IIUG Software Repository

ASK Database Management

IIUG Software Repository

There are lots of tools available in the IIUG Software Repository in areas of:

- 4GL Code Libraries
- ESQL/C Libraries
- Data export and import
- Bladelets
- Administration and Monitoring
- Developers' Tools
- General Data Management
- Miscellaneous

IIUG Software Repository

4GL Libraries

- 4gl_lib - Ivaylo Todorov’s library of functions for:
  - String manipulation functions (substrings, find & replace, etc.)
  - Tokenizing
  - SQL Building & enhanced CONSTRUCT
  - Data change logging
  - Schema decoding
  - Dialog boxes, etc.
- 4gltools_ak - My library of functions for:
  - Dialog and prompt boxes, both simplified and complex
  - File IO functions like the “C” printf() functions
  - Printer management
  - Transaction control functions
- Getopt – Jonathan Leffler’s implementation of the “C” getopt() function for 4GL

IIUG Software Repository

ESQL/C Libraries

- Datelib_ak – My library of:
  - C functions to convert between DATE and DATETIME types and several different UNIX Date and Time formats
  - C Structures mapping DATE and DATETIME

IIUG Software Repository

Data export and import

- Myexport – My replacement for dbexport & dbimport with additional features:
  - Does not require a lock on the database
  - Can be run on a secondary without stopping replication
  - Can export/import faster using the HPLoader or External Tables
  - Parallel unload/reload to process multiple tables in parallel
  - Process tables in dependency order
  - Compress unload files and import from compressed files
  - Duplicate data distributions or run dostats during import
  - Remap dbspaces from the source to target
- Ifx_xferdb - Santosh Sajip's scripts to copy a database directly to another database or server using external tables and pipes.
- Myonpload – Ravi Krishna's simplified command line interface to the HPLoader. Easier to use than onpload.
- ul.ec - A binary file data unload & load utility. File format is compatible with the external table "INFORMIX" format and portable between processor architectures.

IIUG Software Repository

Bladelets

- Period –
  - Data types:
    - Period – DATE delimited periods (ie start and end)
    - Dt_period – DATETIME delimited periods
    - EPOCH (earliest) and FOREVER constants
  - Support functions for processing time periods including functions for:
    - Period comparison (equality, less than, less than or equal, etc.)
    - Conversion from string ranges, date ranges, datetime ranges
    - Convert period to interval
    - Period overlap
    - Intersection of two periods
    - Union of two periods

- Random_udr – Random number generation functions including:
  - Rand() & SeededRand()
  - Binomial() & SeededBinomial()
  - Normal() & SeededNormal()

IIUG Software Repository

Administration and Monitoring

- Ratios.shr_ak – An SQL function, ratios(), and shell script, newratios, to calculate and report on some basic server metrics I've developed over the years.
- Utils2_ak – Utilities to manage the server from this package include:
  - dbping.ec - Tests connections and reports connection time as well as the actual host and servername connected as well as which alias was used for the connection.
  - dbsavail.ec - Summary dbspace report. Reports used and free space in pages, KB, and/or percent. Optional chunk detail. Several sort options.
  - listdb7.ec - Displays details about your databases and tables and indexes.
  - mydbdiff - Script to compare two schema files or a schema file to a live schema.
  - myschema - My dbschema replacement utility. Does everything that dbschema does except display data distributions (dbschema -hd) and much more.
  - printfreeB.ec - Print out a table detail report similar to oncheck -pT without locks
  - dostats_ng.ec - The original and still the best way to automate your update statistics runs.
  - drive_dostats - Divide and conquer. Runs multiple copies of dostats updating subsets of your tables. If you have the cores to spare, get the job done faster.

IIUG Software Repository

Developers’ Tools

- Utils2_ak – Developer aids:
  - sqlstruct.ec - Generate data structures from SQL statements including input and output structures for SELECT, INSERT, DELETE, and UPDATE statements. Options to generate C, C++, ESQL/C, x4GL, or SQL DDL.
  - dbstruct.ec - Generate data structures from your database for C, C++, ESQL/C, Structured FORTRAN, x4GL, or SQL DDL.

