VAX Rdb/VMS

Mandatory Update for Versions 3.1B and 4.0

August 1991

This is a mandatory update for all customers currently using VAX Rdb/VMS Version 3.1B and Version 4.0.

This document contains the installation verification procedure (IVP) for installing updated images and the VAX Rdb/VMS release notes that describe the problems fixed in these images, additional known problems not fixed, and additional restrictions for VAX Rdb/VMS Version 3.1B and Version 4.0.

The release notes sections in this document contain descriptions of and other information relating to this mandatory update only. These release notes supplement but do not supersede the VAX Rdb/VMS release notes shipped with VAX Rdb/VMS Version 3.1, Version 3.1B, and Version 4.0.

If you are installing this mandatory update, read the instructions contained in this document.

Operating System:	VMS
Software Version:	Mandatory Update for VAX Rdb/VMS Version 3.1B and Version 4.0

Digital Equipment Corporation
Maynard, Massachusetts

August 1991

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

© Digital Equipment Corporation 1991.

All Rights Reserved.
Printed in U.S.A.

The postpaid Reader's Comment form at the end of this document requests your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation: ACMS, ALL–IN–1, CDD/Plus, DEC, DECdecision, DECdtm, DECforms, DECintact, DECnet, DECtp, DECtrace, DECwindows, MicroVAX, PATHWORKS, Rdb/VMS, ULTRIX, UNIBUS, VAX, VAX Ada, VAX BASIC, VAX C, VAX CDD, VAX COBOL, VAX DATATRIEVE, VAX DBMS, VAX DOCUMENT, VAX FMS, VAX FORTRAN, VAX Pascal, VAX RALLY, VAX Rdb/ELN, VAX RMS, VAX SCAN, VAX SPM, VAXcluster, VAXELN, VAXset, VAXstation, VAX TEAMDATA, VIDA, VMS, VT, and the DIGITAL logo.

AppleTalk and Macintosh are registered trademarks of Apple Computer, Inc. MPW is a trademark of Apple Computer, Inc. Motorola and 68000 are registered trademarks of Motorola, Inc. MS–DOS is a registered trademark of Microsoft Corporation. OS/2 and IBM are trademarks of International Business Machines Corporation.

This document is available on CDROM.

This document was prepared using VAX DOCUMENT, Version 2.0.

Contents

Preface xiii

Part I Mandatory Update for VAX Rdb/VMS Version 3.1B

1 Installing the Rdb/VMS Version 3.1B Mandatory Update

1.1 Before Installing the Rdb/VMS Mandatory Update Package for V3.1B . . . 1–1
1.1.1 Prerequisite Hardware and Software 1–1
1.1.2 Back Up All Existing Rdb/VMS Databases 1–2
1.1.3 Disk Space Required to Install the MUP 1–2
1.1.4 Shut Down the Rdb/VMS Monitor 1–2
1.1.5 Obtain the VMS Privileges Required to Install the MUP 1–3
1.1.6 Ensure Sufficient Process Account Quotas to Install the MUP 1–3
1.1.7 Obtain System Parameter Values Required to Install the MUP 1–4
1.1.7.1 Checking System Parameter Values 1–5
1.1.7.2 Calculating the Values for GBLPAGES and GBLSECTIONS 1–5
1.1.7.3 Changing System Parameter Values with AUTOGEN 1–6
1.1.7.4 Setting Dynamic System Parameters 1–6
1.1.8 Back Up Your System Disk 1–7
1.1.9 Avoid Giving Users Access to HELP 1–7
1.1.10 Prevent Interactive Users from Gaining Access to the System 1–7
1.2 Installing the Mandatory Update Package 1–8
1.2.1 Time Required to Install the MUP 1–8
1.2.2 Invoking VMSINSTAL 1–8
1.2.3 Steps of the Installation Procedure 1–9
1.2.4 Completing the Installation Procedure 1–11
1.2.5 Errors That Cause the Installation to Fail 1–12
1.3 After Installing the Mandatory Update Package 1–13
1.3.1 Accessing the Online Release Notes 1–13
1.3.2 Tailoring Your System 1–13
1.3.3 Returning the System to Original Settings 1–14

2 Software Errors Fixed in the Mandatory Update for Rdb/VMS V3.1B

2.1 General Information 2–1
2.1.1 A VMS Sort Utility Problem Affected Rdb/VMS 2–1
2.1.2 The VMS Sort Utility for VMS V5.1, V5.2, and V5.3 Caused Problems with Rdb/VMS Databases 2–2
2.2 Software Errors Fixed That Apply to All Interfaces in the Mandatory Update for Version 3.1B 2–2
2.2.1 Rdb/VMS Returned Errors When Retrying Failed Multidatabase Transactions 2–3
2.2.2 Active Transactions in Application Programs Could Not Recover from Network Failures 2–3

2.2.3 Data Transfer from the V3.1B Server Caused Problems 2–3
2.2.4 A Partitioned Sorted Index Stored the First Record Incorrectly 2–3
2.2.5 Certain Queries with Intended MODIFY Operations Within Read/Write Transactions Caused Unnecessary Writes to the .AIJ File 2–4
2.2.6 Partitioned Sorted Indexes Caused Bugchecks 2–4
2.2.7 A MODIFY Operation Caused Index Corruption on Partitioned Hash Indexes 2–4
2.2.8 Partitioned Sorted Indexes Resulted in Various Problems 2–5
2.2.9 With Compression Disabled, Altering the Storage Map STORE Clause and Then Selecting a Row Resulted in a Bugcheck 2–5
2.2.10 A Bugcheck Sometimes Resulted When a Sorted Index Rebalanced Itself 2–6
2.2.11 Problem Occurred When Rdb/VMS and the User Application Both Allocated Event Flag 63 2–6
2.2.12 Query Returned Records in Wrong Order with the SQL ORDER BY <column-name> DESCENDING or the RDO SORTED BY DESCENDING <field> Clauses 2–7
2.2.13 Negate Operator Incorrectly Propagated the NULL Bit While Processing a Record Stream 2–8
2.2.14 Query with Computed-By and OR Index Retrieval Strategy Returned Incorrect Results 2–9
2.3 RDO, RDBPRE, and RDML Problems Fixed in the Mandatory Update for V3.1B 2–9
2.3.1 RDBPRE Generated Incorrect Code for Request Handles 2–10
2.4 RMU Problems Fixed in the Mandatory Update for V3.1B 2–10
2.4.1 RMU/VERIFY Returned Spurious Errors Involving Fragmented Records 2–11
2.4.2 Attempting to Recover a Database from an .AIJ File Using the RMU/RECOVER Command or RDO RECOVER Statement Caused an Exception Condition 2–11

3 Known Problems, Restrictions, and Other Notes for the Mandatory Update for Rdb/VMS V3.1B

3.1 Problems, Restrictions, and Notes for All Interfaces 3–1
3.1.1 Using Quoted Threshold Values for Binary Data Types for Partitioning Data or Indexes Results in Data or Index Corruption . . . 3–1
3.1.2 Problem with SQL LIKE and RDO MATCHING Clauses 3–2
3.1.3 RDB$REMOTE Account Has SYSTEM as Owner 3–3
3.1.4 RDMSHRP_DS Image Displays Incorrect Values 3–3
3.1.5 An Arithmetic Exception Results When Joining Integer Columns 3–3
3.1.6 Collating Sequences That Use Two-to-Two Character Mapping May Bugcheck 3–3
3.1.7 Synchronization Problem for an Empty Sorted Index 3–4
3.1.8 Rdb/VMS Does Not Accept the Database File Specification in a Logical Name 3–5
3.1.9 Constraints Cause Looping and LCKCCH$COMMIT_SUBTREE Bugchecks 3–5
3.1.10 Query Optimizer Does Not Use Index-Only Retrieval When the Dbkey Is Selected 3–6
3.1.11 Query Optimizer Chooses an Incorrect Strategy for a Write Operation Within a Selection Loop and Goes into an Infinite Loop 3–7
3.1.12 Singleton Subselect Statement Returns Incorrect Results 3–8

3.1.13 SPAM Pages Are Not Updated Correctly 3–9
3.1.14 Rdb/VMS Monitor Fails When the Last User Finishes on a Particular Database 3–9
3.1.15 Triggers That Affect Subject Table Rows Can Cause Loops or Inconsistent Results 3–9
3.1.16 Query Using Descending Indexes Returns Incorrect Results 3–10
3.1.17 Query with a Computed-By Field and OR Logic Returns Incorrect Results 3–10
3.1.18 NOWAIT Transactions Have Their Buffers Invalidated at COMMIT 3–10
3.2 SQL Problems, Restrictions, and Notes 3–11
3.2.1 Using the IGNORE CASE Option of the LIKE Clause Sometimes Results in a Query That Incorrectly Returns No Rows 3–11
3.3 SQL/Services Problems, Restrictions, and Notes 3–11
3.3.1 SQL/Services VMS API Shipped with the Rdb/VMS Run-Time Kit . . . 3–11
3.4 RDO, RDBPRE, and RDML Problems, Restrictions, and Notes 3–12
3.5 Rdb/VMS Management Utility (RMU) Problems, Restrictions, and Notes 3–12
3.5.1 Do Not Delete After-Image Journal (.AIJ) Backup Files If the AIJ Backup Fails or Is Terminated 3–12
3.5.2 EXPORT Operations Fail with an Access Violation When the Database Has a Default Collating Sequence Defined 3–13
3.5.3 Use of Undocumented RMU/REPAIR Command Corrupts Databases 3–13
3.6 Notes and Restrictions Related to CDD/Plus 3–13
3.6.1 Restrictions Lifted by CDD/Plus Version 4.3 3–13
3.7 Rdb/VMS Documentation Errors and Omissions in V3.1B 3–13
3.7.1 Documentation Error in V3.1 *VAX Rdb/VMS SQL Reference Manual*, Appendix D.4 3–13

4 Optional ECO Patches for the Mandatory Update for VAX Rdb/VMS V3.1B

4.1 Optional ECO Patches That Can Be Applied to the Mandatory Update for Rdb/VMS V3.1B 4–1
4.1.1 RDMSHRP ECO 1: Constraints Cause Looping and LCKCCH$COMMIT_SUBTREE Bugchecks 4–1
4.1.2 RDMSHRP ECO 14: Query Optimizer Does Not Use Index-Only Retrieval When the Dbkey Is Selected 4–1
4.1.3 RDMSHRP ECO 19: Query Optimizer Chooses an Incorrect Strategy for a Write Operation Within a Selection Loop and Goes into an Infinite Loop 4–2
4.1.4 RDMMON ECO 1: Rdb/VMS Monitor Fails When the Last User Finishes on a Particular Database 4–2

Part II Mandatory Update for VAX Rdb/VMS Version 4.0

5 Installing the Rdb/VMS Mandatory Update for Version 4.0

5.1 Before Installing the Rdb/VMS Mandatory Update Package for Rdb/VMS Version 4.0 5–1
5.1.1 Prerequisite Hardware and Software 5–1
5.1.2 Back Up All Existing Rdb/VMS Databases 5–2
5.1.3 Disk Space Required to Install the MUP 5–2
5.1.4 Shut Down the Rdb/VMS Monitor 5–2
5.1.5 Obtain VMS Privileges Required to Install the MUP 5–3
5.1.6 Ensure Sufficient Process Account Quotas to Install the MUP 5–3
5.1.7 Obtain System Parameter Values Required to Install the MUP 5–4
5.1.7.1 Checking System Parameter Values 5–5
5.1.7.2 Calculating the Values for GBLPAGES and GBLSECTIONS 5–5
5.1.7.3 Changing System Parameter Values with AUTOGEN 5–6
5.1.7.4 Setting Dynamic System Parameters 5–6
5.1.8 Back Up Your System Disk 5–7
5.1.9 Avoid Giving Users Access to HELP 5–7
5.1.10 Prevent Interactive Users from Gaining Access to the System 5–7
5.2 Installing the Mandatory Update Package 5–8
5.2.1 Time Required to Install the MUP 5–8
5.2.2 Invoking VMSINSTAL 5–8
5.2.3 Steps of the Installation Procedure 5–9
5.2.4 Completing the Installation Procedure 5–12
5.2.5 Errors That Cause the Installation to Fail 5–12
5.3 After Installing the MUP 5–13
5.3.1 Accessing the Online Release Notes 5–13
5.3.2 Tailoring Your System 5–13
5.3.3 Returning the System to Original Settings 5–15

6 Software Errors Fixed in the Mandatory Update for Rdb/VMS V4.0

6.1 General Information 6–1
6.1.1 The VMS Sort Utility for VMS V5.1, V5.2, and V5.3 Caused Problems with Rdb/VMS Databases 6–1
6.2 Software Errors Fixed That Apply to All Interfaces in the Mandatory Update for Version 4.0 6–1
6.2.1 Wrong RDBINTSHR.EXE Image Was Installed for Interactive License Customers 6–2
6.2.2 Active Transactions in Application Programs Could Not Recover from Network Failures 6–2
6.2.3 Using Event Flags Caused Conflicts with Other Software Products . . 6–2
6.2.4 An Access Violation Resulted When DECdtm Services and DECnet Services Were Not Running 6–2
6.2.5 If a Commit Failed During a One-Phase Commit Protocol When an Explicit Distributed Transaction Was Run, It Caused a Premature $FINISH_RMOP to DECdtm 6–3
6.2.6 The Rdb/VMS DISTRIBTRAN Privilege Was Not Available for Remote Database Access 6–3
6.2.7 A Partitioned Sorted Index Stored the First Record Incorrectly 6–3
6.2.8 A MODIFY Operation Caused Index Corruption on Partitioned Hash Indexes 6–4
6.2.9 Partitioned Sorted Indexes Resulted in Various Problems 6–4
6.2.10 You Could Not Define Views Based on System Relations 6–5
6.2.11 An Incorrect Value Was Stored or a Bugcheck Resulted When Using BEFORE UPDATE or BEFORE MODIFY Triggers 6–5

6.2.12 Queries with Computed Expressions and Indexes Returned the Wrong Results . . . 6–6
6.2.13 Queries with Computed Expressions Returned the Wrong Results . . . 6–6
6.2.14 Some Update-Intensive Applications Experienced a Performance Degradation in Rdb/VMS V4.0 Compared to V3.1B . . . 6–7
6.2.15 Poor Performance Was Experienced While Retrieving Views by Dbkey . . . 6–7
6.2.16 Wrong Results Were Returned from Queries That Used Collating Sequences and the STARTING WITH "" Relational Operator . . . 6–8
6.2.17 Recovery-Unit Journal (.RUJ) Files Could Not Be Created Using Angle Brackets (< >) . . . 6–8
6.2.18 A Bugcheck Sometimes Resulted When a Sorted Index Rebalanced Itself . . . 6–9
6.2.19 NOWAIT Transactions Started During a Recovery Process Caused an RDMS-F-AREABUSY Fatal Error . . . 6–9
6.2.20 The Query Optimizer Caused Various Bugchecks When Queries Were Run . . . 6–10
6.2.21 Shared Write Queries Consumed More Memory Than Expected . . . 6–10
6.2.22 Locking Protocol Problem Caused Bugchecks . . . 6–10
6.2.23 Deleting and Then Creating a Logical Area and Accessing the Schema Caused a Page Checksum Bugcheck . . . 6–10
6.2.24 Rdb/VMS Behavior Had Changed so That Buffers Were Emptied on Rollback . . . 6–11
6.2.25 Lock-Related Looping Problem . . . 6–11
6.2.26 Problem with SPAM Thresholds in a Recover Operation . . . 6–11
6.2.27 Query Returned Records in Wrong Order with the SQL ORDER BY <column-name> DESCENDING or the RDO SORTED BY DESCENDING <field> Clauses . . . 6–11
6.2.28 The Predicate CONTAIN Uppercased the Second Byte of Some Two-Octet Characters Incorrectly . . . 6–12
6.2.29 Negate Operator Incorrectly Propagated the NULL Bit While Processing a Record Stream . . . 6–12
6.2.30 An UPDATE Operation Stored Incorrect Results . . . 6–13
6.2.31 An UPDATE Operation Caused a Bugcheck . . . 6–14
6.2.32 Query Using Descending Indexes Returned Incorrect Results . . . 6–14
6.2.33 Query with SQL LIKE Returned Incorrect Results . . . 6–15
6.2.34 Query with Compressed Indexes Returned Incorrect Results . . . 6–16
6.2.35 Query Returned Incorrect Results . . . 6–17
6.2.36 Poor Performance Was Observed with Queries Using Dynamic OR Optimization Within the Leaf Retrieval . . . 6–17
6.2.37 SPAM Pages Were Not Updated Correctly . . . 6–18
6.2.38 Global Section Was Corrupted When a User Had Multiple Attaches . . . 6–18
6.2.39 Under Certain Circumstances a Committed Update Was Not Completely Written to the .AIJ File . . . 6–18
6.3 IMPORT/EXPORT Problems Fixed in the Mandatory Update for Version 4.0 . . . 6–18
6.3.1 Importing a Database with Tables Containing Lists (Segmented Strings) Failed . . . 6–18
6.4 SQL Problems Fixed in the Mandatory Update for V4.0 . . . 6–19
6.4.1 SQL$STARTUP.COM Startup File Contained an Error in the SQL/Services Startup Logical Name . . . 6–19
6.4.2 Opening a Cursor That Was Already Opened Caused the Cursor to Lose Its State . . . 6–19

