Document Purpose

This document is intended to summarize the suite of Comprehensive Solutions (CCSI) tools by providing a synopsis for each tool, along with real-world examples of the tools usage. These tools have been developed for and used to support large, mission-critical production environments. Their use has helped minimized production outages, and has led to improved performance and stability.

This document is not intended to be a technical specification or an all-inclusive installation and configuration guide for these tools. Many of these tools already possess detailed specifications, test plans, and installation instructions.

Please note that as of March of 2005, all CCSI tools that are written in the Korn Shell are to use the “toolslibrary.ksh” standardization library. This provides a consistent look and feel across all of our tools and provides standard modules to be used for common functions.

INGMON

Purpose

INGMON is the cornerstone of CCSI’s suite of Ingres monitoring and management tools, and is used to provide early warning of possible issues within a database or an Ingres installation. INGMON’s notification framework provides the ability to send alerts to email or pagers through a defined SMTP server, as well as log messages to user defined logfile(s) for auditing purposes.

Description

At a high level, INGMON provides the following features:

- Monitoring of log file utilization.
- Monitoring of number and type of Ingres processes.
- Validation of database connectivity to all databases through all DBMS servers.
- Validation of key Ingres files.
- Review of Ingres log files contents.
- Detection and notification of long running queries.
- Detection and notification of blocking lock conditions.

INGMON is fully configurable by the use of configuration files, and has an integrated plug-in architecture. This allows the extension of Ingmon through the use of programs and/or scripts written by Comprehensive Solutions, the DBA staff, or by other third-parties.

INGMON is currently a series of Korn shell scripts but is undergoing conversion to a compiled binary. It is intended to be run from the system scheduler, with the exact timing intervals determined by Comprehensive Solutions at installation and configuration time. INGMON does not use excessive CPU, memory, or disc in its operations.

INGMON has been deployed and utilized without problems by Comprehensive Solutions at a number of sites in the course of the last 8 years, including one of the world’s largest Ingres sites.

The INGMON process is divided into the following checks and validations:

- INGRES CHECK: Validates the overall health of the Ingres installation, including number of processes, key file status, and critical log messages.
- CONNECTIVITY CHECK: Validates access to all databases within an installation through all defined DBMS servers.
- LONGRUN: Reviews queries currently executing within the DBMS server and sends notification once queries have exceeded a user-defined threshold.
- LOCK CHECK: Reviews lockstat output, checks for blocking locks, and sends notification once blocking locks have exceeded a user-defined threshold.
- LOG TRENDING: Reviews the Ingres errlog.log at user-defined intervals and produces a trend list of all errors seen. Non-critical or “noise”-type errors are skipped by default, however this behaviour can be modified by the user (either to remove or add errors to the ignore list).

Although it can be run without them, INGMON is designed to be run in conjunction with the Comprehensive Solutions INGBEGIN and INGEND utility scripts for startup and shutdown of Ingres.

Rfwd.ksh and Ckpt.ksh

Purpose

In order to better manage backup and recovery in an Ingres environment, Comprehensive Solutions has created two wrapper scripts for the standard ckpdb and rollforwarddb commands. These wrappers handle several aspects of backup and recovery that usually need to be either performed manually by the DBA, or require the creation of scripts or scripted procedures.

Feature Description

Both scripts feature the following:

- Detailed help and usage information.
- Dependency checking – the scripts will validate that all required directories, files, and binaries are available before starting.
- Integration with other Comprehensive Solutions tools.
- Fully configurable for multiple Ingres installations via a configuration file.
- Tested on all Ingres versions from OpenINGRES 1.2 through Ingres 2006.
- Email and pager notification on key events.
- Implementation of parallel checkpoint logic, with runtime configurable parallel streams of operation.
- Implementation of checksum logic, which ensures that the checkpoint file(s) are not corrupt **before** recovery begins.
- Management of journal, dump, and checkpoint files based on a user-defined value for the number of checkpoints to keep on-disc.
- Detailed logging, with a defined log rotation parameter limit disc usage if desired.

These scripts, which have been deployed at several high-volume and high-availability clients with great success, have been involved in several high profile recovery efforts for our clients. Like all of our products, these scripts are used to support our internal Ingres infrastructure.

Simplified Query Analysis Tool

Purpose

The Simplified Query Analysis tool is designed to parse through Ingres source code, review the queries contained within the source, and provide several reports. These reports are designed to help understand the interrelationships that exist in the code – for example, what tables are in use by a particular source module? What non-key columns are used in joins, and as such should have statistics built on them?