IIUG Software Repository

General Data Management

- Utils2_ak – Data management tools:
  - dbcopy.ec - Copy data from table to table directly across databases, servers, instances, versions, even between databases at different logging levels without worrying about long transaction rollbacks.
  - dbdelete.ec - Delete large amounts of data from a table FAST without risking long transaction rollbacks.
  - dbmove.ec - Another data copy utility using the methods that dbdelete uses. Can copy some data that dbcopy cannot.

- Sqlcmd – Jonathan Leffler's SQL query utility. Lots of features not in dbaccess:
  - Server mode & client app – stays in memory making scripts faster
  - Different output formats: unload, csv, fixed field, XML
  - Benchmark mode – times queries
  - Unload mode & command line utility
  - Load mode & command line utility
  - SQL command history and rerun
  - Sqlupload utility included (insert or update conditionally)

IIUG Software Repository

Miscellaneous

- utils4_ak – A set of AWK scripts that post-process dbschema and myschema output to produce various SQL and shell scripts to accomplish various DBA tasks. These are mostly presented as examples of how-to-do-it
- Ar2 – A portable version of the UNIX ar utility. Different UNIX environments have different formats for the ar archive file making the file non-portable. (AIX, older HPUX versions, and SystemV/Linux/HPUX each are different). This supports all those and its own portable format that can be extracted on any platform. Used to extract myschema source on AIX. (Does not handle binary ar libraries.)
- Utils2_ak
  dbscript utility. Generates an SQL statement or shell command line for the tables specified by options on the command line.
  - Ex:

    ```
    $ dbscript -d art -t ‘f*’ -c 'myschema -d art -t %s'
    myschema -d art -t foo
    myschema -d art -t fred
    myschema -d art -t fragtest
    myschema -d art -t fragtest2
    myschema -d art -t fred_row_col
    myschema -d art -t for_remote
    ```

```
$ dbscript -d art -t 'f*' -c 'select count(*) from systables t, ssycolumns c where t.tabid = c.tabid and tabname = "%s";'
select count(*) from systables t, ssycolumns c where t.tabid = c.tabid and tabname = "foo";
select count(*) from systables t, ssycolumns c where t.tabid = c.tabid and tabname = "fred";
select count(*) from systables t, ssycolumns c where t.tabid = c.tabid and tabname = "fragtest";
select count(*) from systables t, ssycolumns c where t.tabid = c.tabid and tabname = "fragtest2";
select count(*) from systables t, ssycolumns c where t.tabid = c.tabid and tabname = "fred_row_col";
select count(*) from systables t, ssycolumns c where t.tabid = c.tabid and tabname = "for_remote";
```

Generate a C/C++ Structure:

```
$ dbstruct -d art -t systables

typedef struct SYSTABLES_S {
        char tabname[129];
        char owner[32];
        int partnum;
        int tabid;  /* SERIAL */
        short rowsize;
        short ncols;
        short nindexes;
        double nrows;
        time_t created;  /* DATE */
        int version;
        char tabtype[1];
        char locklevel[1];
        double npused;
        int fextsize;
        int nextsize;
        short flags;
        char site[129];
        char dbname[129];
        int type_xid;
        int am_id;
        int pagesize;
        dtime_t ustlowts;  /* DATETIME */
        int secpolicyid;
        char protgranularity[1];
        short statchange;
        char statlevel[1];
} systables_t;
```