6.4.3 Executing the ROLLBACK Statement with OPEN LIST Cursors Left List Cursors in an Unusable State 6–20
6.4.4 Executing the COMMIT Statement with OPEN LIST Cursors Did Not Commit the Newly Created Lists 6–24
6.4.5 The OPEN Statement of an INSERT TABLE CURSOR Did Not Properly Return Error Status 6–26
6.4.6 Records Included from the Data Dictionary in the C Preprocessor Did Not Null Terminate Character Strings 6–29
6.4.7 Triggers Created with Long Source Text Strings Were Improperly Displayed 6–30
6.4.8 Triggers Created from Programs Had Their Source Text Truncated by a Word 6–30
6.5 SQL/Services Problems Fixed in the Mandatory Update for V4.0 6–31
6.5.1 Reinstalling SQL/Services APIs After Installation of Mandatory Update Kit for Rdb/VMS Version 4.0 6–31
6.5.2 SQL/Services Sample Application ULTRIX API Compiled with a Syntax Error 6–31
6.5.3 SQL/Services MS–DOS IVP Failed with a –2003 and 9 Error Status Codes, Indicating That Numbers Were Not Being Allowed Within Server Node Names 6–31
6.5.4 SQL/Services Length Packet Split Problem 6–32
6.5.5 SQL/Services ULTRIX API Was Not Freeing Network Connections . . . 6–32
6.5.6 SQL/Services Communication Server Did Not Report Error Status . . . 6–32
6.5.7 SQL/Services Shutdown Procedure Hung, Causing the Subsequent Startup Procedure Not to Work 6–32
6.5.8 SQL/Services Startup File Changes 6–33
6.5.9 SQL/Services Macintosh API Code Fixes 6–33
6.5.9.1 SQL/Services SQLSRV$Volume Installation Volume Could Not Be Accessed on the Macintosh 6–33
6.5.9.2 PATHWORKS DECtask Tool Name Changed 6–33
6.5.9.3 SQL/Services Macintosh API Fix for Macintoshes Based on the Motorola 68000 Chip 6–33
6.6 RDO, Callable RDO, RDBPRE, and RDML Problems Fixed in the Mandatory Update for V4.0 6–33
6.6.1 RDBPRE Generated Incorrect Code for Request Handles 6–34
6.6.2 A CDD/Plus Informational Message Caused RDML to Abort Compilation 6–34
6.6.3 An RDML-E-READ_ONLY Error Was Returned When Attempting to Update COMPUTED BY Fields 6–35
6.6.4 Problem with Callable RDO and Varying String Descriptors 6–35
6.7 RMU Problems Fixed in the Mandatory Update for V4.0 6–35
6.7.1 RMU/VERIFY Returned Spurious Errors Involving Fragmented Records 6–35
6.7.2 RMU/CONVERT Failed with a Default Collating Sequence Defined 6–36
6.7.3 The /INTERVAL Qualifier of the RMU/BACKUP/AFTER_JOURNAL Command Miscalculated a Specified Interval Value 6–36
6.7.4 Problem with RMU/SHOW USERS and RMU/SHOW SYSTEM Commands and VMS WORLD Privileges 6–36
6.7.5 RMU/REPAIR Command Caused Database Corruption–Problem I . . . 6–36
6.7.6 RMU/REPAIR Command Caused Database Corruption–Problem II . . . 6–37
6.7.7 RMU/DUMP and RMU/CLOSE Commands Required VMS SYSPRV Privilege 6–37

7 Known Problems, Restrictions, and Other Notes for the Mandatory Update for Rdb/VMS V4.0

7.1 General Problems, Restrictions, and Notes 7–1
7.1.1 VMS Lock Remastering Changed in VMS V5.4 7–1
7.2 Problems, Restrictions, and Notes for All Interfaces 7–1
7.2.1 Using Quoted Threshold Values for Binary Data Types for Partitioning Data or Indexes Results in Data or Index Corruption . . . 7–1
7.2.2 Problem with SQL LIKE and RDO MATCHING Clauses 7–2
7.2.3 RDB$REMOTE Account Has SYSTEM as Owner 7–3
7.2.4 An Arithmetic Exception Results When Joining Integer Columns 7–3
7.2.5 Collating Sequences That Use Two-to-Two Character Mapping Can Bugcheck 7–4
7.2.6 Query with Keys Scans the Index Instead of Using Direct Tree Lookup 7–4
7.2.7 Synchronization Problem for an Empty Sorted Index 7–5
7.2.8 Rdb/VMS Does Not Accept the Database File Specification in a Logical Name 7–6
7.2.9 Query Optimizer Does Not Choose Index-Only Retrieval When the Dbkey Is Selected 7–6
7.2.10 Rdb/VMS Hangs on a SELECT Statement When a Column Data Type Is Changed from INTEGER to CHARACTER to DATE 7–7
7.2.11 Rdb/VMS Monitor Fails When the Last User Finishes on a Particular Database 7–8
7.2.12 Multisegmented Index Is Not Selected When a Not-Equal Predicate Is Specified 7–8
7.2.13 Triggers That Affect Subject Table Rows Can Cause Loops or Inconsistent Results 7–9
7.2.14 Singleton Subselect Statement Returns Incorrect Results 7–9
7.2.15 Query with a FOR Loop with a MODIFY Statement Followed by a PRINT Statement Can Return Incorrect Results 7–10
7.2.16 Query with a Computed-By Field and OR Logic Returns Incorrect Results 7–10
7.2.17 Defining a View Causes a Bugcheck When a Sorted Index Was Previously Defined 7–11
7.2.18 Problem When Database Is Defined as Remote 7–11
7.2.19 An Incompatible Change for RDO Applications: New Update Rules Will Be Enforced by Default in V4.1 7–12
7.2.20 Relation Name Must Match Dictionary Record Name 7–13
7.2.21 NOWAIT Transactions Have Their Buffers Invalidated at COMMIT 7–13
7.3 SQL Problems, Restrictions, and Notes 7–13
7.3.1 SQL Deprecated Features and Incompatible Changes for VAX Rdb/VMS Version 4.1 7–14
7.3.2 SQL to Support Error Code Values in Rdb/VMS Version 4.1 7–15
7.3.3 Using the IGNORE CASE Option of the LIKE Clause Sometimes Results in a Query That Incorrectly Returns No Rows 7–15
7.3.4 An SQL SELECT Statement Results in an Invalid BLR Error 7–16
7.4 SQL/Services Problems, Restrictions, and Notes 7–16
7.4.1 SQL/Services VMS API Shipped with the Rdb/VMS Run-Time Kit . . . 7–16
7.4.2 VMS API Installation Without Rdb/VMS 7–16
7.4.3 Trailing Characters on SQL/Services Sample Program Error Messages 7–17
7.5 RDO, RDBPRE, and RDML Problems, Restrictions, and Notes 7–17
7.5.1 RDO IMPORT Does Not Save All SQL Defined Attributes 7–17

7.5.2 RDO CONVERT on V3.0 Databases Causes Database Corruption When the Database Is Converted to V4.0 7–17
7.6 Rdb/VMS Management Utility (RMU) Problems, Restrictions, and Notes 7–18
7.6.1 Do Not Delete After-Image Journal (.AIJ) Backup Files If the AIJ Backup Fails or Is Terminated 7–18
7.6.2 Concealed Logicals Are Supported but No Longer Recommended for Use After V4.0 7–18
7.6.3 Warnings from an RMU/VERIFY Operation 7–19
7.6.4 RMU/VERIFY/INDEX or RMU/VERIFY/ALL Command Causes a Bugcheck If You Have Hashed Indexes Defined 7–19
7.7 Notes and Restrictions Related to CDD/Plus 7–19
7.7.1 Restrictions Lifted by CDD/Plus Version 4.3 7–19
7.8 DECtrace Problems, Restrictions, and Notes 7–19
7.8.1 Rdb/VMS Version Number Used for DECtrace Will Remain at V4.0 7–19
7.9 Rdb/VMS Documentation Errors and Omissions in V4.0 7–20
7.9.1 Buffer Management Changes for V4.0 7–20
7.9.2 Incorrect Reference in V4.0 *VAX Rdb/VMS SQL Reference Manual*, Chapter 3 7–21
7.9.3 Printing Error in V4.0 *VAX Rdb/VMS SQL Reference Manual*, Chapter 4 7–21
7.9.4 Documentation Error in V4.0 *VAX Rdb/VMS SQL Reference Manual*, Appendix D.4 7–21
7.9.5 SQL/Services Error Documentation 7–22
7.10 SQL/Services Troubleshooting Suggestions 7–22
7.10.1 Common SQL/Services Network Errors 7–22
7.10.2 Common SQL/Services Fatal Execution Server Errors 7–23
7.10.3 Common SQL/Services API Installation Failures 7–24
7.10.4 SQL/Services Compatibility Issues 7–25
7.10.4.1 SQL/Services V4.0 Server Uses Proxy-Like and Default Access to Authorize V3.0 or V3.1 Client Applications 7–25
7.10.4.2 SQL/Services V4.0 Server Error –2031 Returned to V3.1 Client APIs 7–25
7.10.4.3 Queue Manager Must Be Started for the SQL/Services IVP to Work 7–25

8 Optional ECO Patches for the Mandatory Update for VAX Rdb/VMS V4.0

8.1 Optional ECO Patches That Can Be Applied to the Mandatory Update for Rdb/VMS V4.0 8–1
8.1.1 RDMSHRP ECO 30: Poor OR Optimization Performance on Read/Write Transactions 8–1
8.1.2 RDMMON ECO 01: Rdb/VMS Monitor Fails When the Last User Finishes on a Particular Database 8–2
8.1.3 RDMSHRP ECO 31: Multisegmented Index Is Not Selected When a Not-Equal Predicate Is Specified 8–2
8.1.4 RDMSHRP ECO 32: Singleton Subselect Statement Returns Incorrect Results 8–2
8.1.5 RDMSHRP ECO 33: Query with a FOR Loop with a MODIFY Statement Followed by a PRINT Statement Can Return Incorrect Results 8–3

A Sample V3.1C Installation

B Sample V4.0A Installation

Examples

6–1 Cursor Losing Its State 6–19
6–2 Executing ROLLBACK with LIST CURSORS in a Host C Program . . 6–20
6–3 Executing ROLLBACK with LIST CURSORS in SQL Module Language 6–23
6–4 Executing COMMIT with OPEN LIST Cursors 6–25
6–5 OPEN Statement Not Returning Error Information 6–27
6–6 OPEN Statement Not Returning Error Information in SQL Module Language 6–28
6–7 Records from the Data Dictionary Not Terminated with the NULL Character 6–29
6–8 Triggers Properly Displayed 6–30
6–9 Trigger Text Truncated 6–31

Tables

1–1 Disk Space Requirements 1–2
1–2 Process Account Quotas for the Installing Account 1–3
1–3 Required Minimum System Parameter Values 1–4
5–1 Disk Space Requirements 5–2
5–2 Process Account Quotas for the Installing Account 5–3
5–3 Required Minimum System Parameter Values 5–4
7–1 SQL/Services Network Errors 7–22

Preface

VAX Rdb/VMS software, Version 3.1 and Version 4.0, often referred to as Rdb/VMS in this manual, is a general-purpose database management system based on the relational data model.

This manual describes new and changed features, problems fixed in this release, current problems, additional restrictions, optional ECO patches, and other notes.

Intended Audience

This mandatory update includes an Installation Verification Procedure (IVP) and set of release notes intended for all users of Rdb/VMS Version 3.1B and 4.0, and should be read to supplement information contained in the Rdb/VMS Version 3.1 and 4.0 documentation sets.

To get the most out of this manual, you should be familiar with Rdb/VMS, data processing procedures, and basic database management concepts and terminology.

A Note on the Terminology

When the SQL and RDO interfaces use different terms to describe the same entity or concept, this manual uses the SQL term, unless the discussion is specifically about RDO or RDML. (This is also true of most other manuals in the Rdb/VMS documentation set.) For example, this manual normally uses table instead of relation, column instead of field (of a relation), and row instead of record.

The *VAX Rdb/VMS Introduction and Master Index* contains a more detailed list of SQL terms and their RDO equivalents.

Operating System Information

The version of VMS running on your system must be at least Version 5.2 for Rdb/VMS V3.1B and Version 5.3 for Rdb/VMS V4.0. Other information about the versions of the operating system and related software that are compatible with these versions of Rdb/VMS is included in the Rdb/VMS media kits and the *VAX Rdb/VMS Installation Guide* for each version.

For information on the compatibility of other software products with these versions of Rdb/VMS, refer to the System Support Addendum (SSA) that comes with the Software Product Description (SPD) for each version. You can use the SPD/SSA to verify which versions of your operating system are compatible with these versions of Rdb/VMS.

Structure

This manual is divided into two parts. Part I contains Chapter 1 through Chapter 4 and describes information pertaining to Rdb/VMS Version 3.1B. Part II contains Chapter 5 through Chapter 8 and Appendix A and Appendix B, and describes information pertaining to Rdb/VMS Version 4.0.

Part I	Mandatory Update for VAX Rdb/VMS Version 3.1B
Chapter 1	Describes preparations for installation and the Installation Verification Procedure for installing the mandatory update for Rdb/VMS Version 3.1B.
Chapter 2	Describes known software errors that are fixed in the mandatory update for VAX Rdb/VMS Version 3.1B.
Chapter 3	Describes current problems, additional restrictions, and workarounds known to exist for VAX Rdb/VMS Version 3.1B; may also include other information.
Chapter 4	Describes the optional ECO patches that can be applied to the mandatory update for VAX Rdb/VMS Version 3.1B.
Part II	Mandatory Update for VAX Rdb/VMS Version 4.0
Chapter 5	Describes preparations for installation and the Installation Verification Procedure for installing the mandatory update for Rdb/VMS Version 4.0.
Chapter 6	Describes known software errors that are fixed in the mandatory update for VAX Rdb/VMS Version 4.0.
Chapter 7	Describes current problems, additional restrictions, and workarounds known to exist for VAX Rdb/VMS Version 4.0; may also include other information.
Chapter 8	Describes the optional ECO patches that can be applied to the mandatory update for VAX Rdb/VMS Version 4.0.
Appendix A	Illustrates a sample installation of the mandatory update package on VAX Rdb/VMS Version 3.1B.
Appendix B	Illustrates a sample installation of the mandatory update package on VAX Rdb/VMS Version 4.0.