Feature Description

The following reports are produced by the SQA Tool.

Report Name	Report Detail
JOINS	Detail on all joins found for the specified source code and the database used.
FILES	All source code files considered as part of the processing.
RA_XREF	Cross reference on relation and attribute along with SQL usage.
R_XREF	Cross reference on relation along with SQL usage.
SUMMARY	Summary of source code processed for the current database.
WHERE	Detail on how attributes are used in where clause, in terms of operation (equal, not equal, etc)
ERRORS	Errors encountered during parsing – these may be possible errors in the source code module or errors in the way SQL statements are accessing data (based on table access).

In addition to the these reports, the Simplified Query Analysis tool will create a series of tables in the database that the applications connect to that can then be used as part of additional user-defined reports or queries. Also, the Simplified Query Analysis tool will generate the appropriate “optimizedb” flags to handle the specific needs of your applications.

The Simplified Query Analysis tool currently can parse the following source file types:

- ESQLC
- ESQLF
- ABF
- RW
- OpenROAD
- SQL

Since the Simplified Query Analysis tool can parse SQL, additional source code – such as .NET, JDBC, etc – can be processed, provided that it is first parsed down to SQL. This can usually be done via a shell script, or by using Ingres’s PRINTTRACE facility to extract SQL. If there is sufficient need by a client, CCSI will consider writing customised parsers for other languages.

Output Examples

CMD_JOINS.SQA

```
FOP_CHECK.SF |     244 | TEMP_TAB6 | testdate | TEMP_TAB1 | testdate
FOP_CHECK.SF |     250 | TEMP_TAB6 | testdate | TEMP_TAB2 | testdate
FOP_CHECK.SF |     256 | TEMP_TAB6 | testdate | TEMP_TAB3 | testdate
FOP_CHECK.SF |     263 | TEMP_TAB6 | testdate | TEMP_TAB4 | testdate
FOP_CHECK.SF |     269 | TEMP_TAB6 | testdate | TEMP_TAB5 | testdate
```

CONTROL.SQA

```
-db borges::ulol
FOP_CHECK.SF
WELLTEST_GRAPH.SF
WTI_LOCATION.SF
```

ERRORS.SQA

```
FILE:   ERRORS.SQA              Tue Mar 28 16:08:28 2006
PROGRAM "PARSE.C" ERROR OUTPUT:

incident #: 1  error #: 30  line: 237  col: 34  file: FOP_CHECK.SF
error: Illegal oper in where clause:unparsable
     1                  FROM   foptemp1 f

     1        ON COMMIT PRESERVE ROWS WITH NORECOVERY

                                 ^
incident #: 2  error #: 18  line: 1447  col: 1  file: FOP_CHECK.SF
error: Syntax error, no end of string exists
        end

        end

^
incident #: 3  error #: 18  line: 742  col: 1  file: WTI_LOCATION.SF
error: Syntax error, no end of string exists
        end

        end

^
```

JOINS.SQA

```
                                                 Simplified Query Analysis  (tm)           Page:
1
        Copyright 2006, Comprehensive Solutions (http:/www.comp-soln.com)             Run Time:
Tue Mar 28 16:08:28 2006
                                                    RELATION JOIN INFORMATION

Relation Name              #Join Atts  Attribute List
-------------------------  ----------  ------------------------------------
The number of joins found was 6.

TEMP_TAB1                 #ATTS:   1    testdate ==
                    Relations #:        TEMP_TAB6               1
                        Files #:        FOP_CHECK.SF            1

TEMP_TAB2                 #ATTS:   1    testdate ==
                    Relations #:        TEMP_TAB6               1
                        Files #:        FOP_CHECK.SF            1

TEMP_TAB3                 #ATTS:   1    testdate ==
                    Relations #:        TEMP_TAB6               1
                        Files #:        FOP_CHECK.SF            1

TEMP_TAB4                 #ATTS:   1    testdate ==
                    Relations #:        TEMP_TAB6               1
                        Files #:        FOP_CHECK.SF            1

TEMP_TAB5                 #ATTS:   1    testdate ==
                    Relations #:        TEMP_TAB6               1
                        Files #:        FOP_CHECK.SF            1

TEMP_TAB6                 #ATTS:   1    testdate ==
                    Relations #:        TEMP_TAB1               1    TEMP_TAB2               1
TEMP_TAB3               1
                                        TEMP_TAB4               1    TEMP_TAB5               1
                        Files #:        FOP_CHECK.SF            5
```

