Oracle® CODASYL DBMS

Release Notes

August 2005

Release 7.1.2.1

ORACLE®

Oracle CODASYL DBMS Release Notes, Release 7.1.2.1

Contents

Preface

Purpose of This Manual

The Oracle CODASYL DBMS release 7.1.2.1 release notes summarize new features, corrections to software, restrictions, workarounds, and problems. They also include new features and corrections provided in release 7.1.2. These release notes cover Oracle CODASYL DBMS for OpenVMS Alpha which is referred to by its abbreviated name, Oracle CODASYL DBMS.

Intended Audience

This document is intended for users responsible for:

- System management
- Database administration
- Application programming

Document Structure

This document consists of four chapters:

Chapter 1	Describes installation requirements and location of documents
Chapter 2	Describes corrected software errors
Chapter 3	Describes known problems, restrictions, and workarounds, as well as documentation errors and omissions
Chapter 4	Describes new features and corrected software errors in release 7.1.2.

Conventions

The following conventions are used in this document:

word	A lowercase word in a format example indicates a syntax element that you supply.
[ ]	Brackets enclose optional clauses from which you can choose one or none.
{ }	Braces enclose clauses from which you must choose one alternative.
. . .	A horizontal ellipsis means you can repeat the previous item.

A vertical ellipsis in an example means that information not directly related to the example has been omitted.

.
.
.

1
Installation and Documentation

This chapter contains installation and documention information for Oracle CODASYL DBMS release 7.1.2.1.

1.1 Installation of Oracle CODASYL DBMS Software

Please refer to the *Oracle CODASYL DBMS Installation Guide* for release 7.1 for detailed Oracle CODASYL DBMS installation instructions. Oracle strongly recommends that you read the installation guide before attempting an installation.

To extract either the PostScript (PS) or text (TXT) version of the installation guide from the kit, use one of the following commands:

```
$ BACKUP <device>:DBM07121A071.A/SAVE/SEL=DBM071_INSTALL_GDE.PS
```

or

```
$ BACKUP <device>:DBM07121A071.A/SAVE/SEL=DBM071_INSTALL_GDE.TXT
```

The release 7.1 installation guide is available on MetaLink in Adobe Acrobat PDF format:

```
Top Tech Docs\Database\Rdb\CODASYL DBMS\Documentation\
    Oracle CODASYL DBMS Documentation Index\CODASYL DBMS V7.1 Installation Guide
```

The installation guide is also available on OTN (www.oracle.com/rdb):

```
Documentation\Oracle CODASYL DBMS Documentation\
    CODASYL DBMS V7.1 Installation Guide
```

1.2 Documentation in Adobe Acrobat Format

You can view the documentation in Adobe Acrobat format using the Acrobat Reader, which allows anyone to view, navigate, and print documents in the Adobe Portable Document Format (PDF). For information about obtaining a free copy of Acrobat Reader and for information on supported platforms, see the Adobe Web site at:

```
http://www.adobe.com
```

The Oracle CODASYL DBMS and Hot Standby documentation in Adobe Acrobat format is available on MetaLink:

```
Top Tech Docs\Database\Rdb\CODASYL DBMS\Documentation\
    Oracle CODASYL DBMS Documentation Index\<bookname>
```

The documentation is also available on OTN:

```
Documentation\Oracle CODASYL DBMS Documentation\<bookname>
```

2
Problems Corrected

This chapter describes software errors corrected in Oracle CODASYL DBMS release 7.1.2.1.

2.1 Transaction State Not Set in Root

If the COMMIT TO JOURNAL OPTIMIZATION feature was enabled, the transaction state for database users was not correctly shown by the DBO/DUMP /USERS command. That is, the DBO/DUMP/USERS output may have indicated that a process did not have a transaction active when in fact it did. This problem would occur because the database root file data structures that reflected the transaction state of a user were not always updated when the COMMIT TO JOURNAL OPTIMIZATION feature was enabled.

This problem has been corrected in this release of Oracle CODASYL DBMS. The database root file will now correctly show that a user has a transaction active. Note that when the COMMIT TO JOURNAL OPTIMIZATION feature is enabled, the transaction sequence number (TSN) displayed by the DBO/DUMP/USER command will be zero. The TSN is not maintained in the database root file when this optimization is being used.

2.2 Database Recovery Process May Fail When AIJs Are Full

When AIJ files were full, it was possible for the database recovery process to fail when trying to recover processes with active read-write transactions. The failure occurred when trying to undo the current uncommitted transaction. The DBR log shows:

```
<timestamp> - Starting transaction UNDO for TSN 0:128
<timestamp> - UNDO TSN 0:128 starts at RUJ JFA (2:0)
<timestamp> - Scanning AIJ for optimistic commit
<timestamp> - Starting AIJ scan at 1:513
%DBM-I-BUGCHKDMP, generating bugcheck dump file SYS$SYSROOT:[SYSEXE]DBMDBRBUG.DMP;
```

Note that the AIJ scan started at 1:513 while the AIJ file is only 512 blocks.

```
The exception in the dump is :
***** Exception at 001B3418 : UTIO$READ_BLOCK + 00000208
%DBM-F-FILACCERR, error reading disk file
-SYSTEM-W-ENDOFFILE, end of file
```

This problem has now been corrected.