Related Manuals

For more information on VAX Rdb/VMS, see the following manuals in the Rdb/VMS documentation set. Note that all books cited are found in both the V3.1 and V4.0 documentation sets unless stated otherwise.

- *VAX Rdb/VMS Introduction and Master Index*

  Introduces Rdb/VMS and explains major terms and concepts. Includes a glossary, a directory of Rdb/VMS documentation, and a master index that combines entries from all the Rdb/VMS manuals.

- *VAX Rdb/VMS Guide to Distributed Transactions*

  A V4.0 book that describes the two-phase commit protocol and distributed transactions, explains how to start and complete distributed transactions using SQL, RDBPRE, and RDML, and how to recover from unresolved transactions using RMU commands.

- *VAX Rdb/VMS Guide to Database Design and Definition*

  Explains how to design a logical database and how to translate that design into a physical database using Rdb/VMS data definition statements.

- *VAX Rdb/VMS Guide to Database Maintenance and Performance*

  Provides guidelines for maintaining good database performance and explains how to use the database maintenance utilities to perform backup and recovery operations, restore journals, and analyze the database.

- *VAX Rdb/VMS Guide to Database Tuning*

  A V4.0 book that introduces the concept of tuning, and explores how tuning the system, the database, and the application affects database performance. Describes steps to follow in identifying, analyzing, isolating, and solving a performance problem, and in monitoring the resulting solution. Includes a set of decision trees that provide an organized approach to solving some common database tuning problems.

- *VAX Rdb/VMS Guide to Using RDO, RDBPRE, and RDML*

  Describes how to use the features of Rdb/VMS to retrieve, store, change, and erase data. Shows how to write programs that use Rdb/VMS as a data access method; contains information on writing programs in high-level languages that are supported by Rdb/VMS preprocessors, including Relational Data Manipulation Language (RDML); and describes Callable RDO, an interactive utility for languages without preprocessors.

- *VAX Rdb/VMS Guide to Using SQL*

  Introduces the Rdb/VMS SQL (structured query language) interface, and shows how to retrieve, store, and update data interactively and through application programs. Can be used as a tutorial for learning the major features of SQL.

- *VAX Rdb/VMS Guide to Using SQL/Services*

  Describes how to develop application programs that use SQL/Services, a client/server software component of Rdb/VMS that allows programs from various remote computers running the Macintosh, MS–DOS, OS/2, ULTRIX, ULTRIX for RISC, or VMS operating systems to access Rdb/VMS or VIDA databases on a VMS server system.

- *VAX Rdb/VMS SQL Reference Manual*

  Provides reference material and a complete description of the statements, the interactive, dynamic, and module language interfaces, and the syntax for SQL, the structured query language interface for Rdb/VMS.

- *VAX Rdb/VMS SQL Quick Reference Guide*

  Summarizes the information in the *VAX Rdb/VMS SQL Reference Manual*.

- *VAX Rdb/VMS RDO and RMU Reference Manual*

  Provides reference material and a complete description of the statements and syntax of the Rdb/VMS Relational Database Operator (RDO) interface and the commands of the Rdb/VMS Management Utility (RMU).

- *RDML Reference Manual*

  Describes the syntax and use of the Relational Data Manipulation Language (RDML), which can be embedded in VAX C or VAX Pascal programs to access Rdb/VMS or Rdb/ELN databases.

- *VAX Rdb/VMS Installation Guide*

  Describes how to install Rdb/VMS.

- *VAX Rdb/VMS Release Notes*

  Describes new features, problems and problems fixed, restrictions, and other information related to the current release of Rdb/VMS. Contains information about SQL and other Rdb/VMS interfaces and utilities.

Conventions

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

Often in examples the prompts are not shown. Generally, they are shown where it is important to depict an interactive sequence exactly; otherwise, they are omitted in order to focus full attention on the statements or commands themselves.

This section explains the conventions used in this manual:

Ctrl/x	This symbol in examples tells you to press the Ctrl (control) key and hold it down while pressing the specified letter key.
Return	This symbol in examples indicates the Return key.
Tab	This symbol in examples indicates the Tab key.
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
...	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.
$	The dollar sign represents the DIGITAL Command Language prompt. This symbol indicates that the DCL interpreter is ready for input.

References to Products

The Rdb/VMS documentation to which this document belongs often refers to products by their abbreviated names:

- DECdecision software is referred to as DECdecision.
- DEC RdbExpert for VMS software is referred to as RdbExpert.
- DECtrace for VMS software is referred to as DECtrace.
- VAX ACMS software is referred to as ACMS.
- VAX Ada software is referred to as Ada.
- VAX BASIC software is referred to as BASIC.
- VAX C software is referred to as C.
- VAX CDD/Plus software is referred to as CDD/Plus or the data dictionary.
- VAX COBOL software is referred to as COBOL.
- VAX Data Distributor software is referred to as Data Distributor.
- VAX DATATRIEVE software is referred to as DATATRIEVE.
- VAX DBMS software is referred to as VAX DBMS.

- VAX FORTRAN software is referred to as FORTRAN.
- VAXELN Pascal and VAX Pascal are both referred to as Pascal except when the use of a Relational Data Manipulation Language (RDML) statement is not the same in the VAXELN and VMS environments. In the latter case, either VAXELN Pascal or VAX Pascal is specified.
- VAX PL/I software is referred to as PL/I.
- VAX RALLY software is referred to as RALLY.
- VAX Rdb/ELN software is referred to as Rdb/ELN.
- VAX Rdb/VMS software is referred to as Rdb/VMS. VAX Rdb/VMS software Versions 3.1, 3.1A, and 3.1B, are often referred to as V3.1, V3.1A, and V3.1B respectively. VAX Rdb/VMS software Version 4.0 is often referred to as V4.0.
- VAX SQL software is referred to as VAX SQL whenever it is correct to refer to Version 2.0 or earlier of SQL. The use of SQL by itself indicates the SQL interface now included as part of VAX Rdb/VMS Version 3.1 and Version 4.0.
- VAX TEAMDATA software is referred to as TEAMDATA.
- VAX TDMS software is referred to as TDMS.
- VIDA software is referred to as VIDA.

Part I

Mandatory Update for VAX Rdb/VMS Version 3.1B

This part contains Chapter 1 through Chapter 4 and describes information pertaining to the mandatory update of VAX Rdb/VMS Version 3.1B.

1

Installing the Rdb/VMS Version 3.1B Mandatory Update

This chapter describes information necessary for installing the mandatory update for Rdb/VMS Version 3.1B.

1.1 Before Installing the Rdb/VMS Mandatory Update Package for V3.1B

Before you install the mandatory update package (MUP), you must install Rdb/VMS V3.1B; see the *VAX Rdb/VMS Installation Guide* for instructions on how to install Rdb/VMS. The following sections describe what steps you need to take to install the MUP.

The installation of the MUP checks to see whether you have Rdb/VMS V3.1B or Rdb/VMS V4.0 installed on your system. It then installs either V3.1C or V4.0A. The name of the mandatory update kit is RDBVMS_MUPA040. This name appears in installation messages whether you are installing the MUP for V3.1B or V4.0.

1.1.1 Prerequisite Hardware and Software

This section discusses the hardware and software you must have installed on your system before you install the mandatory update package of Rdb/VMS.

You can install the MUP only when your system meets or exceeds the minimum hardware requirements as shown in the SPD. Table 1–1 lists the approximate system disk storage required for the installation of the MUP. Your system may require additional mass storage for backup and restore operations.

The VMS operating system Version 5.3 or higher must be installed on your VAX system if you are installing V3.1C.

Rdb/VMS V3.1B or Rdb/VMS 4.0 must be installed before you install the MUP. If one of these versions is not present, the installation aborts. The installation uses the RDO.EXE image to check the existing Rdb/VMS version number. If you have deleted your RDO.EXE image, you must restore it from the saveset using the VMS BACKUP command. For example, to restore RDO.EXE from the full development kit, enter the following command:

```
$ BACKUP <device>:RDBVMSDEV031.F/SAVE/SELECT=RDO.EXE -
_$ SYS$SYSROOT:[SYSEXE]RDO.EXE
```

To restore RDO.EXE from the interactive kit, enter the following command:

```
$ BACKUP <device>:RDBVMSINT031.E /SAVE/SELECT=INTRDO.EXE *.*
```

To restore RDO.EXE from the run-time kit, enter the following command:

```
$ BACKUP <device>:RDBVMSRTO031.B/SAVE/SELECT=RTORDO.EXE *.*
```

1.1.2 Back Up All Existing Rdb/VMS Databases

As a precaution, Digital recommends that you back up any Rdb/VMS databases, including DECtrace and CDD/Plus databases, with the RMU/BACKUP command before installing the MUP.

Before installing a new version of Rdb/VMS, Digital recommends that you perform a full RMU/BACKUP of the DECtrace administration and history databases, including DECtrace databases produced with the DECtrace FORMAT command. To back up the DECtrace administration database, use the following command:

```
$ RMU/BACKUP ERC$ADMIN_DB EPC$ADMIN_DB.RBF
```

To backup the DECtrace history database, use the following command:

```
$ RMU/BACKUP EPC$HISTORY_DB EPC$HISTORY_DB.RBF
```

1.1.3 Disk Space Required to Install the MUP

Installing the MUP requires a certain amount of available disk storage space during the installation. Once the MUP is installed, it takes the same amount of space as your previous version of Rdb/VMS. Table 1–1 summarizes the storage requirements for Rdb/VMS.

Table 1–1 Disk Space Requirements

Rdb/VMS Kit	Blocks During Installation
Full development	13,000
Interactive	13,000
Run-time	13,000

To determine the number of available disk blocks on the current system disk, enter the following command at the DCL prompt:

```
$ SHOW DEVICE SYS$SYSDEVICE
```

1.1.4 Shut Down the Rdb/VMS Monitor

The installation procedure terminates if the Rdb/VMS monitor is running. Before starting the installation, ensure that there are no active Rdb/VMS users by shutting down the Rdb/VMS monitor.

Note

If DECtrace is installed on your system, you must turn DECtrace off *before* you attempt to shut down the Rdb/VMS monitor. Turn DECtrace off using the following command:

```
$ COLLECT STOP SYSTEM/ABORT
```

Alternatively, you could stop both DECtrace and the Rdb/VMS monitor using the RMU/MONITOR STOP/ABORT=DELPRC command.

Run the RMONSTOP.COM procedure from SYS$STARTUP to shut down the monitor on all nodes in a VAXcluster system. For example:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
SYSMAN> DO @SYS$STARTUP:RMONSTOP
SYSMAN> EXIT
```

If you want to stop the Rdb/VMS monitor on only one node, enter the following command on that node:

```
$ @SYS$STARTUP:RMONSTOP
```

1.1.5 Obtain the VMS Privileges Required to Install the MUP

VMSINSTAL is located in SYS$UPDATE, which is a restricted directory. To install the MUP, you must use an account that has SETPRV privilege. As one of its first actions, the VMSINSTAL command procedure grants all privileges except BYPASS to the process that invokes it. The VMSINSTAL command succeeds only if the account has SETPRV privilege.

To check the default privileges of the installing account, log in and enter this DCL command:

```
$ SHOW PROCESS/PRIVILEGES
```

If the account lacks the SETPRV privilege, you cannot install the MUP. You have two options:

- Ask your system manager to use AUTHORIZE to modify the default privileges of the account to include the SETPRV privilege.
- Run AUTHORIZE and make the changes yourself, if your account has the SYSPRV privilege:

  ```
  $ SET DEFAULT SYS$SYSTEM
  $ RUN AUTHORIZE
  UAF> MODIFY account-name/PRIVILEGES=(SETPRV)
  UAF> EXIT
  ```

  To activate the change in privileges, you must log out and log in again.

(Note that the VMSINSTAL procedure turns off the BYPASS privilege at the start of the installation.)

1.1.6 Ensure Sufficient Process Account Quotas to Install the MUP

The account you use to install the MUP must have sufficient quotas to enable you to perform the installation. Table 1–2 summarizes the minimum process quotas required to install the MUP.

Table 1–2 Process Account Quotas for the Installing Account

Account Quota	Value
ASTLM	24
BIOLM	18
BYTLM	20,480
DIOLM	18

(continued on next page)

Table 1–2 (Cont.) Process Account Quotas for the Installing Account

Account Quota	Value
ENQLM	2000
FILLM	50
PGFLQUO	20,000

User account quotas are stored in the file SYSUAF.DAT. You use AUTHORIZE to verify and change user account quotas. First set your directory to SYS$SYSTEM and then run AUTHORIZE:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF>
```

At the AUTHORIZE prompt (UAF>), use the SHOW command with an account name to check a particular account. For example, to check the SYSTEM account enter:

```
UAF> SHOW SYSTEM
```

To change a quota, use the MODIFY command at the UAF> prompt. The MODIFY command has the following syntax:

MODIFY account-name /quota-name=NNN

The following example changes the FILLM quota for the SYSTEM account and then exits from AUTHORIZE:

```
UAF> MODIFY SYSTEM /FILLM=50
UAF> EXIT
```

After you exit from the utility, the VMS system displays messages that indicate whether or not changes were made. Once the changes have been made, you must log out and log in again for the new quotas to take effect.

For more information on modifying account quotas, see the description of AUTHORIZE in the VMS documentation set.

1.1.7 Obtain System Parameter Values Required to Install the MUP

Installing the MUP requires certain system parameter settings. Table 1–3 lists the minimum required system parameter values for the installation. Depending on the kinds of programs and applications running at your site, you might need higher values for some settings.

Table 1–3 Required Minimum System Parameter Values

System Parameter	Value
CHANNELCNT	A number larger than the largest FILLM used on the system
CLISYMTBL[1]	250 pages

[1]The CLISYMTBL dynamic system parameter must be set to a minimum value of 250 pages during the installation procedure. If the current CLISYMTBL setting is less than 250 pages, you can lower the setting to its original value once the installation is finished.

(continued on next page)

Table 1–3 (Cont.) Required Minimum System Parameter Values

System Parameter	Value
GBLPAGES[2]	2078 available pages
GBLSECTIONS[2]	80 available sections
LOCKIDTBL	256 entries
LOCKIDTBL_MAX[3]	2048 entries
MAXBUF	2048 bytes
PROCSECTCNT	32 sections
RESHASHTBL	512 entries
SRPCOUNT	1024 packets
SRPCOUNTV	2048 packets
VIRTUALPAGECNT	20,000 (a number larger than largest PGFLQUOTA used on the system)

[2]For systems where you are performing a reinstallation, this number is the current value of GBLSECTIONS or GBLPAGES when the RMONSTOP command file or the RMU/MONITOR STOP command has been executed.

[3]This dynamic system parameter must be set permanently to a value equal to or greater than the value listed. Do not lower this value after the installation.

Section 1.1.7.1 through Section 1.1.7.3 show you how to check system parameter values, calculate values for the GBLPAGES and GBLSECTIONS system parameters, and change parameter values with the VMS AUTOGEN command procedure. Section 1.1.7.4 shows you how to use SYSGEN to change the values for dynamic system parameters.