RA_XREF.SQA

```
                                                 Simplified Query Analysis  (tm)           Page:
1
        Copyright 2006, Comprehensive Solutions (http:/www.comp-soln.com)             Run Time:
Tue Mar 28 16:08:28 2006
                                                    RELATION ATTRIBUTE CROSS-REFERENCE USAGE

source file name           table name               attribute name                delet  insrt
selct  updat  where   join
-------------------------  ------------------------ ------------------------     -----  -----  --
---  -----  -----  -----
WTI_LOCATION.SF            TEMP_DSS_CELLAR          *                               .      .
.      .      .      .
FOP_CHECK.SF               TEMP_TAB1                *                               .      .
.      .      .      .
FOP_CHECK.SF               TEMP_TAB1                atests                          .      .
1      .      .      .
FOP_CHECK.SF               TEMP_TAB1                testdate                        .      .
.      .      1      1
FOP_CHECK.SF               TEMP_TAB2                *                               .      .
.      .      .      .
FOP_CHECK.SF               TEMP_TAB2                btests                          .      .
1      .      .      .
FOP_CHECK.SF               TEMP_TAB2                testdate                        .      .
.      .      1      1
FOP_CHECK.SF               TEMP_TAB3                *                               .      .
.      .      .      .
FOP_CHECK.SF               TEMP_TAB3                ctests                          .      .
1      .      .      .
FOP_CHECK.SF               TEMP_TAB3                testdate                        .      .
.      .      1      1
FOP_CHECK.SF               TEMP_TAB4                *                               .      .
.      .      .      .
FOP_CHECK.SF               TEMP_TAB4                dtests                          .      .
1      .      .      .
FOP_CHECK.SF               TEMP_TAB4                testdate                        .      .
.      .      1      1
FOP_CHECK.SF               TEMP_TAB5                *                               .      .
.      .      .      .
FOP_CHECK.SF               TEMP_TAB5                jtests                          .      .
1      .      .      .
FOP_CHECK.SF               TEMP_TAB5                testdate                        .      .
.      .      1      1
FOP_CHECK.SF               TEMP_TAB6                atests                          .      .
1      1      .      .
FOP_CHECK.SF               TEMP_TAB6                btests                          .      .
1      1      .      .
FOP_CHECK.SF               TEMP_TAB6                ctests                          .      .
1      1      .      .
FOP_CHECK.SF               TEMP_TAB6                dtests                          .      .
1      1      .      .
FOP_CHECK.SF               TEMP_TAB6                jtests                          .      .
1      1      .      .
FOP_CHECK.SF               TEMP_TAB6                testdate                        .      .
1      .      5      5
FOP_CHECK.SF               fop                      csg_psi                         .      .
1      .      .      .
FOP_CHECK.SF               fop                      fop                             .      .
1      .      .      .
FOP_CHECK.SF               fop                      remark                          .      .
1      .      .      .
FOP_CHECK.SF               fop                      test_date                       .      .
1      .      1      .
FOP_CHECK.SF               fop                      wellname                        .      .
1      .      .      .
FOP_CHECK.SF               foptemp1                 *                               .      .
.      .      .      .
FOP_CHECK.SF               foptemp1                 atests                          .      .
1      .      .      .
FOP_CHECK.SF               foptemp1                 btests                          .      .
1      .      .      .
FOP_CHECK.SF               foptemp1                 csg_psi                         .      .
1      .      2      .
FOP_CHECK.SF               foptemp1                 ctests                          .      .
1      .      .      .
```

```
FOP_CHECK.SF               foptemp1                 dtests                          .      .
1      .      .      .
FOP_CHECK.SF               foptemp1                 fop                             .      .
1      .      1      .
FOP_CHECK.SF               foptemp1                 jtests                          .      .
1      .      .      .
FOP_CHECK.SF               foptemp1                 remark                          .      .
1      .      .      .
FOP_CHECK.SF               foptemp1                 test_date                       .      .
8      .      .      .
FOP_CHECK.SF               foptemp1                 wellname                        .      .
7      .      5      .
WTI_LOCATION.SF            wellcmptdate             date_begun                      .      .
1      .      .      .
WTI_LOCATION.SF            wellcmptdate             wellname                        .      .
1      .      .      .
```

