
        j.index_name,
        j.num_rows,
        j.sz_num_rows,
        (a.db_block_size - j.block_header) -
         ((a.db_block_size - j.block_header)*(pct_free/100)) availspace,
        a.db_block_size,b.kcbh,c.ub4,d.ktbbh,e.ktbit,f.kdbh,g.kdbt,h.ub1,i.sb2,j.ini_trans,j.pct_free
from
(select "NAME" ,"VALUE" db_block_size from v$parameter where name = 'db_block_size' ) a,
 (select "TYPE" ,"TYPE_SIZE"  kcbh          from v$TYPE_SIZE where "TYPE" = 'KCBH' ) b,
 (select "TYPE" ,"TYPE_SIZE"  ub4           from v$TYPE_SIZE where "TYPE" = 'UB4'  ) c,
 (select "TYPE" ,"TYPE_SIZE"  ktbbh         from v$TYPE_SIZE where "TYPE" = 'KTBBH') d,
 (select "TYPE" ,"TYPE_SIZE"  ktbit         from v$TYPE_SIZE where "TYPE" = 'KTBIT') e,
 (select "TYPE" ,"TYPE_SIZE"  kdbh          from v$TYPE_SIZE where "TYPE" = 'KDBH' ) f,
 (select decode(g.kdbt,0,c.ub4,g.kdbt) kdbt from
   (select "TYPE" ,"TYPE_SIZE"  ub4 from v$TYPE_SIZE where "TYPE" = 'UB4'  ) c,
(select "TYPE" ,decode("TYPE_SIZE",NULL,0,"TYPE_SIZE") kdbt from v$TYPE_SIZE ,dual
    where "TYPE"(+) = 'KDBT' and dummy = "TYPE"(+) ) g                    ) g,
 (select "TYPE" ,"TYPE_SIZE"  ub1          from v$TYPE_SIZE where "TYPE" = 'UB1'  ) h,
 (select "TYPE" ,"TYPE_SIZE"  sb2          from v$TYPE_SIZE where "TYPE" = 'SB2'  ) i,
 (select j0.owner,j0.index_name,
         (113+(24*j0.INI_TRANS)) block_header,
         j0.INI_TRANS                ini_trans,
         j0.PCT_FREE                 pct_free,
         j1.NUM_ROWS                   num_rows,
         j2.NUM_ROWS                   sz_num_rows
    from &&dba_indexes j0, dba_tables j1, sz_tables j2
   where j0.index_name  like upper('&index_name%')
    and j0.table_owner = j1.owner(+)
    and j0.table_name  = j1.table_name(+)
    and j0.table_owner = j2.owner(+)
    and j0.table_name  = j2.table_name(+)                            ) j ) z
 where y.owner = z.owner
   and y.index_name = z.index_name
/
undef index_name
undef dba_indexes
```

The info in ***Table 4.*** will provide data on index sizes along with source info.

TABLE 4.

Column	**dba_indexes**	**sz_indexes**
Catalog Index Entry Size	The sum of column lengths from dba_ind_columns	Same as for dba_indexes, if index exists in the catalog
Catalog Blocks for Index	The computed number of blocks to hold index, given the catalog index-entry size, num_rows from dba_tables, and available space per block	Same as for dba_indexes, if index exists in the catalog
Catalog Number of Rows	The value of num_rows from dba_tables	Same as for dba_indexes, if index exists in the catalog
Catalog Space Needed (megabytes)	The calaculated space given the cataloged blocks for index and available space per block	Same as for dba_indexes, if index exists in the catalog
Sizing Index Entry Size	Same as sz_indexes, if entry in sz_indexes exists	Index_entry_size from sz_indexes
Sizing Number of Rows	Same as sz_indexes, if entry in sz_indexes exists	The value of num_rows from sz_tables

Sizing Space Needed (megabytes)	Same as sz_indexes, if entry in sz_indexes exists	The calculated space, given the sizing blocks for index and available space per block

The output of the ixrowsz.sql script should look a lot like ***Table 5***. This shows current and future size estimates on the same line (an incredible concept) for planning DB growth.

TABLE 5.