2.3 Bugchecks in AIJUTL$FREE_DIRTY_ARBS When Journals Full

BUG 4088221

Various processes could fail with bugchecks in AIJUTL$FREE_DIRTY_ARBS if all journals became full, a user process was terminated, a journal was backed up, and further journaling activity in the database occurred.

The bugcheck exception was similar to the following:

```
***** Exception at 011E9E94 : AIJUTL$FREE_DIRTY_ARBS + 000005F4
%COSI-F-BUGCHECK, internal consistency failure
```

This problem can be avoided by ensuring that journals are backed up prior to all journals becoming full.

This problem has now been corrected.

2.4 Excessive CPU Consumed by LRS Process

BUG 4268610

When you enabled the Hot Standby feature the AIJ Log Recovery Server (LRS) process would sometimes consume inordinate amounts of CPU. This would especially be true when the following conditions were met:

- Many short duration transactions were being generated by the application
- Many journal slots were allocated in the database
- The current journal sequence number was relatively large, for example, in the tens of thousands

When the above conditions were met, a problem in the recovery code statistics collection was exposed. The recovery code was attempting to determine how many blocks of journal were spanned in each committed transaction. However, the recovery code did not record the journal starting point for the transaction, thus it would appear that the transaction started in journal sequence number 0. The statistics code would then first attempt to find journal sequence 0. When it did not find sequence 0, it would then try to find sequence 1 so that it could approximate the number of blocks in the transaction. That journal would not be found so it would continue searching for journals until it tried all possible journal sequence numbers up to the current journal. This could consume huge amounts of CPU time.

There is no workaround for this problem.

This problem has been corrected in this release. The recovery code will no longer attempt to gather statistics on the number of journal blocks per transaction because the number is not useful in the context of database recovery.

2.5 DBO /CLOSE /ABORT=DELPRC /WAIT Hang

Bug 4304166

Starting with Oracle CODASYL DBMS release 7.1, it was possible that when you enabled the row cache feature, closing a database with the /ABORT=DELPRC and /WAIT qualifiers could hang. The problem caused the database recovery processes and the record cache server process to become deadlocked. To actually close the database may require manual intervention to explicitly STOP/ID the database recovery processes.

With this combination of command line qualifiers, the DBMS monitor process was incorrectly issuing a $DELPRC command to the record cache server process. The /ABORT=DELPRC and /ABORT=FORCEX qualifiers are intended to apply only to database user processes and not to database servers when the /WAIT qualifier is not specified.

This problem has been corrected. The DBMS monitor process no longer attempts to issue a $DELPRC command to the record cache server process when the /WAIT qualifier is specified.

2.6 Failed User Processess Not Recovered if DBR Startup Fails

Failed user processes would not always be recovered if a database shutdown was forced due to an error that occurred when the database monitor attempted to create a database recovery (DBR) process.

For example, if there were insufficient process slots available on the system to create another process and the database monitor could not create a DBR, then the database would be shut down. The forced shutdown would leave behind user entries in the database for all the processes that were accessing the database from that node. However, the monitor would incorrectly clear the entry in the data structure used to determine if a node recovery was needed. This would prevent the failed processes from being recovered the next time that the database was accessed.

Since the failed processes were not recovered, it is possible that database corruption could be introduced because changes made by the failed processes were not rolled back before other processes accessed the data. It is possible that logical inconsistincies could have been introduced in the database that cannot be detected by the DBO/VERIFY command. If this problem is encountered, it is advised that the database be restored from the last backup and after-image journals be applied to recover the database up to the point of failure. Recovery past the point of failure may be possible, but could re-introduce corruption.

This problem can be avoided by ensuring that there are sufficient system resources available to start DBR processes when needed.

This problem has now been corrected.

2.7 DBO/SHOW AFTER/BACKUP May Stall When AIJs Are Full

Bug 4347617

In previous releases of Oracle CODASYL DBMS, if you used mutiple circular journals which had a state of Full, the DBO/SHOW AFTER/BACKUP command could stall with a "waiting for AIJ journal lock 0 (PW)" error.

This problem has now been corrected.

2.8 Bugcheck at KOD$UNBIND

Bug 4469657

Starting with Oracle CODASYL DBMS V7.1.1.1, if you attempted to UNBIND from an application or DBQ session without first terminating the transaction (COMMIT or ROLLBACK), a bugcheck would be generated and the process would be terminated.

This problem has now been fixed. Now, DBMS will raise a DBM-F-TRAN_IN_ PROG exception, indicating that there is a transaction in progress, and allow the application to deal with the error.

2.9 Journals not Initialized After Backup If Backing Up to Tape Device

BUG 2808539

If after image-journal backups were being done to tape, the backed up journal would not get properly initialized. The could lead to various issues, such as journaling being shutdown with a "journal is not empty" error. When that occurred, the journal state displayed by the DBO/DUMP/HEADER=JOURNAL command would show the following lines of output:

```
File is inaccessible
 journal has been made inaccessible by system
 journal is not empty
```

Other symptoms were also possible. The database recovery process (DBR) could fail with a bugcheck in DBR$RECOVER_ALL. Attempts to recover a database using an existing journal that was not properly re-initialized could fail with AIJCORRUPT errors. For example:

```
%DBO-W-AIJCORRUPT, journal entry 1451795/3364123 contains a new AIJBL that doesn't
 have the start flag set
```

This problem was introduced in Oracle CODASYL DBMS release 7.1.1 and only occurs with circular AIJs. Extensible After Image Journaling is not affected.