R_XREF.SQA

```
                                                 Simplified Query Analysis  (tm)           Page:
1
        Copyright 2006, Comprehensive Solutions (http:/www.comp-soln.com)             Run Time:
Tue Mar 28 16:08:28 2006
                                                  RELATION CROSS-REFERENCE USAGE

source file name           table name                creat  delet  insrt  selct  updat
-------------------------  ------------------------  -----  -----  -----  -----  -----
WTI_LOCATION.SF            TEMP_DSS_CELLAR               1      .      .      .      .
FOP_CHECK.SF               TEMP_TAB1                     1      .      .      1      .
FOP_CHECK.SF               TEMP_TAB2                     1      .      .      1      .
FOP_CHECK.SF               TEMP_TAB3                     1      .      .      1      .
FOP_CHECK.SF               TEMP_TAB4                     1      .      .      1      .
FOP_CHECK.SF               TEMP_TAB5                     1      .      .      1      .
FOP_CHECK.SF               TEMP_TAB6                     .      .      .      1      5
FOP_CHECK.SF               fop                           .      .      .      1      .
FOP_CHECK.SF               foptemp1                      1      .      .      7      .
WTI_LOCATION.SF            wellcmptdate                  .      .      .      1      .
```

SUMMARY.SQA

```
                                                 Simplified Query Analysis  (tm)           Page:
1
        Copyright 2006, Comprehensive Solutions (http:/www.comp-soln.com)             Run Time:
Tue Mar 28 16:08:28 2006
                                                        SUMMARY STATISTICS



In this run there were 3 files processed. The Control file was located in the
directory CONTROL.SQA.

Optimize files were not generated.

The number of errors found was:      3

The number of errors suppressed was: 0

Loaded data into borges::ulol.

Total CREATEs: 8

Command  Tot_Num  #Repeat_pos  #Repeated  %Repeated  #In db_procs  #Dyn_where
No DELETE commands.

No INSERT commands.

SELECT         6            6          0       0.0%             0           0

UPDATE         5            5          0       0.0%             0           0


Within WHERE clauses, Simplified Query Analysis counts the number of times
a function references either an attribute or a value.  This can
produce more ineffecent QEPs.  See User Guide for more information.
```

```
Total WHERE clauses parsed:  13
    Total functions containing attributes:  0
    Total functions containing variables:  0
    Function::WHERE ratio:    0.0%


The following is a list of the number of times #tables were used in a command.
It is not good practice to have > 4 tables referenced in a command.

#TABLES ..in.. #COMMANDS

    1            3
    2            13
    3            0
    4            0
    5            0
    6            0
    7            0
    8            0
    9            0
   10            0
   11            0
   12            0
   13            0
   14            0
   15            0
   16            0
```

WHERE.SQA

```
                                                 Simplified Query Analysis  (tm)           Page:
1
        Copyright 2006, Comprehensive Solutions (http:/www.comp-soln.com)             Run Time:
Tue Mar 28 16:08:28 2006
                                                        WHERE CLAUSE USAGE

Table Name               Attribute Name           equal   neq   rng   set  expr qualf subqy  join
------------------------ ------------------------ -----  ----  ----  ----  ----  ----  ----  ----
TEMP_TAB1                testdate                     1     .     .     .     .     .     .     1
TEMP_TAB2                testdate                     1     .     .     .     .     .     .     1
TEMP_TAB3                testdate                     1     .     .     .     .     .     .     1
TEMP_TAB4                testdate                     1     .     .     .     .     .     .     1
TEMP_TAB5                testdate                     1     .     .     .     .     .     .     1
TEMP_TAB6                testdate                     5     .     .     .     .     .     .     5
fop                      test_date                    .     .     1     .     .     .     .     .
foptemp1                 csg_psi                      1     .     1     .     .     .     .     .
foptemp1                 fop                          .     .     1     .     .     .     .     .
foptemp1                 wellname                     .     .     5     .     .     .     .     .
```

ChecksumDB

Purpose

The ChecksumDB process is an ESQLC binary that is used to validate data – the process operates at the table level and reads each row and column in a table and computes a “checksum” for the table.

Feature Description

This tool is often used by CCSI in migrations – for example, all tables in the source database are processed by ChecksumDB, then all tables in the target database are processed with the results compared. This ensures that the data has been migrated properly. Another use of ChecksumDB is to validate access to a table – since ChecksumDB works on the entire table, it can be used to make sure that each row in a given table is able to be selected. This has been used by CCSI in the past to locate corrupt tables, which then can be addressed.

Finally, ChecksumDB can be used in testing – by specifying a flag on the command line, the program will compute a separate checksum for each column in a table (it also can be restricted to only certain columns). This can be then used to validate changes to specific tables or columns as part of development or testing operations.