Generate an ESQL/C structure:

```
$ dbstruct -d art -t systables -e
EXEC SQL BEGIN DECLARE SECTION;

typedef struct SYSTABLES_S {
        char tabname[129];
        char owner[32];
        int partnum;
        int tabid;  /* SERIAL */
        short rowsize;
        short ncols;
        short nindexes;
        double nrows;
        time_t created;  /* DATE */
        int version;
        char tabtype[1];
        char locklevel[1];
        double npused;
        int fextsize;
        int nextsize;
        short flags;
        char site[129];
        char dbname[129];
        int type_xid;
        int am_id;
        int pagesize;
        dtime_t ustlowts;  /* DATETIME */
        int secpolicyid;
        char protgranularity[1];
        short statchange;
        char statlevel[1];
} systables_t;
EXEC SQL END DECLARE SECTION;
```

Generate SQL DDL:

```
$ dbstruct -d art -t systables -D

CREATE TABLE systables (
        tabname VARCHAR(128,0),
        owner CHAR(32),
        partnum INT,
        tabid SERIAL,
        rowsize SMALLINT,
        ncols SMALLINT,
        nindexes SMALLINT,
        nrows FLOAT,
        created DATE,
        version INT,
        tabtype CHAR(1),
        locklevel CHAR(1),
        npused FLOAT,
        fextsize INT,
        nextsize INT,
        flags SMALLINT,
        site VARCHAR(128,0),
        dbname VARCHAR(128,0),
        type_xid INT,
        am_id INT,
        pagesize INT,
        ustlowts DATETIME YEAR TO FRACTION(5),
        secpolicyid INT,
        protgranularity CHAR(1),
        statchange SMALLINT,
        statlevel CHAR(1)
);
```

Generate a 4GL record:

```
$ dbstruct -d art -t systables -G

DEFINE systables_rec RECORD
    tabname VARCHAR(128),
    owner CHAR(32),
    partnum INT,
    tabid INT,
    rowsize SMALLINT,
    ncols SMALLINT,
    nindexes SMALLINT,
    nrows FLOAT,
    created DATE,
    version INT,
    tabtype CHAR(1),
    locklevel CHAR(1),
    npused FLOAT,
    fextsize INT,
    nextsize INT,
    flags SMALLINT,
    site VARCHAR(128),
    dbname VARCHAR(128),
    type_xid INT,
    am_id INT,
    pagesize INT,
    ustlowts DATETIME YEAR TO FRACTION(5),
    secpolicyid INT,
    protgranularity CHAR(1),
    statchange SMALLINT,
    statlevel CHAR(1)
END RECORD
```

Generate a structured FORTRAN (FTN99) structure:

```
$ dbstruct -d art -t systables -F

    structure/SYSTABLES_t/
      character*128 tabname
      character*32 owner
      integer*4 partnum
      integer*4 tabid;  ! SERIAL
      integer*2 rowsize
      integer*2 ncols
      integer*2 nindexes
      real*8 nrows
      integer*4 created  ! INFORMIX DATE
      integer*4 version
      character*1 tabtype
      character*1 locklevel
      real*8 npused
      integer*4 fextsize
      integer*4 nextsize
      integer*2 flags
      character*128 site
      character*128 dbname
      integer*4 type_xid
      integer*4 am_id
      integer*4 pagesize
C       record/dtime_t/ ustlowts  ! DATETIME - NOT YET SUPPORTED
      integer*4 secpolicyid
      character*1 protgranularity
      integer*2 statchange
      character*1 statlevel
    end structure
    record/systables_t/ systables
    common/systables_c/ systables
```

Interesting Options: dostats_ng

Dostats -i – Supply table names more complex than -t <wildcard> can handle:

-i ! - introduces a WHERE clause to filter table names to include
-i ! SELECT tabname … - a full query returning a list of table names to include
-i @filename - introduces a filename containing a list of tables to include, one per line
-i @ - reads the table name list from stdin

Dostats -x:

-x ! - introduces a WHERE clause to supply table names that should be excluded
-x @filename – introduces a filename containing a list of tables to exclude, one per line
-x @ - reads the table name list from stdin
-x : - introduces a database name that should not be processed when -d includes a wildcard (ex: dostats -d '*' -x sysmaster)

--small-tables-high – Process small tables with a simple HIGH on all columns
--small-tables-threshold – Sets the # of rows that define a small table
--distributions-high=filename – Specify a file containing a list of columns for which to produce HIGH distributions rather than follow the usual rules.