The problem can be avoided by backing up the journals to a disk destination or using extensible journaling.

This problem has now been corrected.

2.10 Constant Snapshot File Growth

With Oracle CODASYL DBMS it was possible for snapshot files to continually extend. This would occur after an abnormal user termination was handled by a database recovery (DBR) process. The problem was seen after installing Oracle CODASYL DBMS release 7.1.2.

There are two data structures in the database root (.ROO) file that are used to represent the state of a database user:

- The RTUPB list
- The TSNBLKs

In release 7.1.2, when the DBR process would clear the failed user's entries in the root file, it would neglect to clear the TSNBLK entry. That would cause DBMS to include the old user's last transaction sequence number (TSN) when determining what TSN should be granted to new snapshot (read only) transactions. Typically, the TSNBLK entry would soon get reused by another user and no symptoms of the problem would be seen. Occasionally, it was possible for the old TSNBLK entry to linger for quite some time. As long as the old TSN was still in the TSNBLK, pages in the snapshot files with TSNs greater than the old user's TSN could not be reclaimed. This would cause the snapshots files to grow since pages in the file could not be reused.

Take the following steps to determine if this problem is affecting a database:

1. Determine the TSN being granted to all snapshot transactions:

```
$ DBO/DUMP/HEADER/OPTION=DEBUG/OUTPUT=TEMP.TXT DB
$ SEARCH TEMP.TXT /WINDOW=(0,3) "Snapshot transaction in progress"
    Snapshot transaction in progress
    Last Process quiet-point was AIJ sequence 0
    Transaction sequence number is 0:3392
```

Note that in the above example the TSN is 3392.

2. See if there is a TSNBLK entry showing an active transaction with that TSN:

```
$ SEARCH TEMP.TXT 3392
    Transaction sequence number is 0:3392
  SLOT[1.] SIP TSN = 0:3392, COMMIT_TSN = 0:0.
  SLOT[2.] WIP TSN = 0:3392, COMMIT_TSN = 0:3390.
```

Note that in the above example there is a line that contains "WIP TSN = 0:3392". This indicates that there should be a user with a read-write transaction with sequence number 3392.

3. Confirm that there are no read-write transactions active with that TSN:

```
$ PIPE SEARCH /WINDOW=(0,3) TEMP.TXT "Read/write transaction in progress" | -
  SEARCH SYS$INPUT "0:3392"
%SEARCH-I-NOMATCHES, no strings matched
```

No read-write transactions exist with that TSN, thus the TSNBLK contains an invalid entry.

This problem can be corrected by forcing the TSNBLK entry to get reused. That can be done by adding multiple concurrent attaches to the database until the TSNBLK entry is reclaimed by one of the new users. For example:

```
DBQ> BIND DB ON STREAM 1;
DBQ> READY CONC UPDATE;
DBQ> BIND DB ON STREAM 2;
DBQ> READY CONC UPDATE;
...
```

Since each TSNBLK can only be used by one node in the cluster it may be necessary to do attaches from each node that currently has the database open. Continue to add attaches to the database until the "WIP TSN =" string noted above is no longer found in the DBO/DUMP/HEADER output.

This problem has been corrected in Oracle CODASYL DBMS Release 7.1.2.1. The TSNBLK is now properly cleared by the DBR when recovering a failed user.

3

Known Problems, Workarounds, and Documentation Errors

This chapter describes known problems, restrictions, and workarounds, as well as documentation errors and omissions for Oracle CODASYL DBMS release 7.1.2.1.

3.1 AIJ Log Server Process May Loop or Bugcheck

Under unknown but extremely rare conditions, on busy databases where the After Image Journal (AIJ) Log Server process is enabled, the ALS process has been observed to enter a loop condition writing AIJ information to the .AIJ files.

In the worst case, this problem could cause all available journal files to be filled with repeating data. If no remedial action were taken, this condition could cause the database to be shut down, and the AIJ journals to be considered inaccessable.

The database is not corrupted by this problem.

Stopping and restarting the ALS process will clear the looping condition, even if the ALS process must be stopped using the STOP/ID command.

Stopping the ALS process will not impact production as AIJ writes automatically revert to the non-ALS behaviour.

In this release, the behavior has been changed so that if this problem is detected, the ALS process will automatically shut down, producing a bugcheck dump file. This will prevent any danger of filling all available journals and ensure that the dabase remains available.

ALS may be safely restarted immediately as the conditions that cause such a loop are resolved during recovery of the ALS process.

3.2 VMS$MEM_RESIDENT_USER Rights Identier

Oracle CODASYL DBMS version 7.1 introduced additional privilege enforcement for the database or row cache qualifiers MEMORY_MAPPING=SYSTEM and LARGE_MEMORY. If a database utilizes any of these features, the user account that opens the database must be granted the VMS$MEM_RESIDENT_USER rights identifier. Also, any process attempting to change these attributes, or to convert or restore a database with these attributes enabled must also hold the same right.

Oracle recommends that the DBO/OPEN command be used when utilizing these features.