Owner	**Index Name**	**Catalog Index Entry Size**	**Catalog Blocks for Index**	**Catalog Number of Rows**	**Catalog Space Needed (Meg)**	**Sizing Index Entry Size**	**Sizing Blocks for Index**	**Sizing Number of Rows**	**Sizing Space Needed (Meg)**
<OWNER>	<INDEX_01>	19	11649	4220403	91	19	17473	6330605	137
<OWNER>	<INDEX_02>	37	10529	1958850	82	37	15787	2937150	123
<OWNER>	<INDEX_03>	28	197	48442	2	28	283	69483	2
<OWNER>	<INDEX_04>	19	11649	4220403	91	29	26670	6330605	208
<OWNER>	<INDEX_05>	37	10529	1958850	82	66	28161	2937150	220
<OWNER>	<INDEX_06>	28	197	48442	2	48	485	69483	4

These scripts only cover the basics but it should get you started automating your database sizing. You may want to capture growth info in another set of tables. Over several weeks you can determine all your needs for tablespace sizes and hard drives. Los Alamos currently runs a deficiency based system where we can find from 1 to 30,000 deficiencies in a building. Many of these will never be fixed or even looked at again so storage requirements for 2500 buildings and several thousand structures over 48 square miles can take a large set of tables. Without proper sizing a database can crash like a big dog (or a small Stinson). If you are even remotely interested in this subject the ***References*** are about a million times more useful than I could ever be.

REFERENCES:

- Space Estimations for Schema Objects, Appendix A,
- Oracle Server Administrator's Guide Release 8.0; Oracle Corporation, 1997.
- Millsap, Cary V.--Oracle7 Server Space Management, Revision 1.4b, OraPub (10/31/95)
- Shallahamer, Craig A.--Avoiding a Database Reorganization, OraPub (11/2/94) V2.2
- Koopmann, James F.--Cookbook for Sizing Objects, Oracle Magazine, Vol. XII, Number 3.

NOTES FROM THE CHAIRMAN

Bob Von Eschen, Pantex, Network Chairman

I assume everyone got home from the Network Meeting held at Lawrence Livermore National Laboratory, without any incidents. Comments from the meeting ranged from “Best Yet!” to “They seem to get better and more information each time!” My thanks goes out to PAUL REYNOLDS and the many persons who supported him in facilitating the meeting and lodging.

No final instruction has been received down the “Chain of Command”, on the requirements, reporting medium, and proceed authority for the “Deferred Maintenance Cost Analysis”. Ken Baker should have some information of the required effort very soon, and some reaction comments from the DOE Managers Meeting held in Chicago on April 23-24.

Set aside the week in October 19-23, 1998 to meet at the next LCAM/CAS Network Meeting to be held in Las Vegas. Mitzi Stone has volunteered to ramrod the meeting with assistance from Dick Schlueter, Mike Horn, and one of our previous sponsors Bruce Charlton. Mitzi also has tentatively setup a tour of the "Yucca Mountain Project". Spread the word and be sure to invite your DOE Area and Operations Office personnel.

Keep in touch by participating in the monthly Network conference call, held the second Wednesday of each month at 11:00 Eastern Time.

Promote the CAS program at your every opportunity.

CAIS HOTLINE

Charlie Lu, DynCorp

CAIS Hotline support now has a new FTP server to serve the CAIS community's needs.

The FTP address is 146.138.254.157

User name: cais
Password: cais

The FTP site has a new directory structure:
Download – Please download all your files here and in your particular site subdirectory.
Upgrade – Find the latest Site-CAIS upgrade software here.
Full – Find Full Site-CAIS installations here.

Beta – All beta site participants can find beta software here.

Public – Generic area for transferring and exchanging files (bug fixes, patches, sql scripts, exchange of ideas).

Please feel free to give us input on how to make the FTP site work better for you.

If you have nay problems using the FTP site please feel free to call Kevin Kiah (301) 903-0923 or Charlie Lu (301) 903-0923.

JUST A FEW FINAL COMMENTS

Terry Christie, ORNL, Network Secretary

I would like to say that the last Network meeting held at LLNL was great. Thanks goes out to Paul Reynolds, Bill Denton, Nelda Fondse and all others who were involved in the planning of this very successful meeting. The folks in Las Vegas are currently preparing things for our next meeting which will be held October19-22, 1998. I hope everyone will try to make every effort to attend this meeting because we will be discussing the very important issue of reports. Charlie and Kevin are looking for input from the Network for what route we wish to take for good reporting capability from the CAIS data. I will be letting everyone know further details of the meeting, as they become available.

The article on the LYNX system, which is in this newsletter, came from a brochure that was passed out during a recent demonstration of this system. It is a very impressive system and one, which our site is considering purchasing. We are working on getting a demonstration of the LYNX system at our next meeting.

The next LCAM/CAS conference call will be held Wednesday, May 13, 1998 at 11:00 EST. The phone number for this call will be 202-287-1053. Please try to make time to be on line for this call. If anyone has anything they wish to discuss please let myself or Bob know and we will make sure it is put on the agenda.

The next newsletter will be in August.

Have a great summer.