1.1.7.1 Checking System Parameter Values

To check the values of your system parameters, enter the following command at the DCL prompt to invoke the VMS System Generation utility (SYSGEN):

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>
```

At the SYSGEN prompt (SYSGEN>), enter the SHOW command to display the value of a system parameter. The values displayed should equal or exceed the value of each parameter listed in Table 1–3. The following command displays the value for the LOCKIDTBL_MAX system parameter:

```
SYSGEN> SHOW LOCKIDTBL_MAX
Parameter Name    Current   Default   Minimum   Maximum Unit  Dynamic
--------------    -------   -------   -------   ------- ----  -------
LOCKIDTBL_MAX     65535     65535       200     65535 Entries     D
```

After you finish checking the parameters with the SHOW command, you can enter the EXIT command at the SYSGEN prompt to return to DCL.

1.1.7.2 Calculating the Values for GBLPAGES and GBLSECTIONS

To install and run the MUP, you must set the correct values for the GBLPAGES and GBLSECTIONS system parameters.

The 2078 value for GBLPAGES and the 80 value for GBLSECTIONS in Table 1–3 indicate that you must have at least 2078 unused pages and 80 unused sections available on your system for the installation to proceed successfully.

To see how many unused global pages and global sections your system has, enter the following DCL commands:

```
$ WRITE SYS$OUTPUT F$GETSYI ("FREE_GBLPAGES")
8900
$ WRITE SYS$OUTPUT F$GETSYI ("FREE_GBLSECTS")
90
```

Section 1.1.7.3 describes the procedures for increasing these values as well as those of other system parameters. Refer to the VMS documentation on system management and operations for more information.

1.1.7.3 Changing System Parameter Values with AUTOGEN

You use the AUTOGEN command procedure to change system parameters. The AUTOGEN procedure automatically adjusts values for parameters that are associated with the ones you set manually. To change system parameters with AUTOGEN, you must edit the SYS$SYSTEM:MODPARAMS.DAT file.

Use an editor to access the file. If you need to change a parameter value that is already in the SYS$SYSTEM:MODPARAMS.DAT file, delete the current value associated with that parameter and enter the new value.

To add a new value, add a line to the MODPARAMS.DAT file. The line contains the name of the parameter and its value. For example:

```
LOCKIDTBL_MAX = 2048
```

You can also modify incremental parameters in the MODPARAMS.DAT file. The following example increases the global page setting by 2000:

```
ADD_GBLPAGES = 2000
```

After you have made all your changes, run the AUTOGEN procedure to recalculate your system parameters. Enter the following command at the DCL prompt:

```
$ @SYS$UPDATE:AUTOGEN GETDATA REBOOT
```

AUTOGEN automatically adjusts some of the SYSGEN parameters based on the consumption of resources since the last reboot. If you do not want to take advantage of this automatic adjustment, include the NOFEEDBACK parameter at the end of the AUTOGEN command line. The AUTOGEN procedure performs an automatic system shutdown and reboots when it has finished. Rebooting your system makes the new parameter values active.

For more information about using AUTOGEN, see the instructions on modifying system parameters in the VMS documentation on system management and operations.

1.1.7.4 Setting Dynamic System Parameters

You can use SYSGEN to change the values for dynamic system parameters. The following example demonstrates this process for the CLISYMTBL system parameter. (After the installation is complete, you can reset CLISYMTBL to its previous setting or let it be reset automatically when you reboot your system.)

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> USE ACTIVE
SYSGEN> SET CLISYMTBL 250
SYSGEN> WRITE ACTIVE
SYSGEN> EXIT
```

Dynamic parameters changed with the SYSGEN WRITE ACTIVE command become active immediately without any need to reboot your system. In fact, rebooting returns dynamic system parameter values to their previous settings. Once you set values for dynamic parameters, you should complete the installation before rebooting the system.

The values for other dynamic parameters, such as LOCKIDTBL_MAX, must remain at the same level or higher than the values specified in Table 1–3.

1.1.8 Back Up Your System Disk

At the beginning of the installation, the VMSINSTAL command procedure asks if you have backed up your system disk. Digital recommends that you back up your system disk before installing any software on top of the operating system.

This precaution protects your system software. A system failure at a critical point in the installation procedure could leave unusable files. You also protect an existing version of the product, which may, if you request it, be deleted during the installation.

Use the backup procedures that have been established at your site. For details on backing up the system disk, see the section on the VMS Backup utility in the VMS documentation set.

1.1.9 Avoid Giving Users Access to HELP

When the installation inserts the Rdb/VMS Help modules into the VMS Help library, it must have sole access to the VMS Help library. If anyone uses the HELP command when the installation tries to insert the Rdb/VMS Help module, the installation fails. You can prevent other users from using the help library during the installation by either of the following methods:

- Running the installation when no one else is logged in
- Limiting access to the help library SYS$HELP:HELPLIB.HLB to the SYSTEM account:

```
$ SET PROTECTION = (S:RWED, O, G, W) SYS$HELP:HELPLIB.HLB
```

Remember to note the original protection on the library. After the installation, return the protection on the help library to the original setting. Instructions are provided in Section 1.3.3.

1.1.10 Prevent Interactive Users from Gaining Access to the System

If the installation fails for an indeterminable reason, Digital recommends that you install the MUP again, keeping all interactive users off the system during the installation procedure. You might also choose to keep interactive users off the system if you will be changing any system parameter values with the AUTOGEN command procedure. Use the DCL REPLY command to inform users of the schedule for the installation. Prevent other users from logging in by issuing the DCL SET LOGIN command:

```
$ REPLY/USER "Installation of Rdb/VMS starting in 5 minutes. Please
log out."
$ SET LOGIN/INTERACTIVE=0
```

Both of these commands require the OPER privilege.

If any batch or device jobs are running, you have two options:

- Wait until the last one finishes.
- Use the DCL DELETE/ENTRY command to stop any job still running.

1.2 Installing the Mandatory Update Package

This section describes how to install the mandatory update package (MUP).

1.2.1 Time Required to Install the MUP

The installation of V3.1C takes approximately 20 minutes on a VAX 8800 system. This time may vary depending on your type of media, your system configuration, whether or not CDD/Plus is installed, and whether or not you need to reboot your system.

1.2.2 Invoking VMSINSTAL

To start the installation, invoke the VMSINSTAL command procedure from a privileged account, such as the SYSTEM account. The VMSINSTAL procedure is in the SYS$UPDATE directory. You use the following syntax to invoke VMSINSTAL:

```
@SYS$UPDATE:VMSINSTAL product-name device-name OPTIONS N
```

The rest of this section presents the parameters for the VMSINSTAL command line.

product-name
The installation name for the product. For the MUP, provide this parameter as follows:

RDBVMS_MUPA040

device-name
The name of the device on which you plan to mount the media. For example, MTA0: and MUA0: are device names for tape drives. It is not necessary to use the console drive for this installation. However, if you do use the console drive, you should replace any media you removed once the installation is complete.

OPTIONS N
An optional parameter that indicates you want to review the release notes question. If you include the OPTIONS N parameter, VMSINSTAL displays a menu that lets you choose between printing the release notes or displaying them on your terminal. You should always review the release notes before proceeding in case they contain new information about the installation. If you do not include the OPTIONS N parameter, VMSINSTAL does not ask you about the release notes. However, the release notes are automatically copied to SYS$HELP.

Note that there are several other options you can select when you invoke VMSINSTAL. See the VMS documentation on software installation for information on these options.

The following example displays the command to invoke VMSINSTAL to install Rdb/VMS from tape drive MTA0: and shows the system response. This example uses the OPTIONS N release note parameter.

```
$ @SYS$UPDATE:VMSINSTAL RDBVMS_MUPA040 MTA0: OPTIONS N

      VAX/VMS Software Product Installation Procedure V5.3


It is 14-MAR-1991 at 14::00.
Enter a question mark (?) at any time for help.
```

If you do not supply either the product name or the device name, VMSINSTAL prompts you for this information later in the installation procedure.

1.2.3 Steps of the Installation Procedure

This section discusses the installation process itself, presenting all the questions that appear during the installation.

Each question in the installation is marked with an asterisk (*) at the beginning of the line. Some questions show the default response in brackets, for example, [YES]. If you want to use the default response, press the Return key.

1. System backup

   The VMSINSTAL procedure asks if you are satisfied with your system backup. You should always back up your system disk before performing an installation. If you are satisfied with the backup of your system disk, press the Return key. Otherwise, enter NO to discontinue the installation. After you back up your system disk, you can start the installation again.

   ```
   * Are you satisfied with the backup of your system disk [YES]?
   ```

2. Mounting the media

   You should now mount the first distribution volume on the device you specified when you invoked VMSINSTAL. The VMSINSTAL procedure then asks you if you are ready to continue with the installation.

   If you respond YES to indicate that you are ready, VMSINSTAL displays a message that the media containing Rdb/VMS has been mounted on the specified device and that the installation has begun. For example:

   ```
   * Where will the distribution volumes be mounted: MTA0:

   Enter the products to be processed from the first distribution volume set.
   * Products: RDBVMS_MUPA040
   * Options: N
   The following products will be processed:
    RDBVMS_MUPA V4.0

       Beginning installation of RDBVMS_MUPA V4.0 at 14:00

   %VMSINSTAL-I-RESTORE, Restoring product saveset A...
   ```

   If you entered the wrong device name when you invoked VMSINSTAL and need to start the installation again, enter NO when asked if you are ready to install.

3. Release notes

   If you specified OPTIONS N when you invoked VMSINSTAL, you are now asked to choose one of the four options for reviewing the release notes.

   ```
   Additional Release Notes Options:

           1.  Display release notes
           2.  Print release notes
           3.  Both 1 and 2
           4.  None of the above
   ```

```
* Select option [2]: 2
```

The release notes are automatically copied to SYS$HELP no matter which option you choose, and whether or not you specified OPTIONS N.

The release notes are long; you might wish to print them by selecting option 2.

4. Continuing the installation

   The installation procedure now asks if you want to continue the installation. To continue, enter YES. Otherwise, press the Return key. In either case, the release notes are copied to a file in the SYS$HELP directory. For example:

```
* Do you want to continue the installation [N]?: YES
%VMSINSTAL-I-RELMOVED, The product's release notes have been
successfully moved to SYS$HELP.
```

   The release notes, entitled *VAX Rdb/VMS Mandatory Update for Versions 3.1B and 4.0*, are located in the following file:

```
SYS$HELP:RDBVMS_MUPA040.RELEASE_NOTES
```

Note

The name of the release notes file installed by VMSINSTAL consists of the current product name and version number. Digital recommends that you keep the release notes for previous versions of Rdb/VMS.

When you continue the installation, the following message is displayed:

```
    Installation procedures for: "VAX Rdb/VMS V3.1C-0"


Be sure you have read the section on pre-installation steps in the
installation guide before continuing with the installation.


Checking system requirements ...
```

5. Choosing to run the Installation Verification Procedure (IVP)

   The Installation Verification Procedure (IVP) for the MUP verifies the installation. The installation asks if you want to run the IVP as part of the installation. If you respond YES, VMSINSTAL runs the IVP following the installation. It is recommended that you run the IVP to be sure that the MUP is installed correctly.

```
* Do you want to run the IVP after the installation [YES]?
```

   As part of the IVP, the MUP creates the personnel sample database in the directory specified by the logical RDM$DEMO.

   After the MUP is installed, you can run the IVP independently to verify that the software is available on your system. You might also want to run the IVP after a system failure to be sure that users can access Rdb/VMS. Online help contains instructions for running the IVP independently. More information follows later in this section.

6. Choosing to purge files

   You have the option to purge files from previous versions of Rdb/VMS that are superseded by this installation. Purging is recommended; however, if you need to keep files from the previous version, enter NO in response to the question.

```
* Do you want to purge files replaced by this installation [YES]?
```

7. Informational messages

   At this point, the installation procedure displays a number of informational messages that report on the progress of the installation. There are no further questions. If the installation procedure has been successful up to this point, VMSINSTAL moves the new or modified files to their target directories, updates help files, and updates DCL tables, if necessary. If you asked for files to be purged, that work is done now. The following messages are displayed:

```
    There are no more questions.  The installation takes approximately
    20 minutes on a standalone VAX 8800.


    Beginning installation ...


    Installing under VMS V5.3     - 14-MAR-1991 14:10


%VMSINSTAL-I-RESTORE, Restoring product saveset B ...
%VMSINSTAL-I-RESTORE, Restoring product saveset F ...

   .
   .
   .

%VMSINSTAL-I-MOVEFILES, files will now be moved to their target
directories . . .
```

8. Running the IVP

   If you chose to run the IVP, VMSINSTAL runs it now. When the IVP runs successfully, you see the following display:

```
************************************

VAX Rdb/VMS V3.1C

Development

IVP COMPLETED SUCCESSFULLY

************************************

IVP completed for: VAX Rdb/VMS V3.1C-0
```

1.2.4 Completing the Installation Procedure

The following messages indicate that the entire installation procedure is complete:

```
Installation of RDBVMS_MUPA V4.0 completed at 14:45

VMSINSTAL procedure done at 14:45
```

You can now log out of the privileged account:

```
$ LOGOUT
SYSTEM     logged out at 14-MAR-1991 14:45:00.0
```

Note that VMSINSTAL deletes or changes entries in the process symbol tables during the installation. Therefore, if you are going to continue using the system manager's account and you want to restore these symbols, you should log out and log in again.

1.2.5 Errors That Cause the Installation to Fail

If errors occur during the installation itself or when the IVP is running, VMSINSTAL displays failure messages. If the installation fails, you see the following message:

```
%VMSINSTAL-E-INSFAIL, The installation of RDBVMS_MUPA V3.1C has failed.
```

If the IVP fails, you see these messages:

```
The RDBVMS_MUPA V3.1C Installation Verification Procedure failed.

%VMSINSTAL-E-IVPFAIL, The IVP for RDBVMS_MUPA V3.1C has failed.
```

Errors can occur during the installation if any one of the following conditions exists:

- Incorrect Rdb/VMS version

  Unless you have V3.1B, V3.1C, V4.0 or V4.0A of Rdb/VMS installed, the installation will fail.

- Incorrect operating system version

  Unless you have the VMS Version 5.2 or higher operating system installed, the installation will fail.

- If you have deleted the RDO or RDMPRV images, you must restore them from the saveset using the VMS RESTORE command.

- Insufficient privileges

  The account you use to install the MUP must have the SETPRV privilege.

- Insufficient disk space on system disk

  If the system disk does not have enough blocks available to install the MUP, purge or delete unnecessary files according to the policies of your site. When you have enough disk space, you are ready to continue the installation procedure.

  See Table 1–1 for disk space requirements.

- Insufficient system parameter values for successful installation

  You must have the necessary minimum settings for system parameters on the installing account. See Table 1–3 for more system parameter information.

- Insufficient quotas for successful installation

  You must have the necessary minimum account quotas set. See Table 1–2 about process account quotas.

- VMS Help Library currently in use

  The installation must have sole access to the VMS Help Library when it tries to insert the Rdb/VMS Help module into the library.

- CDD/Plus installed but not started up prior to MUP installation

  If CDD/Plus is installed on your system but not started up, the IVP will commonly fail in the COBOL precompiler tests. If this occurs, start up CDD/Plus and rerun the IVP. Use the following command to start up CDD/Plus:

  ```
  $ @SYS$STARTUP:CDDSTRTUP
  ```

1.3 After Installing the Mandatory Update Package

After installing the mandatory update package (MUP), you need to perform the following tasks:

- Tailor your system.
- Return the system to original settings.

This section also explains how to access the online release notes and how to run the Installation Verification Procedure (IVP) independently after the software has been installed.

1.3.1 Accessing the Online Release Notes

Once the MUP has been installed, the release notes (this book) are located in SYS$HELP:RDBVMS_MUPA040.RELEASE_NOTES.

Online help also directs you to the release notes file. After the installation, you can enter the following command to find the location of the release notes:

```
$ HELP RDBVMS RELEASE_NOTES
```

1.3.2 Tailoring Your System

This section discusses steps you must take to tailor your system to run Rdb/VMS after installing the MUP.

1. If you had the RDBPRE problem with request handles, recompile your programs. This problem has been fixed in this release.
2. Run RMONSTART.COM manually or by means of the Installation Verification Procedure (IVP).

   If you chose not to run the IVP as part of the installation, you will have to run the RMONSTART.COM command procedure manually to start the Rdb/VMS monitor and perform other related tasks such as installing shareable images and defining necessary logical names. The RMONSTART.COM command procedure is located in SYS$STARTUP. (If you have edited RMONSTART.COM to define LNK$LIBRARY, you will have to run the IVP on the boot node as well as on the VAXcluster satellite nodes.)