3.3 DBM$BIND_MAX_DBR_COUNT Documentation Clarification

The following is an updated description for the DBM$BIND_MAX_DBR_COUNT logical.

When an entire database is abnormally shut down (for example, due to a system failure), the database must be recovered in a node failure recovery mode. This recovery is performed by another monitor in the cluster if the database is opened on another node or is performed the next time the database is opened.

The DBM$BIND_MAX_DBR_COUNT logical name and the DBM_BIND_MAX_DBR_COUNT configuration parameter define the maximum number of database recovery (DBR) processes to be simultaneously invoked by the database monitor for each database during a node failure recovery. This logical name and configuration parameter apply only to databases that do not have global buffers enabled. Databases that utilize global buffers have only one recovery process started at a time during a node failure recovery.

In a node failure recovery situation with the Row Cache feature enabled (regardless of the global buffer state), the database monitor starts a single database recovery (DBR) process to recover the Row Cache Server (RCS) process and all user processes from the oldest active checkpoint in the database.

Per-Database Value

The DBM$BIND_MAX_DBR_COUNT logical name specifies the maximum number of database recovery processes to run at once for each database. For example, if there are 10 databases being recovered and the value for the DBM$BIND_MAX_DBR_COUNT logical name is 8, up to 80 database recovery processes would be started by the monitor after a node failure.

The DBM$BIND_MAX_DBR_COUNT logical name is translated when the monitor process opens a database. Databases must be closed and reopened for a new value of the logical to become effective.

4

New Features and Corrections in Oracle CODASYL DBMS Release 7.1.2

4.1 New Features for Release 7.1.2

This section contains new features and technical changes for Oracle CODASYL DBMS release 7.1.2.

4.1.1 Support for OpenVMS Version 8.2

This release of Oracle CODASYL DBMS, release 7.1.2, is the first 7.1-x release that supports OpenVMS Version 8.2. Prior releases of Oracle CODASYL DBMS 7.1 will not run on OpenVMS V8.2.

The minimum version of OpenVMS supported by this release is Version 7.2.

4.1.2 DBO /BACKUP /MULTITHREAD New ALLOCATION_QUANTITY Qualifier

An ALLOCATION_QUANTITY=number-blocks qualifier has been added to the DBO /BACKUP /MULTITHREAD command. This qualifier specifies the size, in blocks, which is initially allocated to the backup file. The minimum value for the number-blocks parameter is 1; the maximum value allowed is 2147483647. If you do not specify the ALLOCATION_QUANTITY qualifier, the EXTEND_QUANTITY value effectively controls the initial allocation of the file.

The ALLOCATION_QUANTITY qualifier cannot be used with backup to tape operations.

4.1.3 DBM$BIND_SNAP_QUIET_POINT Logical Reinstated

Over the years, various problems have been reported related to quiet point backups. In particular, database backups and journal backups would sometimes fail with the following error:

```
%DBO-F-TIMEOUT, timeout on quiet
```

Quiet point backups are required so that recovery of a journal can be done without always requiring previous journals. Full database backups can avoid lock conflicts if they wait for the quiet point lock. See the documentation for the DBO/BACKUP commands for more information regarding quiet point backups.

Lock timeout errors for the quiet lock are intended to be returned when a long running update transaction has not completed within the timeout period. However, Oracle CODASYL DBMS Release 6.0 forced all transactions (read-only or read-write) to obtain the quiet point lock when starting. That greatly increased the incidence of timeout errors from backups. To remedy this situation the logical DBM$BIND_SNAP_QUIET_POINT was implemented to force processes starting read-only transactions to release the quiet point lock. If that logical was defined to the value 0, read-only transactions would not obtain the quiet point lock.

Defining the DBM$BIND_SNAP_QUIET_POINT logical to 0 would usually resolve problems with timeouts on the quiet lock, but it would effectively disable the fast commit performance feature for applications that often switched between read-only and read-write transactions. (See BUG 884004.) To remedy that situation the transaction start code was modified to retain the quiet lock during read-only transactions, but release the lock during the read-only transaction if a backup process started. With that change the DBM$BIND_SNAP_QUIET_POINT lock logical could be defined without impacting the performance of the fast commit feature. The change was introduced in releases 7.0.4.3 and 7.1.0.

That fix resolved performance problems, but the logical still could not be defined to 0 if the hot standby feature was being used. If the hot standby feature was being utilized then the logical had to be undefined, or defined to the default value of 1. Applications that utilized hot standby were still subject to undeserved timeouts from backup commands. In releases 7.0.5 and 7.1.1 the hot standby feature was modified so that it did not require read-only transactions to hold the quiet point lock. Also, because read-only processes would usually release the quiet point lock when a backup started, the DBM$BIND_SNAP_QUIET_POINT logical was completely removed.

After these changes, quiet lock timeouts were no longer an issue for most applications. However, a read-only transaction would only release the quiet point lock when Oracle CODASYL DBMS had returned control to the user application. Some complicated queries could execute for an exceptionally long period of time before returning a record to the user application and thus might not release the quiet lock in time to prevent timeout errors for the backup process. In addition, any update transaction that started after the backup process requested the quiet point lock would stall until the long running read-only request returned control to the user application. That means that it was possible for many database users to stall waiting for the read-only transaction to complete, even though they had released the quiet point lock.