DUMP_INFO.ksh

Purpose

The DUMP_INFO process, which is implemented as a series of korn shell scripts, provides a “snapshot” of the current state of an Ingres system, both at the OS level and at the Ingres level. This process – which can be run automatically via a monitoring script when a problem is detected, or manually by the DBA on an Ad Hoc basis – creates a “DUMP_INFO” directory and populates it with key data that can then be uploaded to Ingres technical support, utilized by the DBA staff to perform root cause analysis **after** the system has been brought back into production mode, or archived and used for trend analysis.

Feature Description

In its basic form, DUMP_INFO collects information on the following key areas:

- Operating system – process table, disk space, network configuration, etc.
- General Ingres configuration data
- All current Ingres log files
- Ingres locking system
- Ingres logging system
- Ingres DBMS servers
- Shared memory information
- Other data as required

Recovery Tool

Purpose

The CCSI Recovery Tool was designed to allow recovery of otherwise unrecoverable databases by extracting the database journal data (via auditdb), creating SQL to redo the changes to the database, and applying them. This tool has been used successfully in several high-profile engagements to recover databases that were regarded to be “unrecoverable.”

Feature Description

The recovery tool relies on the data stored in the Ingres journal files, and because of this is only effective in recovering database tables that are journaled. The use of the Ingres “auditdb” utility eliminates the need to read the journal files directly – however, due to differences in the output from auditdb between OpenINGRES 1.2 and Advantage Ingres 2.0 and later, the tool needs to be built specifically for one version of Ingres.

On the surface, this tool provides similar functionality to the Ingres Journal Analyzer product that is distributed with the Ingres Windows distribution. However, the recovery tool is much more robust and has been used in situations where the IJA has failed. Additionally, the recovery tool has the advantage of being cross-platform capable, and runs as a separate process for each table to be recovered (so the failure of one process does not kill the entire recovery effort, as is the case with the IJA).

Wdw.ksh (who did what)

Purpose

This script produces a summary of database operations by user, table and type of operation for the given database and optionally time period. The output can be restricted to a particular set of users by supply a user file

Feature Description

The script has the ability to include/exclude users from the report based on a configuration file. This script is mostly used as a cronjob to provide detail data on a specific time interval (daily, weekly, monthly).

The script has a dependency on Ingres journal files – if it is unable to read the journal files (either due to permissions or that they do not exist) the script will not be able to process.

CheckDep.pl

Purpose

CheckDep.pl is a Perl script operates at the Unix level and is used to validate a list of dependencies (with modifications, this script could be used on other platforms – the biggest holdup is the file specification used in various operating systems. The most common usage is to verify that all components (binaries, directories, configuration files, etc) are present for an Ingres installation.

Feature Description

This Perl script uses a plain text configuration file that contains a number of lines – the lines contain a "Test Name" followed by an object (file/directory/server/etc). The remainder of the line contains data specific to the test to be performed.

Test Operators Understood by CheckDep.pl

Test Name	Description
PRGM	Test to validate that file is executable, verify checksum, size, timestamp.
REDC	Test to validate that file is readable, verify checksum, size, timestamp.
WRTC	Test to validate that file is writable, verify checksum, size, timestamp.
REDN	Test to validate that file is readable.
WRTN	Test to validate that file is writable.
RDIR	Test to validate that directory is readable.
WDIR	Test to validate that directory is writable.
PRTR	Test to validate that printer queue exists (does not work on all platforms).
DUPL	List duplicates (same name).
PING	Perform ping test on host (deprecated due to non-root ICMP issues).
CKMD	Check mode and owner (no checksum). Normally used on directories.

The script will run in one of two modes:

- In "validate" mode the script will scan the configuration file passed on the command line and perform the tests requested, reporting on the total number of lines tested, the number of lines without errors, the number of lines with errors, and the total number of errors (multiple errors can be generated from each line).
- In "update" mode the script performs the same tasks as described above, with the additional step of creating an updated configuration file (containing the new MD5 sums, owner/size changes, etc) that is written out to /tmp (can be changed in the script). This file can then replace the existing configuration file.

There is an associated shell script – "CheckDep.ksh" – which has been written to automate the usage of this script. Additionally, logic from this script has been incorporated into the standard tools library for usage in other scripts and utilities.

SanityCheck.ksh

Purpose

This script performs three checks on the systems catalogs to ensure consistency between certain key tables/views. The checks are based on simple counts – in the event of an error, email and/or pager alerts are raised. The precise detail of any discrepancies must be determined manually.