Interesting Options: myschema

-K – Use long names when creating unnamed constraints and unnamed constraint indexes: <table>_<constr_type> mytable_pk, mytable_fk1, etc. (Default: use short names: <constr letter>_<tabid>_<constrid>: P105_34, R105_33, U105_35

-k – Do or don't create unnamed constraints and unnamed constraint indexes explicitly <toggle> (Default: Do explicit constraints)

-F – When run for specific table(s), include foreign keys that reference the reported table.

--infrastructure=cmd – like dbschema -c -ns
--infrastructure=sql – like dbschema -c

--simple-fragments – Don't name unnamed fragments

Specify these two together to get a simpler schema like dbschema without -ss supplies:
--no-extent-clause – Do not generate EXTENT SIZE and NEXT SIZE clauses
--no-storage-options – Do not generate IN <dbspace> or FRAGMENT BY clauses

Interesting Options: myschema

--dependency-order – Create parent tables before child tables.
-o – Create objects in alphabetical order

--reorg-api – generate “defragment” SQL API commands for selected tables

--primary-filename=tbl-file – Tables & objects needed to define them to tbl-file
--secondary=idx-file – Indexes, constraints, privileges, etc to idx-file
--distribution-file=dist-file – UPDATE STATISTICS commands to dist-file

Extent management options:
-a – Use actual page counts to calculate extent sizes
-m – Use actual pages for initial extent even if that’s less than current allocation
-M [min|max|avg] – For fragmented tables, use min, max, or avg fragment to calculate extent sizing
-e eadj – Adjust the calculated initial extent by eadj percent (0-10000%)
-n nadj – Adjust the calculated next extent by nadj percent (0-10000%)

Owner management options:
-A – Suppress “AS OWNER” clauses in GRANT statements (toggle)
-O – Suppress all owners of objects (sets -A)
--set-owner newowner – Change the owner everywhere to newowner

EXAMPLES
Let's see these in action!

More than one way to skin a cat:

```
$ myschema -d art -t 'ts*' --reorg-api -q
execute function task( 'defragment', 'art:"informix".tsinstancetable' );
execute function task( 'defragment', 'art:"informix".tscontainertable' );
execute function task( 'defragment', 'art:"informix".tscontainerusageactivewindowvti' );
execute function task( 'defragment', 'art:"informix".tscontainerusagedormantwindowvti' );
execute function task( 'defragment', 'art:"informix".tscontainerwindowtable' );
execute function task( 'defragment', 'art:"informix".ts_dumb' );
execute function task( 'defragment', 'art:"art".tst' );
```

More than one way to skin a cat:

```
$ dbscript -d art -t 'ts*' -c "execute function task( 'defragment', 'art:%s' );"
execute function task( 'defragment', 'art:tsinstancetable' );
execute function task( 'defragment', 'art:tscontainertable' );
execute function task( 'defragment', 'art:tscontainerusageactivewindowvti' );
execute function task( 'defragment', 'art:tscontainerusagedormantwindowvti' );
execute function task( 'defragment', 'art:tscontainerwindowtable' );
execute function task( 'defragment', 'art:ts_dumb' );
execute function task( 'defragment', 'art:tst' );
```