This release of Oracle CODASYL DBMS reinstates the DBM$BIND_SNAP_QUIET_POINT logical so that read-only transactions can be forced to release the quiet point lock before starting. The logical now has a slightly different meaning than the original implementation. The default value is still 1, but that value now signifies that all transactions will hold the quiet point lock until a backup process requests it. Read-only transactions will not obtain the quiet point lock; only read-write requests will obtain the quiet point lock. This is the behavior that was introduced in response to BUG 884004. If the logical is defined to be 0, read-only transactions will always release the quiet point lock at the beginning of the transaction, regardless of the existence of a backup process. That implies that all modified buffers in the buffer pool have to be written to disk before the transaction proceeds. Applications that utilize the fast commit feature and often switch between read-only and read-write transactions within a single attach may experience performance degradation if the logical is defined to 0. This is the behavior that was in place prior to releases 7.0.4.3 and 7.1.0.

Oracle Corporation recommends that you do not define the DBM$BIND_SNAP_QUIET_POINT logical for most applications. If you encounter the scenario described in BUG 3908414, you can define the logical to be 0 to force read-only transactions to always release the quiet point lock. Rather than define the logical system wide, this logical can be defined for specific jobs that are likely to execute for an extensive period of time before returning to the user application. This parameter may also be manipulated from the DBO /SHOW STATISTICS locking dashboard. That way most users will not have

to sacrifice fast commit performance when switching between read-only and read-write transactions. As of releases 7.0.5 and 7.1.1, hot standby is no longer affected by this logical, so hot standby is not a factor when determining how to define the logical.

4.1.4 DBO Operator Notification Syntax Change

The DBO syntax to enable or disable system notification for certain database events was changed in Oracle CODASYL DBMS release 7.1.0. Due to an omission, this new syntax was never documented.

In versions of Oracle CODASYL DBMS prior to 7.1.0, the DBO operator notification facility was used to provide notification of after-image journal changes or problems that may occur during normal database activity. Refer to the *Oracle CODASYL DBMS Database Administration Reference Manual* for more information. The DBO/CREATE and DBO/MODIFY syntax reflected this association between notifications and journaling:

Old Syntax:

```
$ DBO/CREATE/JOURNAL_OPTIONS=([NO]NOTIFY=(operator-name) db-name
$ DBO/MODIFY/JOURNAL_OPTIONS=([NO]NOTIFY=(operator-name) db-name
```

Operator-name was a list of one or more standard VMS operator classes:

- CENTRAL
- CLUSTER
- CONSOLE
- DISKS
- OPER1
- OPER2
- OPER3
- OPER4
- OPER5
- OPER6
- OPER7
- OPER8
- OPER9
- OPER10
- OPER11
- OPER12
- SECURITY

Starting with Oracle CODASYL DBMS release 7.1.0, operator notification has been expanded to include non-journal related events (refer to the *Oracle CODASYL DBMS Release Notes, Release 7.1* for more details), including:

- Bugcheck notification
- Corrupt Page Table Additions
- Storage Area Extensions

- AIJ Fullness
- Server startup and termination messages

The old syntax is now obsolete. The syntax for enabling or disabling system notification has been changed to correspond to this new database-wide behavior.

New Syntax:

```
$ DBO/CREATE/ALERT_OPERATOR=(ENABLED=operator-name)  db-name
$ DBO/CREATE/ALERT_OPERATOR=(DISABLED=operator-name) db-name

$ DBO/MOD/ALERT_OPERATOR=(ENABLED=operator-name)  db-name
$ DBO/MOD/ALERT_OPERATOR=(DISABLED=operator-name) db-name
```

The ENABLED and DISABLED keywords can be combined within the same DBO command. The list of valid operator-names remains unchanged.

4.2 Corrections in Release 7.1.2

This section describes software errors corrected in Oracle CODASYL DBMS release 7.1.2.

4.2.1 Logical Names Translated Twice

Starting with Oracle CODASYL DBMS release 7.1, many logical name translation requests within DBMS components were incorrectly performed two times.

This problem has been corrected. Logical names are no longer translated twice.

4.2.2 Incorrect Backup of Empty Single AIJ File

Starting with Oracle CODASYL DBMS release 7.1.1, if you backed up an empty single extendable AIJ file, that is, an AIJ with only an open record, the "last commit TSN" field in the open record was incorrectly reset to zero as shown below:

```
$ dbo/dump/after/nodata foo_aij.aij

1/1             TYPE=O, LENGTH=510, TAD=24-SEP-2004 04:48:43.32, CSM=00
    Database MYDISK:[MYDB]FOO.ROO;2
    Database timestamp is 24-SEP-2004 04:48:41.11
    Facility is "DBMAIJ ", Version is 711.1
    Database version is 71.0
    AIJ Sequence Number is 0
    Last Commit TSN is 0:0
    .
    .
    .
```

If you tried to recover the AIJ file, the recovery process ignored all transactions from the file as shown below :

```
$ dbo/recover foo_aij.aij
%DBO-I-LOGRECDB, recovering database file MYDISK:[MYDB]FOO.ROO;2
%DBO-I-LOGOPNAIJ, opened journal file MYDISK:[MYDB]FOO_AIJ.AIJ;1 at 24-SEP-2004 05:07:56.35
%DBO-I-LOGRECSTAT, transaction with TSN 0:128 ignored
%DBO-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%DBO-I-LOGRECOVR,  0 transactions committed
%DBO-I-LOGRECOVR,  0 transactions rolled back
%DBO-I-LOGRECOVR,  1 transaction ignored
   .
   .
   .
```

To avoid the problem, you can use multiple circular AIJs instead of a single extendable AIJ.