   Simply running the system startup command procedure SYSTARTUP_V5.COM or the RMU/MONITOR START command does not perform all of the tasks that the RMONSTART.COM procedure does and that Rdb/VMS requires.
3. Obtain the list of files installed by Rdb/VMS.

   A file is written to your system that identifies all the Rdb/VMS files installed on your system. To obtain this list after the installation ends, print (DCL PRINT) or display (DCL TYPE) a copy of the following file:

   ```
   SYS$COMMON:[SYSMGR.VAXINFO$PRODUCTS]RDBVMS_MUPA040_FILES.DAT
   ```

4. Run the MUP IVP as a standalone procedure.

   The MUP IVP procedure can be run at any time after the successful installation of the MUP. For example, if Rdb/VMS does not appear to be running properly, you may want to verify that the correct MUP distribution kit files are present on your system.

   The account you use to run the IVP must have the TMPMBX and SYSPRV privileges. If the data dictionary is installed on the system, the account must also have BYPASS privilege or the CDD EXTEND privilege at the CDD$TOP dictionary directory. Also, the quotas for the account you use must be sufficient to run Rdb/VMS.

   To run the MUP IVP after installation:

   a. Set the default to the following directory:

      ```
      $ SET DEFAULT SYS$COMMON:[SYSTEST]
      ```

   b. The command you enter to invoke the IVP depends on whether or not you have installed the VAX Rdb/VMS full development, interactive, or run-time kit license option:

      ```
      $ @RDBIVP DEV   ! Executes full development kit IVP
      $ @RDBIVP INT   ! Executes interactive kit IVP
      $ @RDBIVP RTO   ! Executes run-time kit IVP
      ```

      The standalone IVP procedure runs in the same manner as the VMSINSTAL IVP procedure. If the IVP fails, it creates a log file, SYS$UPDATE:RDBIVP.LOG, of the failed portion of the test.

1.3.3 Returning the System to Original Settings

If you have set interactive logins to 0 or changed the protection on the help library, you must reverse these actions.

- To restore interactive logins, enter the following command:

  ```
  $ SET LOGIN/INTERACTIVE=value
  ```

- To change the protection on the help library, enter the following commands:

  ```
  $ SET DEFAULT SYS$HELP
  $ SET PROTECTION=(S:RWED,O:RWED,G:RWED,W:RE) HELPLIB.HLB
  ```

- If the system parameter CLISYMTBL was less than 250 before the installation, you can now set it to the original setting.

2

Software Errors Fixed in the Mandatory Update for Rdb/VMS V3.1B

The following sections describe problems with previous versions of the Rdb/VMS software that are fixed in the mandatory update for Version 3.1B. These software problems no longer exist.

2.1 General Information

This section contains notes and problem descriptions of a general nature.

2.1.1 A VMS Sort Utility Problem Affected Rdb/VMS

In some instances bad data was returned from queries when the query involved a sort on a large number of records. This problem has been identified as a problem in the VMS Sort utility (SORT). This problem has affected Rdb/VMS V3.1 because a variant of the Sort utility is linked to Rdb/VMS. In addition to affecting the output of queries, this problem could also affect definitions of indexes and definitions of constraints.

This timing-dependent problem occurred when a buffer was allocated for both scratch file reading and writing, and generally only became noticeable when the number of records to be sorted was fairly large, such that sorted record retrieval from the scratch file overlapped scratch file write operations through the same buffer.

For example, the following query, embedded in a host program, produced bad data in the output that was generated:

```
FOR   CP IN R_RELATION
   WITH  CP.A = "9005" AND
         CP.B <= "199005" AND
         CP.C = "UT"
   SORTED BY CP.D
             CP.E
             CP.F
             CP.G
```

Several of the output records produced null values, as shown in this example.

```
   .
   .
   .
29365200002407592 1      367.00        0.00AAA6BBB CCC 900531050891D411
T ^@1^@^@^@^@^@^@^@^@^@^@^@^@^@^@        0.00        0.00^@1^@^@    ^@059^@02
T ^@1^@^@^@^@^@^@^@^@^@^@^@^@^@^@        0.00        0.00^@1^@^@    ^@006^@02
T ^@1^@^@^@^@^@^@^@^@^@^@^@^@^@^@        0.00        0.00^@1^@^@    ^@008^@02
T ^@1^@^@^@^@^@^@^@^@^@^@^@^@^@^@        0.00        0.00^@1^@^@    ^@010^@02
T ^@1^@^@^@^@^@^@^@^@^@^@^@^@^@^@        0.00        0.00^@1^@^@    ^@012^@02
T ^@1^@^@^@^@^@^@^@^@^@^@^@^@^@^@        0.00        0.00^@1^@^@    ^@018^@02
T ^@1^@^@^@^@^@^@^@^@^@^@^@^@^@^@        0.00        0.00^@1^@^@    ^@025^@02
```

Note

When the WITH clause was eliminated no bad data resulted. When the SORTED BY clause was amended slightly to reduce the number of sort-by fields again no bad data resulted.

This problem is fixed in the mandatory update for V3.1B. This is RDMSHRP ECO 3. The Sort utility that is used by the query optimizer within Rdb/VMS is fixed.

2.1.2 The VMS Sort Utility for VMS V5.1, V5.2, and V5.3 Caused Problems with Rdb/VMS Databases

When an SQL or RDO IMPORT statement was used to import a database that had a table with a placement via hashed index defined, database corruption resulted. If the table had an index with no duplicates allowed defined for it, an RDB-E-NO_DUP error message was returned. This problem was caused by the VMS Sort utility (SORT) and was similar to the problem described in Section 2.1.1. However, in this case Rdb/VMS used the Sort utility that was not part of Rdb/VMS to perform this IMPORT operation.

Rdb/VMS called the Sort utility to sort a file to process the placement via hashed index, but the Sort utility returned empty (null) records. Rdb/VMS then attempted to store these records in the database. If a no duplicates index was defined for the table, then the RDB-E-NO_DUP error message was returned. Otherwise, corrupt records were stored in the database with no error returned to the user.

This Sort utility problem could also occur if you used the RMU/UNLOAD command, the DCL SORT command, and the RMU/LOAD/PLACE command to sort and reload your database tables.

This problem is fixed in VMS V5.4. In addition, a VMSINSTAL kit is available from your Customer Support Center to fix the problem for V5.2 and V5.3.

2.2 Software Errors Fixed That Apply to All Interfaces in the Mandatory Update for Version 3.1B

This section contains notes and problems fixed in all interfaces.

2.2.1 Rdb/VMS Returned Errors When Retrying Failed Multidatabase Transactions

Rdb/VMS V3.1, V3.1A, and V3.1B did not completely clear internal transaction information after a multidatabase SQL SET TRANSACTION or RDO START TRANSACTION failure (usually due to a lock conflict or deadlock). This was a problem in both RDO and SQL.

If, after the failure, you attempted to retry the SQL SET TRANSACTION or RDO START TRANSACTION operation, you received an error as follows:

```
%RDB-E-EXCESS_TRANS, exceeded limit of 1 transaction per database attachment
```

If you attempted a ROLLBACK operation, you received an error as follows:

```
%SQL-F-NO_TXNOUT, No transaction outstanding
```

There is no workaround for this problem.

This problem is fixed in the mandatory update for V3.1B by RDBSHR ECO 1.

2.2.2 Active Transactions in Application Programs Could Not Recover from Network Failures

If a network failure occurred during a transaction, an RDB$_IO_ERROR error was returned and the program running was not able to do any further Rdb/VMS processing.

This problem is fixed in the mandatory update for V3.1B. The program can now finish and detach from the database. If a network failure occurs or there are other errors, such as constraint errors, an error message is returned indicating why the call failed. This enables applications to continue without exiting by trapping the RDB$_IO_ERROR and taking appropriate actions, such as finishing all attaches and reattaching later.

2.2.3 Data Transfer from the V3.1B Server Caused Problems

There was a problem in data transfer from a V3.1B server to a V3.1B client and from a V3.1B server to V4.0 client. There was no problem in data transfer from a V4.0 server to V3.1B client and from a V4.0 server to a V4.0 client.

This problem resulted in the following error message when you performed a query such as:

```
SQL> SELECT * FROM EMPLOYEES;
%RDB-F-IO_ERROR, input or output error
-SYSTEM-F-LINKABORT, network partner aborted logical link
```

This problem was caused by mismatching buffer sizes in the V3.1B server.

This problem is fixed in the mandatory update for V3.1B by RDBSERVER ECO 1.

2.2.4 A Partitioned Sorted Index Stored the First Record Incorrectly

A partitioned index allows you to specify limits for partitioning the index keys across multiple storage areas, as in the following example:

```
CREATE INDEX EMPLOYEES_INDEX
    ON EMPLOYEES(LAST_NAME)
    TYPE IS SORTED
    STORE USING (LAST_NAME)
    IN AREA_1 WITH LIMIT OF ("MILLER")
    OTHERWISE IN AREA_2;
```

If a table had a sorted partitioned index and at least one other index (partitioned or single area), the index keys for the first row stored in the table were stored in the wrong storage area. This problem could lead to an inability to select the first row stored in the table (if the index was used), or it could lead to a bugcheck. If you have a table with a partitioned index and at least one other index (partitioned or single area), you can work around this problem by dropping the indexes on the table and re-creating them.

This problem is fixed in the mandatory update for V3.1B by RDMSHRP ECO 10. This mandatory update *will not* fix existing indexes with this problem. You must delete and redefine your indexes to repair them. New tables loaded after applying the mandatory update to V3.1B are not subject to this problem. However, tables that were loaded prior to Version 3.1B still require the previous workaround to correct the problem.

2.2.5 Certain Queries with Intended MODIFY Operations Within Read/Write Transactions Caused Unnecessary Writes to the .AIJ File

Certain queries within read/write transactions wrote journal records to the after-image journal (.AIJ) file unnecessarily. This resulted in extra I/O operations and excess disk space usage for the .AIJ file. This generally happened when a query read some records with the intent of later updating some of them.

There is no workaround for this problem.

This problem is fixed in the mandatory update for V3.1B by RDMSHRP ECO 9.

2.2.6 Partitioned Sorted Indexes Caused Bugchecks

Database operations involving partitioned sorted (B-tree) indexes caused PSII$REMOVE_BOTTOM and PSII$INSERT_BOTTOM bugchecks. An analysis of the bugchecks showed that the partition used was always one off of the partition that should have been used.

There is no workaround for this problem other than to avoid the use of partitioned sorted indexes.

This problem is fixed in the mandatory update for V3.1B by RDMSHRP ECO 11.

2.2.7 A MODIFY Operation Caused Index Corruption on Partitioned Hash Indexes

Index corruption was possible with partitioned hash indexes on SQL UPDATE or RDO MODIFY operations where the index key changed. This corruption occurred only when the table had compression disabled.

This problem occurred if all of the following were true:

- A hashed index was defined on the table.
- The hashed index was partitioned over multiple storage areas.
- The data in the table was uncompressed.
- The operation was an SQL UPDATE or an RDO MODIFY operation.
- The hash key was altered as part of the update.

If all of these conditions were true, a hash key could be stored in the wrong storage area during the update operation. This could lead to the following as a result of this corruption:

- Deleting a record or changing the key of a record could result in the PSIHASH$DELETE + 00000051 bugcheck exception.
- Searching for a record by direct key lookup (using the hashed index) could result in no records returned when records should have been returned.

When the index key changed, Rdb/VMS put the hash bucket in the wrong partition. If you have this problem, you can correct your indexes by re-creating them. The problem could be worked around by eliminating any of the previously stated conditions. For instance, the table could be unloaded and reloaded with compression enabled.

This problem is fixed in the mandatory update for V3.1B by RDMSHRP ECO 12. This mandatory update *will not* fix indexes that already exist with this problem. You must delete and redefine your indexes to repair them.

2.2.8 Partitioned Sorted Indexes Resulted in Various Problems

Any of the following problems was possible when you used partitioned sorted indexes:

- A bugcheck occurred with an exception at LCKCCH$LOCK_RET_NOT_ OK+15.
- "%RDMS-F-NOT_READY, storage area !AC not in ready mode" error message.
- During an IMPORT operation, a bugcheck dump occurred with the following two exceptions:
  - PIO$READY+23A, %RDMS-F-AREA_CORRUPT
  - LCKCCH$LOCK_RET_NOT_OK+15, %SYSTEM-F-ACCVIO
- Incorrect results were returned from queries.
- Inconsistent data or indexes were found as a result of the previous problem.
- A bugcheck occurred with an exception at PSIISCAN$END_SCAN + 0D.

These problems generally occurred when one or more index partitions had no index records stored in it.

There is no workaround for these problems.

These problems are fixed in the mandatory update for V3.1B by RDMSHRP ECO 13, which has been superseded by RDMSHRP ECO 15. This mandatory update *will not* fix indexes that already exist with this problem. You must delete and redefine your indexes to repair them.

2.2.9 With Compression Disabled, Altering the Storage Map STORE Clause and Then Selecting a Row Resulted in a Bugcheck

If you had compression disabled on a storage map and used the SQL STORE IN clause of the ALTER STORAGE MAP statement or the RDO STORE WITHIN clause of the CHANGE STORAGE MAP statement, the data in the table or relation could be unusable. For example, when you tried to access your data in

SQL with the following statement, you would get the following exception:

```
SQL> SELECT * FROM JOBS;

***** Exception at 001B08A2 : RDMS$$EXE_NEXT + 000003C6
```

If you dumped the pages for the new storage area, you would notice that the record length for the records was incorrect. This is what caused the bugcheck to occur when you tried to read your data. Records inserted after the ALTER STORAGE MAP would be stored with the correct record size.

If you have this problem, then you should roll back the ALTER STORAGE MAP if you still can. Otherwise, if you need the information in the table, you should go to your backups.

There are two workarounds:

- Do not disable compression on your storage maps.
- To change the STORE IN or STORE WITHIN clause, perform the following steps:
  1. Enable compression on the storage map.
  2. Alter the storage map STORE IN or STORE WITHIN clause.
  3. Disable compression on the storage map.

This problem is fixed in the mandatory update for V3.1B by RDMSHRP ECO 2.

2.2.10 A Bugcheck Sometimes Resulted When a Sorted Index Rebalanced Itself

Occasionally a sorted or B-tree index in the process of rebalancing itself caused a bugcheck with the following exception: PSIINDEX$JOIN_SCR + 9A.

This problem is fixed in the mandatory update for V3.1B by RDMSHRP ECO 4.

2.2.11 Problem Occurred When Rdb/VMS and the User Application Both Allocated Event Flag 63

In some instances, Rdb/VMS 3.1B allocated event flag 63. This interfered with user-written application programs that also used event flag 63. The following program illustrates the problem:

```
        program         test_efn

        integer*4       efn

        integer*4       sys$waitfr
        integer*4       lib$get_ef
        integer*4       sys$clref

C ...   Some SQL declarations

        EXEC SQL INCLUDE SQLCA

        EXEC SQL DECLARE test_db SCHEMA FILENAME
        1       'test:[rdb]personnel'

        EXEC SQL DECLARE TEST_CURSOR CURSOR FOR
        1       SELECT EMPLOYEE_ID
        1       FROM test_db.EMPLOYEES
        1       ORDER BY LAST_NAME, EMPLOYEE_ID

c ...   Get a free event flag (it will be 63)
```

```
        istatus = lib$get_ef ( efn )
        if ( .not. istatus )  call lib$signal ( %val(istatus) )
        print '( 2z12.8 )', efn, istatus
c ...   Make sure it is clear
c       istatus = sys$clref ( %val(efn) )
        if ( .not. istatus )  call lib$signal ( %val(istatus) )
c ...   Do the SQL/Rdb stuff
        EXEC SQL SET TRANSACTION READ ONLY
        1       RESERVING test_db.EMPLOYEES FOR SHARED READ
        EXEC SQL OPEN TEST_CURSOR
        EXEC SQL CLOSE TEST_CURSOR
        EXEC SQL ROLLBACK
        EXEC SQL FINISH
c ...   Now wait for our event flag to be set (it shouldn't, but will be...)
        istatus = sys$waitfr ( %val(efn) )
        if ( .not. istatus )  call lib$signal ( %val(istatus) )
        print *, 'efn is set, but should not be...'
        end
```

This problem is fixed in the mandatory update for V3.1B by RDMSHRP ECO 17.