More than one way to skin a cat:

```
$ myschema -d art -t 'ts*' | awk -v database=art -f mkdefragment.awk
execute function task( 'defragment', 'art:"informix".tsinstancetable' );
execute function task( 'defragment', 'art:"informix".tscontainertable' );
execute function task( 'defragment', 'art:"informix".tscontainerusageactivewindowvti' );
execute function task( 'defragment', 'art:"informix".tscontainerusagedormantwindowvti' );
execute function task( 'defragment', 'art:"informix".tscontainerwindowtable' );
execute function task( 'defragment', 'art:"informix".ts_dumb' );
execute function task( 'defragment', 'art:"art".tst' );
execute function task( 'defragment', 'art:"art".tst_unload' );

$ cat mkdefragment.awk
/CREATE TABLE/{
        printf "execute function task( 'defragment', '%s:%s' );\n", database, $3;
}
/create table/{
        split($3, a, ".");
        printf "execute function task( 'defragment', '%s:%s' );\n", database, a[2];
}
$
```

With a nod to Lester:

More than one way to skin a cat:

```
$ dbaccess art -

Database selected.

> unload to defrag.sql delimiter '      '
> select "execute function task( 'defragment', 'art:" || trim(owner) ||"."|| trim(tabname)||"' );"
> from systables
> where tabname matches 'ts*';

10 row(s) unloaded.

>

Database closed.

$ cat defrag.sql
execute function task( 'defragment', 'art:informix.ts_dumb' );
execute function task( 'defragment', 'art:informix.tscontainertable' );
execute function task( 'defragment', 'art:informix.tscontainerusageactivewindowvti' );
execute function task( 'defragment', 'art:informix.tscontainerusagedormantwindowvti' );
execute function task( 'defragment', 'art:informix.tscontainerwindowtable' );
execute function task( 'defragment', 'art:informix.tsinstancetable' );
execute function task( 'defragment', 'art:art.tst' );
execute function task( 'defragment', 'art:art.tst_privs' );
execute function task( 'defragment', 'art:art.tst_privs2' );
execute function task( 'defragment', 'art:art.tst_unload' );
$
```

With a nod to Lester:

More than one way to skin a cat:

```
$ $ cat mkdefragment.awk
/CREATE TABLE/{
        printf "execute function task( 'defragment', '%s:%s' );\n", database, $3;
}
/create table/{
        split($3, a, ".");
        printf "execute function task( 'defragment', '%s:%s' );\n", database, a[2];
}

myschema -d art | awk -f mkdefragment.awk
execute function task( 'defragment', 'art:informix.ts_dumb' );
execute function task( 'defragment', 'art:informix.tscontainertable' );
execute function task( 'defragment', 'art:informix.tscontainerusageactivewindowvti' );
execute function task( 'defragment', 'art:informix.tscontainerusagedormantwindowvti' );
execute function task( 'defragment', 'art:informix.tscontainerwindowtable' );
execute function task( 'defragment', 'art:informix.tsinstancetable' );
execute function task( 'defragment', 'art:art.tst' );
execute function task( 'defragment', 'art:art.tst_privs' );
execute function task( 'defragment', 'art:art.tst_privs2' );
execute function task( 'defragment', 'art:art.tst_unload' );
...
```

With a nod to Lester:

More than one way to skin a cat:

```
$ myschema -d mydatabase -g authfile.sql /dev/null
$ egrep ‘SELECT|UPDATE|INSERT|DELETE|INDEX’ authfile|fgrep public
GRANT SELECT ON ph_bg_jobs_seq TO "public" AS "informix";
GRANT SELECT, UPDATE, INSERT, DELETE, INDEX ON command_history TO "public" AS "informix";
GRANT SELECT, UPDATE, INSERT, DELETE, INDEX ON ph_group TO "public" AS "informix";
GRANT SELECT ON ph_alert TO "public" AS "informix";
GRANT SELECT ON ph_bg_jobs TO "public" AS "informix";
GRANT SELECT ON ph_bg_jobs_results TO "public" AS "informix";
GRANT SELECT ON ph_allow TO "public" AS "informix";
GRANT SELECT, UPDATE, INSERT, DELETE, INDEX ON ph_version TO "public" AS "informix";
GRANT SELECT, UPDATE, INSERT, DELETE, INDEX ON storagepool TO "public" AS "informix";
GRANT SELECT, UPDATE, INSERT, DELETE, INDEX ON mon_syssqltrace_info TO "public" AS "informix";
GRANT SELECT, UPDATE, INSERT, DELETE, INDEX ON mon_syssqltrace_hvar TO "public" AS "informix";
GRANT SELECT, UPDATE, INSERT, DELETE, INDEX ON mon_syssqltrace_iter TO "public" AS "informix";
GRANT SELECT, UPDATE, INSERT, DELETE, INDEX ON mon_syssqltrace TO "public" AS "informix";
...
```