When you back up an empty single extendable AIJ file, no backup file is produced, and the AIJ open record is left unchanged. The AIJ roll forward works as expected.

This problem has been corrected in Oracle CODASYL DBMS release 7.1.2.

4.2.3 DBO/SHOW AFTER_JOURNAL/BACKUP_CONTEXT Reset DBM$AIJ_BACKUP_SEQNO

In previous releases the DBO/SHOW AFTER_JOURNAL/BACKUP_CONTEXT command reset the DBM$AIJ_BACKUP_SEQNO symbol to -1 while the DBO /BACKUP/AFTER command was setting it correctly:

```
$ dbo/backup/after/nolog db ""
$ sh symbol DBM$AIJ_BACKUP_SEQNO
  RDM$AIJ_BACKUP_SEQNO == "0"
$ dbo/show after/backup/out=tdb:x.x db
$ sh symbol DBM$AIJ_BACKUP_SEQNO
  DBM$AIJ_BACKUP_SEQNO == "-1"
```

The DBO/SHOW AFTER_JOURNAL/BACKUP_CONTEXT command now sets the DBM$AIJ_BACKUP_SEQNO symbol correctly:

```
$ dbo/backup/after/nolog db ""
$ sh symbol DBM$AIJ_BACKUP_SEQNO
  DBM$AIJ_BACKUP_SEQNO == "0"
$ dbo/show after/backup/out=tdb:x.x db
$ sh symbol DBM$AIJ_BACKUP_SEQNO
  DBM$AIJ_BACKUP_SEQNO == "0"
```

This problem has been corrected in Oracle CODASYL DBMS release 7.1.2.

4.2.4 DBO /SHOW LOCKS Limits Relaxed

Previously, the LOCK= and PROCESS= qualifiers of the DBO /SHOW LOCKS command were limited to 32 specified values.

This problem has been corrected. The LOCK= and PROCESS= qualifiers of the DBO /SHOW LOCKS command now accept up to 256 values each.

4.2.5 NOREQIDT Error After Many Attach/Detaches

An application that utilized lock timeouts (for example, the WAIT clause of the READY statement) and often disconnected from a database could eventually encounter the following error:

```
%DBM-F-NOREQIDT, reached internal maximum number of simultaneous timer requests
```

This problem could be reproduced by creating two login sessions. The first session would lock a record. The second session would repeatedly attempt to access the same record. After each attempt it would disconnect from the database and attach again.

This problem can be avoided by not repeatedly detaching from and attaching to the database within a single invocation of a database application, or by not using the lock timeout feature.

This problem has been corrected in Oracle CODASYL DBMS release 7.1.2.

4.2.6 Processes Hang in HIB State

Oracle CODASYL DBMS applications that hibernate may occasionally hang in HIB state if after-image journaling (AIJ) is enabled. Applications utilizing the OpenVMS POSIX Threads Library will often hibernate, so threaded applications are especially vulnerable to this issue.

This problem could occur if a user application had executed the OpenVMS $HIBER system service and one of the following Oracle CODASYL DBMS events occurred while the process was hibernating:

- A Global Checkpoint request was issued by a journal switch or a DBO /CHECKPOINT command.
- A checkpoint timer expired.
- A two-phase commit (2PC or DECdtm) transaction that involved the process was ended.

If the OpenVMS POSIX Threads Library is being utilized, the OpenVMS remedial kit VMS732_SYS-V0600 may also be required to completely resolve this issue.

This problem can be avoided by disabling journaling. Oracle Corporation does not recommend disabling journaling unless another suitable database recovery method is available.

This problem has been corrected in Oracle CODASYL DBMS release 7.1.2.

4.2.7 DBR Bugchecks at DBR$DDTM_RESOLVE + 000005A8

If a process was involved in a two-phase commit (2PC or DECdtm) transaction, and it failed after the prepare phase of a transaction but before the commit phase, and the database did not have the FAST COMMIT feature enabled, it was possible for the database recovery process (DBR) to fail with the following exception:

```
***** Exception at 00071858 : DBR$DDTM_RESOLVE + 000005A8
%COSI-F-BUGCHECK, internal consistency failure
```

The problem was introduced in Oracle CODASYL DBMS releases 7.0.5.1 and 7.1.1.

Enabling the FAST COMMIT feature will prevent this problem.

This problem has been corrected in Oracle CODASYL DBMS release 7.1.2.

4.2.8 DBR Bugchecks at DBR$WAKE_ALL + 000000F4

Previously it was possible for a database recovery process (DBR) to fail with the following exceptions:

```
***** Exception at 00088C54 : DBR$WAKE_ALL + 000000F4
%COSI-F-UNEXPERR, unexpected system error
-SYSTEM-F-REMRSRC, insufficient system resources at remote node

***** Exception at 00088C54 : DBR$WAKE_ALL + 000000F4
%COSI-F-UNEXPERR, unexpected system error
-SYSTEM-F-NOSUCHTHREAD, specified kernel thread does not exist
```

The DBR did not anticipate the possibility of those errors occurring when it issued the OpenVMS $WAKE system service and would terminate when those error status codes were returned by the system service.