The script will raise a single email alert if one of more of the checks fail. It will also return a non-zero result and preserve its log file.

JournalArchive.ksh

Purpose

Journal Archive provides a mechanism for creating near realtime backups of Ingres journal files to both a local disc system and a remote server.

ingstatus.ksh

Purpose

The script ingstatus will return an exit code to indicate if Ingres is in the mode specified by the supplied parameter. A zero exit code indicates that the ingres is in the specified mode.

Feature Description

This script was designed for use with ingbegin.ksh and ingend.ksh to check if ingres is actually running in the required mode. The possible modes are NORMAL plus those defined in the ingbegin.ksh configuration file (see ingbegin.ksh.doc for details of mode definitions).

If DOWN mode is specified in the –s parameter ingstatus.ksh will exit with a return code of 9 if any processes are found.

For all other modes the script checks the process list against what is expected. For NORMAL mode or any modes based on NORMAL it interrogates config.dat to determine which processes should be running along with those defined in the ingbegin.ksh configuration file.

MonDBLocks.ksh

Purpose

This script monitors table locks inside Ingres databases. If locks persists after a specified threshold, alerts are raised.

This script makes use of the output from the “lockstat” command it is dependant on the output format. If the format changes, the script will need to be updated.

When checking for locks, there are 2 types of alarms:

Alarm #1

Alarm #1 is raised when locks are persisting, meaning they were detected during the previous run. When locks are detected, they are saved in a file $LOCK_LIST then compared to the previous list $LOCK_HIST. If the lock is found in $LOCK_HIST, it is persisting and should be cleared. The duration of the lock is the elapse time between each run of this script which is typically 2, 4 or 5 minutes. A report will be printed in the log file.

Alarm #2

Alarm #2 is raised when many locks are detected which typically indicates a problem with the Ingres database installation. The default is set at 20 but can be changed using the optional –L parameter.

The log file will contain the output from ‘lockstat’ and report on the individual locks as detected including details about the locked resource and information on the other resource responsible for the locks.

CompareDB.ksh

Purpose

Compares the tables of two databases at the data level or compares a copydb/unloaddb output with the tables of a database.

Feature Description

This script is designed to compare the contents of two databases utilizing the copydb command. This script is for use in conjunction with the CheckSumDB utility, and provides a second level of validation. This script is also available for standalone usage when the amount of time required by CheckSumDB would be prohibitive.

CompareDB.ksh can handle multi location unloads created by MLUnloadDB.ksh

MLUnloadDB.ksh

Purpose

MLUnloadDB.ksh performs an unloaddb to multiple directories, thus providing a mechanism to make use of multiple file systems. It is used in conjunction with MLTransfer.ksh and MLReloadDB.ksh to build new copies of a database.

Feature Description

The script will copy out each table in turn to the directory with the most available space. If the supplied directories have significantly different amounts of free space then the smaller directories may not get used. The script reports at the end which directories it actually used. It also records this information in the MLU.locs file in each of the directories so that it can be used by MLTransfer.ksh and MLReloadDB.ksh.

The script can optionally compress the data. This is done using a pipe so that no extra disc space is required.

MLReloadDB.ksh

Purpose

MLReloadDB.ksh performs a database reload from multiple directories, thus providing a mechanism to make use of multiple file systems. It is also designed for usage in automated refreshes, and to provide a consistent process for loading/unloading databases.

Feature Description

This script relies on several other scripts, including MLUnloadDB.ksh and MLTransfer.ksh. Additionally, most deployments of this script also use CheckSumDB and CompareDB.ksh to validate the data being copied.

The script will automatically uncompress the data. This is done using a pipe so that no extra disc space is required. It does not matter if there is a mix of compressed and uncompressed files. Note that there is no relationship between the number of directories used in the original unload and the directories on the server where this script is run. They may of course be the same but MLTransfer.ksh deal with the mapping of different numbers of locations.

MLReloadDB.ksh can also be instructed to change the database locations used by tables and indexes.

MLTransfer.ksh

Purpose

MLTransfer.ksh copies a multi-directory database unload created by MLUnloadDB.ksh to a server. The script is designed to run on the target server only so that it can determine the available disc space.

Feature Description

The script can copy the unload data to any number of directories on the target server. The directories do not have to correspond in number or name to those on the source server, though of course they can do so if desired.

The script will transfer each table in turn to the directory with the most available space. If the supplied directories have significantly different amounts of free space then the smaller directories may not get used. The script reports at the end which directories it actually used.