Dbsavail:

```
$ dbsavail -f|head -20
Sort by: Free KB.

Dbspace               Number 2K Pages   2K Pages Free    Total KB        Free KB
------------------    ---------------   -------------    ------------    ------------
edi_data                      1048576              64         2097152             128      (PgSz: 2K)
filestore_data                 524288           15247         1048576           30494      (PgSz: 2K)
otto_data                      524288           15796         1048576           31592      (PgSz: 2K)
persistence_data               524288           24559         1048576           49118      (PgSz: 2K)
smartblob_sbspace_1            524288           26274         1048576           52548 SBsp (PgSz: 2K)
batdbs                          50000           48684          100000           97368      (PgSz: 2K)
web_blob                      2096103           55617         4192206          111234 Blob (BlbPg: 2K)
moog_data                      524288           85591         1048576          171182      (PgSz: 2K)
sas_data                      1048576          118050         2097152          236100      (PgSz: 2K)
sas_blob                       524288          153010         1048576          306020 Blob (BlbPg: 2K)
usermgr_index                 1048576          162662         2097152          325324      (PgSz: 2K)
```

Dbsavail:

```
$ dbsavail -f -p|head -20
Sort by: Free KB.

Dbspace               Number 2K Pages   2K Pages Free    Total KB        Free KB
------------------    ---------------   -------------    ------------    ------------
edi_data                      1048576              64         2097152            0.01     (PgSz: 2K)
filestore_data                 524288           15247         1048576            2.91     (PgSz: 2K)
otto_data                      524288           15796         1048576            3.01     (PgSz: 2K)
persistence_data               524288           24559         1048576            4.68     (PgSz: 2K)
smartblob_sbspace_1            524288           26274         1048576            5.01 SBsp(PgSz: 2K)
batdbs                          50000           48684          100000           97.37     (PgSz: 2K)
web_blob                      2096103           55617         4192206            2.65 Blob(BlbPg: 2K)
moog_data                      524288           85591         1048576           16.33     (PgSz: 2K)
sas_data                      1048576          118050         2097152           11.26     (PgSz: 2K)
sas_blob                       524288          153010         1048576           29.18 Blob(BlbPg: 2K)
usermgr_index                 1048576          162662         2097152           15.51     (PgSz: 2K)
```

Ratios:

$ ratios

```
Metric Ratio Report For 2K Cache

        Bufwaits Ratio:             0.000000%
        Buffer Turnover Rate:         0.81/hr
        Used Buffer Turnover Rate:    0.00/hr
        Experimental BTR #2:          0.00/hr
        Experimental BTR #3:          0.00/hr


Metric Ratio Report For 12K Cache

        Bufwaits Ratio:             0.000000%
        Buffer Turnover Rate:         0.00/hr
        Used Buffer Turnover Rate:    0.00/hr
        Experimental BTR #2:          0.00/hr
        Experimental BTR #3:          0.00/hr


Metric Ratio Report Summary For All Caches

        ReadAhead Utilization:      3.060000%
        Bufwaits Ratio:             0.000000%
        Buffer Turnover Rate:         0.80/hr
        Used Buffer Turnover Rate:    0.00/hr
        Experimental BTR #2:          0.00/hr
        Experimental BTR #3:          0.00/hr
        Lock Wait Ratio:            0.00000%
        Sequential Scan Ratio:     18.67000%

Statistics reset at: 2016-11-11 10:26:28
Elapsed time since reset:      477:57:21
```