2.2.12 Query Returned Records in Wrong Order with the SQL ORDER BY <column-name> DESCENDING or the RDO SORTED BY DESCENDING <field> Clauses

Queries using the SQL ORDER BY <column-name> DESCENDING clause or the RDO SORTED BY DESCENDING <field> clause returned rows in the incorrect order. Consider the following SQL defined database and query that shows this problem:

```
CREATE SCHEMA FILENAME MIKE;
CREATE TABLE T1 (F1 CHAR(2),F2 CHAR(3),F3 DATE,F4 SMALLINT,F5 INTEGER);
CREATE TABLE T2 (F1 CHAR(2),F2 CHAR(3),F3 DATE,F4 SMALLINT,F6 DATE);
COMMIT;
CREATE UNIQUE INDEX T1_I ON T1 (F3,F5,F2,F1,F4);
CREATE UNIQUE INDEX T2_I ON T2 (F3,F4,F2,F1);
COMMIT;
INSERT INTO T1 VALUES('NW','VIC','1-OCT-1990',1,80531);
INSERT INTO T1 VALUES('NW','VIC','29-OCT-1990',1,80531);
INSERT INTO T2 VALUES('NW','VIC','1-OCT-1990',1,'1-JAN-1990');
INSERT INTO T2 VALUES('NW','VIC','29-OCT-1990',1,'1-JAN-1990');
COMMIT;
SELECT P.F3, P.F6 FROM T2 P, T1 I
        WHERE I.F1="NW" AND I.F2="VIC" AND I.F5=80531 AND P.F1="NW" AND
              P.F2="VIC" AND I.F4=P.F4 AND I.F3=P.F3 AND P.F3 <= "29-OCT-1990"
        ORDER BY P.F3 DESC;
```

This query returned P.F3 not sorted in descending order:

```
 P.F3                      P.F6
  1-OCT-1990 00:00:00.00    1-JAN-1990 00:00:00.00
 29-OCT-1990 00:00:00.00    1-JAN-1990 00:00:00.00
2 rows selected
```

The query should have returned P.F3 sorted in descending order:

```
 P.F3                      P.F6
 29-OCT-1990 00:00:00.00    1-JAN-1990 00:00:00.00
  1-OCT-1990 00:00:00.00    1-JAN-1990 00:00:00.00
2 rows selected
```

This problem is fixed in the mandatory update for V3.1B by RDMSHRP ECO 18.

2.2.13 Negate Operator Incorrectly Propagated the NULL Bit While Processing a Record Stream

In V3.0, V3.1, V3.1A, V3.1B, and V4.0 of Rdb/VMS, the negate operator incorrectly propagated the NULL bit while processing a record stream.

For instance, of the six values returned from a query the third was NULL. If the negate operator was used, then the third and subsequent values were also returned as NULL.

In RDO, RDBPRE, and RDML the symptoms were zeros (the default MISSING_ VALUE) returned for the subsequent column values.

The following example demonstrates this problem:

```
SQL> CREATE DATABASE FILENAME FOO;
SQL> CREATE TABLE T(A INTEGER, B INTEGER);
SQL> INSERT INTO T VALUES (1, 10);
1 row inserted
SQL> INSERT INTO T VALUES (2, 10);
1 row inserted
SQL> INSERT INTO T VALUES (3, NULL);
1 row inserted
SQL> INSERT INTO T VALUES (4, 10);
1 row inserted
SQL> INSERT INTO T VALUES (5, 10);
1 row inserted
SQL> INSERT INTO T VALUES (6, 10);
1 row inserted
SQL> COMMIT;
SQL> SELECT A, B FROM T;
           A             B
           1            10
           2            10
           3          NULL
           4            10
           5            10
           6            10
6 rows selected
SQL> SELECT A, -B FROM T;
           A
           1           -10
           2           -10
           3          NULL
           4          NULL   <-- should be -10
           5          NULL   <-- should be -10
           6          NULL   <-- should be -10
6 rows selected
SQL> COMMIT;
```

A workaround is to subtract the value from zero, instead of using the negate operator.

This problem is fixed in the mandatory update for V3.1B by RDMSHRP ECO 22. The following query shows the correct results:

```
SQL> SELECT A, 0 - B FROM T;
           A
           1                    -10
           2                    -10
           3                   NULL
           4                    -10
           5                    -10
           6                    -10
6 rows selected
```

2.2.14 Query with Computed-By and OR Index Retrieval Strategy Returned Incorrect Results

The following example demonstrates a problem in which a query with a computed-by column and using OR index retrieval strategy returned incorrect results.

```
INVOKE DATA FILE PERSONNEL
CHANGE RELATION DEGREES. DEFINE MIKE COMPUTED BY YEAR_GIVEN + 0.
END.
COMMIT
FOR D IN DEGREES WITH D.MIKE=1981 AND
  (D.EMPLOYEE_ID="00234" OR D.EMPLOYEE_ID="00238")
  PRINT D.EMPLOYEE_ID,D.YEAR_GIVEN,D.MIKE END_FOR

EMPLOYEE_ID   YEAR_GIVEN          MIKE
00234               1981          1981
00238               1981          1981
00238               1980          1980
```

In this example, the MIKE column is a computed-by field and there is an index with EMPLOYEE_ID as the first segment. The previous query returns incorrect results because the column MIKE should only contain 1981 values.

The following query returns correct results:

```
FOR D IN DEGREES WITH D.YEAR_GIVEN=1981 AND
  (D.EMPLOYEE_ID="00234" OR D.EMPLOYEE_ID="00238")
  PRINT D.EMPLOYEE_ID,D.YEAR_GIVEN,D.MIKE END_FOR

EMPLOYEE_ID   YEAR_GIVEN          MIKE
00234               1981          1981
00238               1981          1981
```

This problem occurred because there was a predicate involving a computed-by field. The value for this field was computed at the first leg of the OR index retrieval, but not at the other leg. So when the computed-by field was evaluated at the first leg of the nested MRSS tree, the BOOL check on top of the topmost MRSS node worked correctly only when data came from the first leg.

This problem is fixed in the mandatory update for V3.1B. This problem is also fixed in V4.0.

2.3 RDO, RDBPRE, and RDML Problems Fixed in the Mandatory Update for V3.1B

This section contains notes and problems fixed in the RDO, RDBPRE, and RDML interfaces.

2.3.1 RDBPRE Generated Incorrect Code for Request Handles

The following run-time error occurred in RDBPRE programs that used request handles:

```
%RDB-F-BAD_REQ_HANDLE, invalid request handle
%TRACE-F-TRACEBACK, symbolic stack dump follows
module name     routine name                     line       rel PC    abs PC
                                                          00054A23  00054A23
----- above condition handler called with exception 0138803C:
%RDB-F-BAD_REQ_HANDLE, invalid request handle
----- end of exception message
RDB$MIKE        RDB$START_1_942272_82CD84                 0000003A  0000074A
MIKE$MAIN       MIKE$MAIN                          40     0000003B  0000063B
```

This problem was caused by RDBPRE, which generated incorrect MACRO code from the source program. Consider the following RDBPRE BASIC program:

```
10      !
&rdb&   INVOKE DATABASE FILENAME "MF_PERSONNEL"
&rdb&   DECLARE_STREAM (REQUEST_HANDLE DIDDLE%) EMP_STREAM
&rdb&   DECLARE_STREAM (REQUEST_HANDLE DIDDLE%) EMP_STREAM
&rdb&   USING EMP IN EMPLOYEES WITH EMP.EMPLOYEE_ID > A$
&rdb&   START_STREAM EMP_STREAM
&rdb&   ON ERROR
            GOTO ERROR_HANDLER
&rdb&   END_ERROR
        STOP
error_handler:
        CALL LIB$STOP(RDB$STATUS)
        END
```

If you compile this program with RDBPRE, save the MACRO code by defining the logical name RDMS$KEEP_PREP_FILES as ''Y'', and inspect the MACRO code in the .MAR file, you will find the following code section:

```
.ENTRY  Rdb$START_1_942272_82CD84,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
MOVAB   G^Rdb$L_TRANSACTION_HANDLE,R6
MOVL    8(AP),R5                ; Request handle
MOVAL   G^RDB$DBHANDLE,R4               ; DBhandle
MOVAB   G^RDMS$GX_BLR_1_942272_82CA76,R8        ; BLR
MOVL    #<RDMS$GX_BLR_1_942272_82CA76_E-RDMS$GX_BLR_1_942272_82CA76>,
    R7 ; BLR length
MOVL    8(AP),R11       ; Message buffer
MOVL    #MESSAGE$1_2_LENGTH,R10
MOVL    #2,R9
$BSBW   Rdb$DEF_START_TXN_R2_TO_R11
BLBS    R0,10000$
CALLG   @4(AP),G^LIB$STOP
```

Inspect the lines with "Request handle" and "Message buffer" comments and note that both lines refer to 8(AP). In this case, the request handle should be 12(AP).

This problem is fixed in the mandatory update for V3.1B by RDBPRE ECO 1.

2.4 RMU Problems Fixed in the Mandatory Update for V3.1B

This section contains notes and problems fixed in the RMU interface.

2.4.1 RMU/VERIFY Returned Spurious Errors Involving Fragmented Records

When an RMU/VERIFY operation was performed the following errors could be incorrectly returned:

```
%RMU-W-INVRELID, invalid relation id at dbkey 61:9198:4
          expected relation id 28, found 12848

%RMU-W-BADFRALEN, area ANNIE_AREA, page 9543, line 24
          storage record UNKNOWN, bad expanded fragment length
          expected: 34, found: 25,
%RMU-I-FRACHNPOS, pointed to by fragment on page 9198, line 21

%RMU-W-BADFRAPTR, area ANNIE_AREA, page 9195, line 19
          storage record UNKNOWN, bad fragment chain pointer
          expected 1 through 15009, found page number 1605660.

%RMU-W-BADFRAEND, area ANNIE_AREA, page 304, line 8
          storage record UNKNOWN, bad last fragment pointer
          expected: 304:8, found: 132864:1.
```

In addition, %SYSTEM-W-ROPRAND errors and bugcheck dumps could result from this problem.

This problem was caused by a buffer-flushing anomaly in the RMU/VERIFY command when RMU/VERIFY was processing record fragments.

A workaround is to restructure your database to avoid record fragmentation.

This problem is fixed in the mandatory update for V3.1B by RMU ECO 1.

2.4.2 Attempting to Recover a Database from an .AIJ File Using the RMU/RECOVER Command or RDO RECOVER Statement Caused an Exception Condition

A PIOFETCH$WITHIN_DB+39 exception was possible when you attempted to recover a database from an .AIJ file using either the RMU/RECOVER command or RDO RECOVER statement.

This could happen under either of the following two circumstances:

- A logical area was created in a uniform storage area and the storage area had not been updated during the rollforward.
- A logical area was created in a mixed format storage area and the first .AIJ update record to the storage area belonged to the logical area just created.

A logical area could be created by creating a table, creating or altering an index, or creating or altering a storage map.

Some operations mentioned here could involve the creation of multiple logical areas. The problem occurred for multiple logical areas as well as for single logical areas.

The only workaround is to ensure that the storage areas that will get updated with the creation of new logical areas have .AIJ update records for the storage areas before the creation of the new logical areas.

This problem is fixed in the mandatory update for V3.1B by RMU ECO 2. Note that the problem is not fixed in the RDO RECOVER statement.

Note

Use the RMU/RECOVER command to recover your databases.

Note that the problem does not cause corruption in the database or in the .AIJ file; you simply cannot apply the .AIJ file to the database.

3

Known Problems, Restrictions, and Other Notes for the Mandatory Update for Rdb/VMS V3.1B

This chapter describes known problems and restrictions relating to Rdb/VMS V3.1B, and includes workarounds where appropriate. It also contains other information not discussed in the preceding chapters.

3.1 Problems, Restrictions, and Notes for All Interfaces

This section contains problems, restrictions, and other notes that pertain to all interfaces.

See also the Restrictions chapter of the V3.1 *VAX Rdb/VMS Release Notes* for other restrictions that still apply.

3.1.1 Using Quoted Threshold Values for Binary Data Types for Partitioning Data or Indexes Results in Data or Index Corruption

Data or index corruption can result from using quotes around threshold values for the signed quadword, signed longword, signed word, and signed byte data types when you partition data and indexes, as shown in the following example:

```
RDO>
DEFINE DATABASE MIKE DICTIONARY IS NOT USED
  DEFINE STORAGE AREA ST1 FILENAME ST1 END ST1 STORAGE AREA
  DEFINE STORAGE AREA ST2 FILENAME ST2 END ST2 STORAGE AREA
  DEFINE STORAGE AREA ST3 FILENAME ST3 END ST3 STORAGE AREA
  DEFINE STORAGE AREA ST4 FILENAME ST4 END ST4 STORAGE AREA
  DEFINE STORAGE AREA ST5 FILENAME ST5 END ST5 STORAGE AREA
  DEFINE STORAGE AREA ST6 FILENAME ST6 END ST6 STORAGE AREA
  DEFINE STORAGE AREA ST7 FILENAME ST7 END ST7 STORAGE AREA.
DEFINE FIELD BILL DATA SIGNED QUADWORD.
DEFINE FIELD ACCT_NO DATA TEXT SIZE 5.
DEFINE RELATION ACCT. BILL. ACCT_NO. END.
COMMIT
DEFINE INDEX I1 FOR ACCT DUPLICATES ARE NOT ALLOWED
  STORE USING BILL WITHIN
  IST1 WITH LIMIT OF "01";IST2 WITH LIMIT OF "03";IST3 WITH LIMIT OF "05";
  IST4 WITH LIMIT OF "07";IST5 WITH LIMIT OF "09";IST6 WITH LIMIT OF "11";
  IST7.
  BILL. ACCT_NO. END.
COMMIT
DEFINE STORAGE MAP ACCT_MAP FOR ACCT RELATION
  STORE USING BILL WITHIN
  ST1 WITH LIMIT OF "01";ST2 WITH LIMIT OF "03";ST3 WITH LIMIT OF "05";
  ST4 WITH LIMIT OF "07";ST5 WITH LIMIT OF "09";ST6 WITH LIMIT OF "11";
  ST7 END.
COMMIT
```

```
START_TRANSACTION READ_WRITE
STOR A IN ACCT USING A.BILL=1;A.ACCT_NO="1" END_STORE
STOR A IN ACCT USING A.BILL=11;A.ACCT_NO="11" END_STORE
STOR A IN ACCT USING A.BILL=3;A.ACCT_NO="3" END_STORE
STOR A IN ACCT USING A.BILL=5;A.ACCT_NO="5" END_STORE
STOR A IN ACCT USING A.BILL=15;A.ACCT_NO="15" END_STORE
STOR A IN ACCT USING A.BILL=7;A.ACCT_NO="7" END_STORE
STOR A IN ACCT USING A.BILL=9;A.ACCT_NO="9" END_STORE
COMMIT
FOR A IN ACCT WITH A.BILL EQ 5 PRINT A.* END_FOR
FOR A IN ACCT WITH A.BILL LE 5 PRINT A.* END_FOR
```

If you have data and index records stored in the wrong partition for a table, reload the table with a storage map that does not use quotes around the threshold values for the signed quadword, signed longword, signed word, and signed byte data types. Note that quotes are permitted around the text data type values and these work correctly.