This problem has been corrected in Oracle CODASYL DBMS release 7.1.2.

4.2.9 Database Corruption When 2PC Transaction Fails

It was possible for a database to become corrupt when the following conditions were met:

- The FAST COMMIT feature was enabled
- A failing process was a participant in a two-phase commit (2PC) transaction
- The process failed after completing the prepare phase of a 2PC transaction but before completing the commit phase
- Other participants in the same transaction failed before completing the prepare phase

In this situation the database recovery process (DBR) would neglect to roll back the failed transaction.

This problem was introduced in Releases 7.0.5.1 and 7.1.1. There is no workaround for this problem.

This problem has been corrected in Oracle CODASYL DBMS release 7.1.2.

4.2.10 DBMS Hangs with No Stall Messages

When Oracle CODASYL DBMS called RMS for asynchronous operation, if the call to RMS returned an error, Oracle CODASYL DBMS would hang waiting for a completion AST from RMS. Often, no stall message would be displayed by DBO/SHOW STATISTICS.

For example, if Oracle CODASYL DBMS was using a temporary file on disk, and it did not have sufficient ASTLM quota, it could hang when attempting to read from or write to the temporary file. Oracle CODASYL DBMS may have issued many asynchronous disk writes (ABW) to flush modified page buffers to disk, consuming all available ASTLM quota. While that I/O was underway, if Oracle CODASYL DBMS attempted to access a temporary RMS file, it would not properly handle the error returned by RMS and would hang.

Oracle CODASYL DBMS will now properly trap RMS errors and report them to the application. For example, the following error may be returned if quota is exhausted:

```
%DBO-F-IO_ERROR, input or output error
-COSI-F-READERR, read error
-RMS-F-CDA, cannot deliver AST
```

This problem can be avoided by ensuring that users have sufficient DIOLM and ASTLM quota to concurrently flush all allocated page buffers, while also having enough additional quota to do I/O to other files that are used by Oracle CODASYL DBMS, such as journals, temporary files, etc. For example, if a process is using 500 page buffers, the minimum quota values for DIOLM and ASTLM would be greater than 500.

This problem has been corrected in Oracle CODASYL DBMS release 7.1.2.

4.2.11 %DBM-F-BADPROTOCOL Error on Remote Access

It was possible, although unlikely, that a remote operation would return a %DBM-F-BADPROTOCOL message from DBMSERVER.

This error would only occur if the amount of data being sent by the client application happened to be an exact multiple of the buffer size used to transfer information to the server.

This problem would only affect remote access and would be most likely to occur while trying to bind to a database using a very large subchema. There is no workaround, although using a different subschema would generally succeed.

This problem has been fixed in Oracle CODASYL DBMS release 7.1.2.

4.2.12 Problems Mixing Stream and Non-Stream DML within the Same Image

In previous versions of Oracle CODASYL DBMS, problems could arise running applications comprised of stream and non-stream modules linked together into the same image. The applications could generate run-time errors or produce wrong results. Specifically, starting with release 7.0.5.1, a '%DBM-F-ID_MAP, ID number mapping' run-time error may be returned.

This problem has been fixed in Oracle CODASYL DBMS release 7.1.2. You can mix stream and non-stream DML modules in the same image.

Applies only to embedded DML applications

This fix applies only to modules containing embedded DML. There is still a long-standing restriction mixing stream and non-stream modules containing callable DBQ. Oracle will consider removing this restriction in a future release.