utils4_ak

```
$ awk -f mkcnt.awk ../myexport/art.sql     |head -20
select ""root".foo", count(*) from "root".foo;
select ""root".regresstab", count(*) from "root".regresstab;
select ""root".bitarraytab", count(*) from "root".bitarraytab;
select ""art".tst", count(*) from "art".tst;
select ""informix".table1", count(*) from "informix".table1;
select ""informix".pagecounts", count(*) from "informix".pagecounts;
select ""informix".tab1", count(*) from "informix".tab1;
select ""informix".tab2", count(*) from "informix".tab2;
select ""art".loadlvtest", count(*) from "art".loadlvtest;
select ""art".loadbigsertest", count(*) from "art".loadbigsertest;
select ""informix".decimaltest", count(*) from "informix".decimaltest;
select ""art".binarytestload", count(*) from "art".binarytestload;
select ""art".binarytestv", count(*) from "art".binarytestv;
select ""art".binarytestlvload", count(*) from "art".binarytestlvload;
…
```

printfreeB

```
$ : printfreeB sysadmin ph_task
Looking at DB: sysadmin, Table: ph_task.
Report for table: sysadmin:ph_task in dbspace #1: root_dbspace.

    Table partition header reports that table has:
        8709% free
        32 pages allocated in 2048 extents
        30 pages used
        81 rows of data in 26 data pages
    Sysptnext reports:   32 pages in 3 extents.
    Bitmap scan reports:
        Unused pages:                     3.
        Bitmap pages:                     1.
        Unused blob pages:                0.
        Partial data pages:               2.
        Partial blob pages:               0.
        Small data pages:                 0.
        Half full blob pages:             0.
        Full data pages:                 25.
        Full index pages:                 1.
        Full blob pages:                  0.
                                -----------
        Total pages reported:            34.
```

listdb7

```
$ listdb7
…
 71 sysadmin
 72 sysmaster
 73 sysuser
 74 sysutils
 75 time
 76 user_db
 77 usermgr
```

listdb7

```
$ listdb7 -d sysadmin -t
There is currently 1 matching database:

  # Database/Table Name
=== ======================
  1 sysadmin
        aus_cmd_info
        aus_command
        aus_work_coldist
        aus_work_dist
        aus_work_icols
        aus_work_info
        aus_work_lock
        command_history
        hadv_emails
        hadv_exception_prof
        hadv_gen_prof
        hadv_profiles
        hadv_run
        hadv_sched_prof
        iwa_datamarts
        iwa_martcolumns
        iwa_martpartitions
        iwa_marttables
…
```

drive_dostats

```
$  drive_dostats

Usage:
  drive_dostats nprocs dbs [tablespec] [-x@excl] [-xexctbl] [dostats options]
                                 [-a] [-i@incl] [-iinctbl]

  Driver script to run 'nprocs' copies of dostats each working on a subset of
  the requested tables.

  tablespec - a MATCHES style wildcard to select tables to include
  -a - Process smallest tables first.
  -x@excl   - excl is a file containing tablenames to ignore
  -xexctbl  - exctbl is a tablename to ignore
       Multiple -x and -x@ options are accepted and can be mixed
  -i@incl   - incl is a file containing tablenames to process
  -iinctbl  - inctbl is a tablename to process
       Multiple -i and -i@ options are accepted and can be mixed
  dostats options - Most dostats options are passed on and are valid.
            Some have no meaning and will fail.
  Mixing -x/-x@ and -i/-i@ options only tables which appear in the include
  list but do not appear in the exclude list will be processed.  Mixing
  these options should be carefully considered and planned to avoid
  unexpected results.
```

ASK Database Management

Email: art@askdbmgt.com
Email: art.kagel@gmail.com
Phone: 732-993-5367

http://askdbmgt.com

Download my latest utils2_ak package:

http://askdbmgt.com/my-utilities.html