3.1.2 Problem with SQL LIKE and RDO MATCHING Clauses

The RDO MATCHING clause, the SQL and RDO CONTAINING and STARTING WITH clauses, and the SQL LIKE clause work as expected on character string and integer data types, but all work a little differently with the DATE data types.

In each case, the syntax is accepted, but unexpected results are returned if a character expression is used, because of the conversion of dates to text strings. When a number is used in the MATCHING expression or LIKE predicate, correct results are returned, but if a character expression is used, no results are returned, as shown in these examples.

```
RDO> FOR E IN EMPLOYEES WITH E.BIRTHDAY MATCHING "*1954*"
cont> PRINT E.BIRTHDAY END_FOR
 BIRTHDAY
 20-MAR-1954 00:00:00.00
 13-MAR-1954 00:00:00.00
 21-NOV-1954 00:00:00.00
 15-MAY-1954 00:00:00.00
 20-JUL-1954 00:00:00.00
RDO>
RDO> FOR E IN EMPLOYEES WITH E.BIRTHDAY MATCHING "*mar*"
cont> PRINT E.BIRTHDAY END_FOR
```

The same behavior is seen in SQL with the LIKE clause.

```
SQL> SELECT BIRTHDAY FROM EMPLOYEES WHERE BIRTHDAY LIKE "%1954%";
 BIRTHDAY
 20-Mar-1954
 13-Mar-1954
 21-Nov-1954
 15-May-1954
 20-Jul-1954
5 rows selected
SQL> SELECT BIRTHDAY FROM EMPLOYEES WHERE BIRTHDAY LIKE "%Mar%";
0 rows selected
SQL> SELECT BIRTHDAY FROM EMPLOYEES WHERE BIRTHDAY LIKE "%MAR%";
0 rows selected
SQL>
```

The reason for this apparent inconsistent behavior is as follows. The RDO MATCHING clause, the SQL and RDO CONTAINING and STARTING WITH clauses, and the SQL LIKE clause, all require TEXT as input. Therefore, the dates will be converted to text strings that have the YYYYNNDDHHMMSSCC format described in the *VAX Rdb/VMS SQL Reference Manual*. The match will be performed on all digit text strings of the date ("MAR" will never be seen

because the month value will have been converted to "03"), and then the binary values are returned to SQL or RDO for printing in the format shown.

Use the RDO MATCHING clause with "*03*" or the SQL LIKE clause with "%03%" to get the results you are expecting.

3.1.3 RDB$REMOTE Account Has SYSTEM as Owner

If V3.1B is installed over an older version of Rdb/VMS that has SYSTEM as the owner of its directory, RDB$SERVER cannot create the NETSERVER.LOG file and fails on attach or on the first transaction. A workaround is to use the SET OWNER command to set the owner to RDB$REMOTE.

3.1.4 RDMSHRP_DS Image Displays Incorrect Values

The Rdb/VMS V3.1B RDMSHRP and RDMSHRP_S images were linked with VMS V5.2 libraries. Because the RDMSHRP_DS image was linked with VMS V5.3 libraries, one of the display routines displays a value incorrectly.

3.1.5 An Arithmetic Exception Results When Joining Integer Columns

The following error is possible when a join operation is performed using integer columns of different sizes:

```
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SYSTEM-F-INTOVF, arithmetic trap, integer overflow at PC=xxxxxx,PSL=01C00000
```

The problem occurs only when the columns being joined are each part of an index, and when the larger join column contains data that exceeds the maximum size of the smaller join column. The following example shows the problem.

```
$ RDO
DEFINE DATABASE MIKE DICTIONARY IS NOT USED.
DEFINE FIELD W1 DATA SIGNED WORD.
DEFINE FIELD L1 DATA SIGNED LONG.
DEFINE RELATION R1. W1. END.
DEFINE RELATION R2. L1. END.
DEFINE INDEX I1 FOR R1. W1. END.
DEFINE INDEX I2 FOR R2. L1. END.
STORE R IN R1 USING R.W1=1 END_STORE
STORE R IN R1 USING R.W1=2 END_STORE
STORE R IN R1 USING R.W1=3 END_STORE
STORE R IN R1 USING R.W1=4 END_STORE
STORE R IN R2 USING R.L1=100000 END_STORE
STORE R IN R2 USING R.L1=100000 END_STORE
STORE R IN R2 USING R.L1=100000 END_STORE
COMMIT
FOR A IN R1 CROSS B IN R2 WITH A.W1=B.L1 PRINT A.*,B.* END_FOR
ROLLBACK
EXIT
```

A workaround is to use the larger data type for both columns of the join operation. Note that indexes may have to be redefined as a workaround to this problem.

3.1.6 Collating Sequences That Use Two-to-Two Character Mapping May Bugcheck

When you define a collating sequence that uses two-to-two character mapping, for example Thai collating sequences, a bugcheck may occur with the following exception:

```
EXCEPTION AT 003C2BC1
RDMS$$MCS$NCS_RECORD_8 + 000000401
```

Two-to-two character mapping is not supported in V3.1.

The following example demonstrates the problem.

```
DEFINE DATABASE AX DICTIONARY IS NOT USED.
DEFINE COLLATING_SEQUENCE TEST TEST.
DEFINE FIELD TEST DATATYPE TEXT SIZE 2 COLLATING_SEQUENCE IS TEST.
DEFINE RELATION MIKE.
TEST.
END RELATION.
COMMIT
START_TRANSACTION READ_WRITE
STORE T IN MIKE USING T.TEST = "AA" END_STORE
STORE T IN MIKE USING T.TEST = "AA" END_STORE
STORE T IN MIKE USING T.TEST = "ZZ" END_STORE
STORE T IN MIKE USING T.TEST = "A " END_STORE
STORE T IN MIKE USING T.TEST = "Z " END_STORE
STORE T IN MIKE USING T.TEST = "BA" END_STORE
STORE T IN MIKE USING T.TEST = "AB" END_STORE
STORE T IN MIKE USING T.TEST = "C " END_STORE
FOR T IN MIKE PRINT T.* END_FOR
FOR T IN MIKE SORTED BY T.TEST PRINT T.* END_FOR
SHOW FIELD TEST
COMMIT
```

3.1.7 Synchronization Problem for an Empty Sorted Index

When two simultaneous attaches to the same database access an empty table via a sorted index, one attach may fail to see a row created by the other attach. Note that this problem occurs only in tables whose indexes reside in a single storage area; it does not occur in partitioned indexes nor does it occur in relations whose indexes are unmapped and reside by default in the RDB$SYSTEM logical area. Furthermore, as stated earlier, this problem occurs only if the table is empty when both processes attach to the database. This problem was originally fixed in V3.1B; however, when two released patches (LCKCCH31B.PAT and SORT_FIX_B.PAT) are applied, the problem recurs and produces the following exception:

```
(Exception at 00211A5C : RDMS$$KOD_REMOVE_TREE + 0000046B,
(%RDMS-F-BUGCHECK, fatal, unexpected error detected)
```

The following example shows this problem:

```
SQL> CREATE SCHEMA FILENAME TEST_DB;
SQL> FINISH;

SQL> DECLARE SCHEMA FILE TEST_DB;
SQL> CREATE TABLE CANDIDATES
cont> (LAST_NAME CHAR (15),
cont> FIRST_NAME CHAR (15));
SQL> COMMIT;
SQL> CREATE INDEX CANDIDATES_INDEX ON CANDIDATES
cont> (LAST_NAME, FIRST_NAME) TYPE IS SORTED;
SQL> COMMIT;
SQL> FINISH;

- Table CANDIDATES is empty.

- User #1 attaches to the database.

SQL> DECLARE SCHEMA FILE TEST_DB;
SQL> SET TRANS READ ONLY;
SQL> SEL * FROM CANDIDATES;
0 rows selected
SQL> COMMIT;

- User #2 attaches to the database and inserts a row.
```

```
SQL> DECLARE SCHEMA FILE TEST_DB;
SQL> SET TRANSACTION READ WRITE;
SQL> INSERT INTO CANDIDATES (LAST_NAME, FIRST_NAME)
cont> VALUES ("Doe", "John");
1 row inserted
SQL> COMMIT;

- User #1 wants to delete the row
SQL> SET TRANSACTION READ WRITE;
SQL> DELETE FROM CANDIDATES;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file $2$DUA1:[DB]RDSBUGCHK.DMP;1
%SYSTEM-F-BREAK, breakpoint fault at PC=00000000, PSL=00000000
```

This problem is fixed in V4.1.

3.1.8 Rdb/VMS Does Not Accept the Database File Specification in a Logical Name

Rdb/VMS V3.1 does not accept the database file specification in a logical name in a remote attach if the colon is missing, as shown in the following example:

```
On node A:
$ DEFINE/SYSTEM LOGICALNAME DB$DISK:[RDB]MF_PERSONNEL

On node B:
$ DIR A::LOGICALNAME:

Directory DB$DISK:[RDB]

MF_PERSONNEL.RDB;1

In SQL:
$ SQL
SQL> DECLARE SCHEMA FILENAME A::LOGICALNAME;

%RDB-E-BAD_DB_FORMAT, LOGICALNAME.RDB; does not reference a database known to Rdb
-RMS-E-FNF, file not found

In RDO:
$ RDO
RDO> INVOKE DATABASE FILENAME A::LOGICALNAME

%RDB-E-BAD_DB_FORMAT, LOGICALNAME.RDB; does not reference a database known to Rdb
-RMS-E-FNF, file not found
```

This is a known problem.

A workaround is to define your logical name with a directory specification and include the colon at the end of the logical name, as the following example shows:

```
$ DEFINE/SYSTEM LOGICALNAME DB$DISK:[RDB]

In SQL:
$ SQL
SQL> DECLARE SCHEMA FILENAME LOGICALNAME:MF_PERSONNEL;

In RDO:
$ RDO
RDO> INVOKE DATABASE FILENAME LOGICALNAME:MF_PERSONNEL
```

3.1.9 Constraints Cause Looping and LCKCCH$COMMIT_SUBTREE Bugchecks

When constraints are used in an Rdb/VMS database, an internal database structure queue, record process local locks (RPLL), can become corrupt and cause a commit operation to hang or bugcheck. The bugcheck or hang takes place after the commit point; the data being manipulated is saved and is safe from corruption. The error occurs as part of the memory structure's releases after the commit point.

The following command procedure demonstrates this problem:

```
START_TRANS
STORE RECORD IN RELATION1
COMMIT
START_TRANS
ERASE RECORD FROM RELATION1
COMMIT
!
! dummy transaction
COMMIT
!
! dummy transaction
!
START_TRANS
STORE RECORD IN RELATION2
COMMIT                          <---- loop occurs here
START_TRANS
ERASE RECORD FROM RELATION2
COMMIT
```

The same command procedure with reserving clauses generates a bugcheck dump with the following exception:

```
***** Exception at 006091C8 : LCKCCH$COMMIT_SUBTREE + 00000025
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=00000000,
      PC=006091C8, PSL=01400009
```

This problem does not have a workaround.

This problem is fixed in V4.0. This problem can be fixed in V3.1C by applying the optional patch supplied on the mandatory update kit. See Section 4.1.1 for more information.

3.1.10 Query Optimizer Does Not Use Index-Only Retrieval When the Dbkey Is Selected

Queries that retrieve views by dbkey may perform poorly due to query optimizer strategies that do not always retrieve data from tables by dbkey. This especially affects DATATRIEVE users that use the FIND command to find collections on views and then process them. DATATRIEVE finds collections by building dbkey lists. Further processing is done by asking Rdb/VMS to retrieve the information using the dbkey lists.

Consider the following RDBPRE BASIC program. This program gets the dbkey for the first record in the CURRENT_JOB view and then saves the dbkey. It then reads the record using the saved dbkey.

```
10      !
&RDB&   INVOKE DATABASE FILENAME "MF_PERSONNEL"
&RDB&   FOR FIRST 1 C IN CURRENT_JOB GET DBKEY$=C.RDB$DB_KEY END_GET
END_FOR
        PRINT "**** Retrieving by view dbkey ****"
&RDB&   FOR C IN CURRENT_JOB WITH C.RDB$DB_KEY=DBKEY$ GET A$=C.LAST_NAME
&RDB&     END_GET END_FOR
        END
```

If this program is run with the RDMS$DEBUG_FLAGS logical name defined as ''S'', then you will see the following output:

```
   .
   .
   .
**** Retrieving by view dbkey ****
Cross block of 2 entries
  Cross block entry 1
    Get     Retrieval by index of relation EMPLOYEES
      Index name  EMP_LAST_NAME [0:0]
  Cross block entry 2
    Conjunct        Conjunct        Firstn  Get
    Retrieval by DBK of relation JOB_HISTORY
```

As this shows, the query optimizer chose retrieval by index on EMPLOYEES instead of retrieval by dbkey. This can be a problem where the retrieval on the first table of a view scans thousands of records using sequential retrieval for every dbkey in a dbkey list.

The proper strategy for the view retrieval by dbkey for the same program is:

```
   .
   .
   .
**** Retrieving by view dbkey ****
Cross block of 2 entries
  Cross block entry 1
    Conjunct      Firstn  Get  Retrieval by DBK of relation JOB_HISTORY
  Cross block entry 2
    Conjunct      Conjunct     Firstn  Get
    Retrieval by DBK of relation EMPLOYEES
```

This problem is fixed in V4.1. This problem can be fixed in V3.1B by applying the optional patch supplied on the mandatory update kit. See Section 4.1.2 for more information.

3.1.11 Query Optimizer Chooses an Incorrect Strategy for a Write Operation Within a Selection Loop and Goes into an Infinite Loop

A write operation within a selection loop causes the query optimizer to choose a zigzag retrieval strategy with the use of a temporary relation. This strategy combination causes an infinite CPU loop to occur. The following example query shows this problem:

```
RDO> FOR VU IN VENDOR_UNIT CROSS AU IN ABRV_UNIT OVER UNIT_NO WITH
Cont> VU.VENDOR_NO = "005550" REDUCED TO AU.DIV_NO
Cont> PRINT AU.DIV_NO
Cont> STORE VDD IN VENDOR_DIV_DIST USING VDD.DIV_NO = AU.DIV_NO;
Cont> VDD.VENDOR_NO = "005550"; VDD.DIST_TYPE_CODE = "U";
Cont> END_STORE
Cont> END_FOR
```

In this query, the ABRV_UNIT table has a cardinality of 498 rows and the VENDOR_UNIT table has a cardinality of 31,096 rows. The ABRV_UNIT table has the ABRV_UNIT_PRI_INDEX index defined for it and the column UNIT_NO is the primary segment. The VENDOR_UNIT table has two indexes defined for it and one of these indexes is the VENDOR_UNIT_PRI_IDX index and the columns VENDOR_NO and UNIT_NO are the primary segments. The query optimizer chooses the following strategy:

```
Reduce  Sort    Conjunct
Match
  Outer loop
    Sort    Get     Temporary relation
    Retrieval by index of relation VENDOR_UNIT
    Index name VENDOR_UNIT_PRI_IDX  00000001 Segments in low Ikey
    00000001 Segments in high Ikey
  Inner loop      (zig-zag)
    Get     Temporary relation      Retrieval by index of relation ABRV_UNIT
    Index name ABRV_UNIT_PRI_IDX
```

The result of the query is an infinite loop. This problem is fixed in the V4.0. This problem can be fixed in V3.1B by applying the optional patch supplied on the mandatory update kit. See Section 4.1.3 for more information.

3.1.12 Singleton Subselect Statement Returns Incorrect Results

The following query in an SQL precompiled C program shows an Rdb/VMS problem in which a query involving a singleton subselect can return incorrect results.