The following example illustrates the problem behavior which occurred in previous releases:

```
$ CREATE MAIN.FOR
$DECK
C       MAIN.FOR:

        PROGRAM MAIN

        external  START_DEFAULT
        external  FETCH_DEFAULT
        external  END_DEFAULT
        external  START_STREAM
        external  FETCH_STREAM
        external  END_STREAM


        CALL START_DEFAULT ()
        CALL START_STREAM ()

        CALL FETCH_DEFAULT ()
        CALL FETCH_STREAM ()
        CALL FETCH_DEFAULT ()

        CALL END_DEFAULT ()
        CALL END_STREAM ()

        END
$EOD
$ CREATE STREAM.FOR
$DECK
C       ------------------------------------------------------------------------
C       STREAM :
C       ------------------------------------------------------------------------

C       ------------------------------------------------------------------------
       SUBROUTINE START_STREAM ()
```

```
      INVOKE (SCHEMA=PARTS,
     1       SUBSCHEMA=SUB2,
     2       DATABASE=PARTS,
     3       STREAM = 11)

      PRINT *, 'READY (STREAM)'
      READY (CONCURRENT, UPDATE)

      RETURN
      END

C     ------------------------------------------------------------------------
      SUBROUTINE FETCH_STREAM ()

      INVOKE (SCHEMA=PARTS,
     1       SUBSCHEMA=SUB2,
     2       DATABASE=PARTS,
     3       STREAM = 11)

      PRINT *, 'FETCH NEXT EMPLOYEE (STREAM)'
      FETCH (NEXT, RECORD = EMPLOYEE)
      PRINT *,EMP_ID

      RETURN
      END

C     ------------------------------------------------------------------------
     SUBROUTINE END_STREAM ()

      INVOKE (SCHEMA=PARTS,
     1       SUBSCHEMA=SUB2,
     2       DATABASE=PARTS,
     3       STREAM = 11)

      PRINT *, 'ROLLBACK (STREAM)'
      ROLLBACK(STREAM)

      RETURN
      END

C     ------------------------------------------------------------------------
$EOD

$ CREATE DEFAULT.FOR
$DECK
C     ------------------------------------------------------------------------
C     DEFAULT STREAM
C     ------------------------------------------------------------------------
      SUBROUTINE START_DEFAULT ()

      INVOKE (SCHEMA=PARTS,
     1       SUBSCHEMA=SUB1,
     2       DATABASE=PARTS)

      PRINT *,'READY (DEFAULT)'
      READY (CONCURRENT, UPDATE)

      RETURN
      END
C     ------------------------------------------------------------------------
      SUBROUTINE FETCH_DEFAULT ()

      INVOKE (SCHEMA=PARTS,
     1       SUBSCHEMA=SUB1,
     2       DATABASE=PARTS)

      PRINT *,'FETCH NEXT CLASS (DEFAULT)'
      FETCH (NEXT, RECORD = CLASS)
      PRINT *,CLASS_CODE
```

```
        RETURN
        END
C       ------------------------------------------------------------------------
        SUBROUTINE END_DEFAULT ()

        INVOKE (SCHEMA=PARTS,
     1       SUBSCHEMA=SUB1,
     2       DATABASE=PARTS)

        PRINT *,'ROLLBACK (DEFAULT)'
        ROLLBACK

        RETURN
        END
$EOD
$
$!-----------------------------------------------------------------------------
$ CREATE SUB1.DDL
$DECK

SUBSCHEMA NAME IS SUB1 FOR PARTS SCHEMA

REALM MAKE
    IS MAKE

REALM BUY
    IS BUY

RECORD NAME IS CLASS
    ITEM CLASS_CODE TYPE IS CHARACTER 2
    ITEM CLASS_DESC TYPE IS CHARACTER 20
    ITEM CLASS_STATUS TYPE IS CHARACTER 1

RECORD NAME IS PART
    ITEM PART_ID TYPE IS CHARACTER 8
    ITEM PART_DESC TYPE IS CHARACTER 50
    ITEM PART_STATUS TYPE IS CHARACTER 1
    ITEM PART_PRICE TYPE IS FLOATING
    ITEM PART_COST TYPE IS FLOATING
    ITEM PART_SUPPORT TYPE IS CHARACTER 2

SET NAME IS ALL_CLASS

SET NAME IS ALL_PARTS

SET NAME IS CLASS_PART

$EOD

$!-----------------------------------------------------------------------------
$ CREATE SUB2.DDL
$DECK

SUBSCHEMA NAME IS SUB2 FOR PARTS SCHEMA

REALM PERSONNEL
    IS PERSONNEL

RECORD NAME IS EMPLOYEE
    ITEM EMP_ID TYPE IS CHARACTER 5
    ITEM EMP_LAST_NAME TYPE IS CHARACTER 20
    ITEM EMP_FIRST_NAME TYPE IS CHARACTER 10
    ITEM EMP_PHONE TYPE IS CHARACTER 7
    ITEM EMP_LOC TYPE IS CHARACTER 5

RECORD NAME IS DIVISION
    ITEM DIV_NAME TYPE IS CHARACTER 20

SET NAME IS ALL_EMPLOYEES

SET NAME IS MANAGES

SET NAME IS CONSISTS_OF
```

```
$EOD
$!-------------------------------------------------------------------------------
$ DBO/RESTORE PARTS.DBB
$ DBO/EXPORT PARTS PARTS.DBM
$ DDL/COMPILE/EXPORT=PARTS.DBM SUB1,SUB2
$ DBO/MODIFY/IMPORT=PARTS.DBM/SUB=(SUB1,SUB2) PARTS
$
$ FORTRAN/LIST/NOOPT MAIN.FOR
$ FORTRAN/DML/LIS/NOOPT     STREAM.FOR
$ FORTRAN/DML/LIS/NOOPT     DEFAULT.FOR
$ LINK/MAP MAIN,DEFAULT,STREAM,SYS$LIBRARY:DBMDML/opt
$!-------------------------------------------------------------------------------
```

If you build and run MAIN.EXE, you would see the following incorrect results:

```
READY (DEFAULT)
READY (STREAM)
FETCH NEXT CLASS (DEFAULT)
FETCH NEXT EMPLOYEE (STREAM)
12333
FETCH NEXT CLASS (DEFAULT)
ROLLBACK (DEFAULT)
ROLLBACK (STREAM)
```

There is no CLASS record returned from the FETCH_DEFAULT module. In this example, if you were to switch the order of START_DEFAULT and START_ STREAM calls in MAIN.FOR, you would get a totally different (and correct) answer:

```
READY (STREAM)
READY (DEFAULT)
FETCH NEXT CLASS (DEFAULT)
BU
FETCH NEXT EMPLOYEE (STREAM)
75624
FETCH NEXT CLASS (DEFAULT)
BT
ROLLBACK (DEFAULT)
ROLLBACK (STREAM)
```