The correct result from interactive SQL is as follows:

```
SELECT Y.A,Y.C,(SELECT SUM(X.B) FROM X WHERE X.A=1) FROM Y WHERE Y.A=1;
           A             C
           1            10                      3
1 row selected
```

The result from the precompiled C program is the following second sum, which is incorrect:

```
 SQLCODE = 0
----> sum = 3
 SQLCODE = 0
----> sum = 0 (incorrect result)
```

A workaround is to store the result of the subselect in a temporary variable, as in the first query in the following precompiled C program. The second query in this precompiled C program shows the problem.

```
CREATE SCHEMA FILENAME PROBLEM;
CREATE TABLE X (A INTEGER, B INTEGER);
CREATE TABLE Y (A INTEGER, C INTEGER);
INSERT INTO X VALUES (1,1);
INSERT INTO X VALUES (1,2);
INSERT INTO X VALUES (2,1);
INSERT INTO Y VALUES (1,10);
INSERT INTO Y VALUES (2,20);
COMMIT;
SELECT Y.A, Y.C, (SELECT SUM(X.B) FROM X WHERE X.A = 1) FROM Y WHERE Y.A = 1;

---------------------- Precompiled C Program To Reproduce ----------------------
#include <stdio.h>
#define check_sqlcode printf(" SQLCODE = %d\n", SQLCODE)
main()
{       int SQLCODE, a, c, sum;
        /* This query works */
exec sql declare schema filename problem;
exec sql select sum(x.b) into :sum from x where x.a = 1;
        check_sqlcode;
        printf("----> sum = %d\n", sum);
        /* This query returns the wrong result */
exec sql select y.a, y.c, (select sum(x.b) from x where x.a = 1)
        into :a, :c, :sum from y where y.a = 1;
        check_sqlcode;
        printf("----> sum = %d\n", sum);}
```

This is a known problem in V3.1B and V4.0. This problem is fixed in V4.1.

3.1.13 SPAM Pages Are Not Updated Correctly

SPAM pages show space to store records when in fact pages are full. A full verification (RMU/VERIFY/ALL) of the database produces error messages indicating that the SPAM pages were not correctly updated.

There is no workaround to this problem. The undocumented RMU/REPAIR command could have been used to periodically correct the problem by rebuilding the SPAM pages so SPAM pages display the correct available space on the data pages, but using this command in V3.1B caused database corruption. See Section 3.5.3.

This is a known problem in V3.1, V3.1A, and V3.1B.

3.1.14 Rdb/VMS Monitor Fails When the Last User Finishes on a Particular Database

If the Rdb/VMS monitor process, RDMMON, fails when the last user finishes from a database, an abbreviated bugcheck dump is written to the monitor log file. The exception is one of the following:

```
MON$UNLOCK_MPLL + 00000031
          or
MON$UNLOCK_MPLL + 00000036
          or
MON$UNLOCK_MPLL + 00000049
```

The secondary error message is either a SYSTEM-F-ACCVIO or SYS$SYSTEM-F-INVLOCKID.

This exception is caused by the way in which Rdb/VMS allocates virtual memory. Customers who see this problem generally have storage areas (live and snapshot) that number in the hundreds (the actual number may vary from database to database).

This is a known problem in V3.1B and V4.0. The problem will *not* cause a database to become corrupt; however, if your Rdb/VMS monitor process is failing with the errors just cited, you should apply the optional patch. This problem can be fixed in V3.1C by applying the optional patch supplied on the mandatory update kit. See Section 4.1.4 for more information.

This problem is fixed in V4.1.

3.1.15 Triggers That Affect Subject Table Rows Can Cause Loops or Inconsistent Results

Triggers that update rows of the trigger subject table or add rows to the trigger subject table can cause infinite loops or inconsistent results to be returned. For example, consider the following two conditions:

- A BEFORE UPDATE trigger on table X that inserts a row into table X
- An UPDATE statement affecting all the rows in table X

Given these two conditions, the UPDATE statement will loop until all resources are consumed because for each row updated, a new row will be added, which in turn will be updated, and so forth.

If subject table rows are being retrieved using an index, a triggered action operating on the same table could affect the index (by changing index key values or adding new keys) such that the triggering statement behaves in a different manner than when no trigger is involved.

To avoid this problem, construct any such triggers to operate only on rows that are either the current subject table row, or that will never be selected by the triggering statement. A more difficult avoidance method is to restructure triggering statements so that they never select a row that could have been updated or added by a trigger action. Some circumstances will require a combination of these methods.

3.1.16 Query Using Descending Indexes Returns Incorrect Results

When a query is performed using descending indexes, the query returns incorrect results. The following example shows the problem:

```
CREATE INDEX J_IDX ON JOB_HISTORY (EMPLOYEE_ID DESC);
CREATE INDEX E_IDX ON EMPLOYEES (EMPLOYEE_ID DESC);
!
CREATE VIEW V AS
        SELECT E.EMPLOYEE_ID FROM
                EMPLOYEES E, JOB_HISTORY J
                        WHERE E.EMPLOYEE_ID=J.EMPLOYEE_ID;
!
! The following query is fine.
!
SELECT * FROM V ORDER BY EMPLOYEE_ID;
!
! **** The following query returned incorrect results ****
!
SELECT * FROM V ORDER BY EMPLOYEE_ID DESC;
!
```

This is a known problem in V3.1B. This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 25. This problem is also fixed in V4.1.

3.1.17 Query with a Computed-By Field and OR Logic Returns Incorrect Results

The following query returns incorrect results:

```
FOR R IN R2 WITH (R.M_OVED="0023" AND R.YYYYMM="199101") OR
        (R.M_KARTIS="0018" AND R.YYYYMM >= "199101") PRINT R.* END_FOR
```

The yyyymm field is a computed-by field.

If the query is slightly reworded as follows, it produces the correct results:

```
FOR R IN R2 WITH (R.M_KARTIS="0018" AND R.YYYYMM >= "199101") OR
        (R.M_OVED="0023" AND R.YYYYMM="199101") PRINT R.* END_FOR
```

This is a known problem in V3.1B and V4.0. The problem was caused by the incorrect optimization of reusing the result of evaluation of a common subquery in this case, the computed-by expression used in both OR legs.

This problem is fixed in V4.1.

3.1.18 NOWAIT Transactions Have Their Buffers Invalidated at COMMIT

Programs that use NOWAIT transactions have their buffers invalidated at commit time. This forces Rdb/VMS to read the data again as can be observed by higher than expected DIO rates. This is a known problem in V3.1B and V4.0.

A workaround is to use WAIT transactions.

3.2 SQL Problems, Restrictions, and Notes

This section describes problems, restrictions, and other information of interest to users of the SQL interface.

See also the Restrictions chapter of the V3.1 *VAX Rdb/VMS Release Notes* for other restrictions that still apply.

3.2.1 Using the IGNORE CASE Option of the LIKE Clause Sometimes Results in a Query That Incorrectly Returns No Rows

In certain SELECT statements in SQL, the LIKE clause works correctly unless an IGNORE CASE option is added to it. The IGNORE CASE option causes the query to return no rows. This seems to occur when the WHERE clause involves more than one condition connected with an AND Boolean operator.

The following command procedure illustrates this problem. The first SELECT statement correctly returns the inserted record. Adding an IGNORE CASE option to one of the items in the WHERE clause results in the return of no rows. Adding another IGNORE CASE option to a different item in the WHERE clause seems to fix the problem.

```
$ SQL
DECLARE SCHEMA FILE PERSONNEL;
CREATE TABLE XXX (F1 CHAR(5), F2 CHAR(3), F3 CHAR(3));
INSERT INTO XXX (F1, F2, F3) VALUES ("ABC", "ABC", "ABC");
SELECT * FROM XXX WHERE
      F1 LIKE "AB%" AND
      F2 LIKE "ABC" AND
      F3 LIKE "ABC";
SELECT * FROM XXX WHERE
      F1 LIKE "AB%" AND
      F2 LIKE "ABC" IGNORE CASE AND
      F3 LIKE "ABC";
SELECT * FROM XXX WHERE
      F1 LIKE "AB%" AND
      F2 LIKE "ABC" IGNORE CASE AND
      F3 LIKE "ABC" IGNORE CASE;
ROLLBACK;
```

This is a known problem in V3.1, V3.1A, V3.1B, and V4.0.

A workaround is to add additional IGNORE CASE clauses to the WHERE clause as previously mentioned, as this seems to correct this problem.

3.3 SQL/Services Problems, Restrictions, and Notes

This section describes problems, restrictions, and other information of interest to users of SQL/Services.

See also the Appendix on SQL/Services Release Notes in the V3.1 *VAX Rdb/VMS Release Notes* for other restrictions that still apply.

3.3.1 SQL/Services VMS API Shipped with the Rdb/VMS Run-Time Kit

In the mandatory update for Version 3.1B, the SQL/Services VMS API is included in the Rdb/VMS run-time kit. With the Rdb/VMS run-time kit installed, you can execute previously developed SQL/Services client applications; however, you cannot develop SQL/Services client applications. Development of SQL/Services client applications is possible only with the Rdb/VMS full development kit.

3.4 RDO, RDBPRE, and RDML Problems, Restrictions, and Notes

See the Restrictions chapter of the V3.1 *VAX Rdb/VMS Release Notes* for restrictions that still apply to users of RDO, RDBPRE, and RDML.

3.5 Rdb/VMS Management Utility (RMU) Problems, Restrictions, and Notes

This section contains problems, restrictions, and other notes that pertain to the Rdb/VMS Management Utility (RMU).

See also the Restrictions chapter of the V3.1 *VAX Rdb/VMS Release Notes* for other restrictions that still apply.

3.5.1 Do Not Delete After-Image Journal (.AIJ) Backup Files If the AIJ Backup Fails or Is Terminated

If an AIJ backup process fails or is prematurely terminated, one possible action the user might take would be to discard the resultant .AIJ backup file because the backup operation was not completed. However, all .AIJ backup files, including those produced by a failed backup process, are necessary to recover a database. If an .AIJ backup file of a failed backup process was discarded, the database would not be recoverable from that point forward. This is especially important if you use magnetic tapes as the .AIJ backup media; in this case, preserve this magnetic tape and do *not* reuse it.

When an AIJ backup process, especially one running in continuous (/CONTINUOUS) mode, writes to the .AIJ backup file, it is possible for the transferred data to be deleted from the database .AIJ file. If the backup process subsequently fails or is prematurely terminated (for example, with Ctrl/Y or the STOP command), it might not be possible to retransfer the data to the subsequent .AIJ backup file because the data was deleted from the active database .AIJ file.

Therefore, it is extremely important that you preserve *all* .AIJ backup files, even those produced by failed or terminated backup processes. If the resultant .AIJ backup file is discarded, the next .AIJ backup file could contain a ‘‘gap’’ in transactions, so that no transactions would ever be rolled forward from that point on.

Note

If this problem occurs, the database is *not* inconsistent or corrupt. Rather, the database cannot be rolled forward past the discarded .AIJ backup file.

The solution to this problem is to preserve all .AIJ backup files to ensure that a database can be completely recovered.

If you have discarded an .AIJ backup file, immediately perform a complete database backup to ensure that the database can be recovered up to the current transaction.

3.5.2 EXPORT Operations Fail with an Access Violation When the Database Has a Default Collating Sequence Defined

If an Rdb/VMS V3.1, V3.1A, V3.1B, or V3.1C database has a default collating sequence and is exported, an access violation results. There is no workaround to this problem. However, Section 6.7.2 describes a problem that was fixed in the V4.0 mandatory update that permits a database that has a default collating sequence to be converted to V4.0.

3.5.3 Use of Undocumented RMU/REPAIR Command Corrupts Databases

The RMU/REPAIR command is an undocumented command in V3.1. Several V3.1 problems were fixed in V3.1B (see Section 3.6.1 in the V3.1B *VAX Rdb/VMS Release Notes*). However, several additional problems were discovered in the V3.1B RMU/REPAIR command that caused database corruption when this command was used. Therefore, avoid using this undocumented command.

These problems are fixed in V4.0A.

3.6 Notes and Restrictions Related to CDD/Plus

This section describes problems and restrictions relating to the use of Rdb/VMS with CDD/Plus.

See also the Restrictions chapter of the V3.1 *VAX Rdb/VMS Release Notes* for other restrictions that still apply.

3.6.1 Restrictions Lifted by CDD/Plus Version 4.3

The restrictions listed in the V3.1B *VAX Rdb/VMS Release Notes*, Section 4.5.2, have been lifted by VAX CDD/Plus Version 4.3.

3.7 Rdb/VMS Documentation Errors and Omissions in V3.1B

This section describes errors or omissions in the Rdb/VMS manuals and documents.

3.7.1 Documentation Error in V3.1 *VAX Rdb/VMS SQL Reference Manual*, Appendix D.4

In Appendix D.4, Table D-2, in the *VAX Rdb/VMS SQL Reference Manual*, the values for the column labeled Value are correct. The stated values indicate that if the parameter marker or select list item data type allows null values, the SQLTYPE field value returned is as cited. The actual value cited includes the null bit. In V4.0, Appendix D.4, Table D-2, reference is made in footnote one that this value is the base value plus one for the values cited. All versions of the Rdb/VMS SQL interface since and including VAX SQL V1.0 have not included this functionality indicated as the base value plus one. Such a feature has never been supported in any released version of the SQL interface for Rdb/VMS.

This table is correct and updated in the documentation for V4.1.

The only known solution for finding the state of the null vector is to query the metadata and then set a program flag that indicates the 'null value allowed' state.

4

Optional ECO Patches for the Mandatory Update for VAX Rdb/VMS V3.1B

The .A saveset of this mandatory update kit contains optional ECO patches for the mandatory update for V3.1B that you can install after the mandatory update for V3.1B is installed. These patches are optional for one or more of the following reasons:

- The patch has reported side effects.
- The patch changes the query optimizer behavior, and some customers may not want the changed behavior.
- The patch will be needed by only a few customers and the problem does not involve data corruption or incorrect results.
- The patch was not completed in time for field test and is therefore not part of the final kit.

If you do not need the patch (that is, you do not have the problem), then probably you should not install the optional patch. Please read each patch article and then follow the specific instructions for any patch you decide to apply.

4.1 Optional ECO Patches That Can Be Applied to the Mandatory Update for Rdb/VMS V3.1B

This section contains the optional ECO patches that can be applied to the mandatory update for Rdb/VMS V3.1B.

4.1.1 RDMSHRP ECO 1: Constraints Cause Looping and LCKCCH$COMMIT_SUBTREE Bugchecks

See Section 3.1.9 for a description of this problem.

This patch has been available for V3.1, V3.1A and V3.1B since June, 1990, and side effects have been reported with it. Many customers, however, use it with no problems. The patch article file name is RDB31C_RDMSHRP_ECO01.ARTICLE.

To get this patch from the mandatory update kit, use the following command:

```
$ BACKUP <DEVICE>RDBVMS_MUPA040.A/SAV/SEL=RDB31C_RDMSHRP_ECO01.ARTICLE *.*
```

4.1.2 RDMSHRP ECO 14: Query Optimizer Does Not Use Index-Only Retrieval When the Dbkey Is Selected

See Section 3.1.10 for a description of this problem.

This patch has been available for V3.1B since March, 1991, to modify a query optimizer behavior in which queries that return dbkeys can avoid the use of beneficial indexes. This patch can change query optimizer behavior to improve performance in some cases. The patch article file name is RDB31C_RDMSHRP_ECO14.ARTICLE.

To get this patch from the mandatory update kit, use the following command:

```
$ BACKUP <DEVICE>RDBVMS_MUPA040.A/SAV/SEL=RDB31C_RDMSHRP_ECO14.ARTICLE *.*
```

4.1.3 RDMSHRP ECO 19: Query Optimizer Chooses an Incorrect Strategy for a Write Operation Within a Selection Loop and Goes into an Infinite Loop

See Section 3.1.11 for a description of this problem.

This patch has been available for V3.1B since April, 1991, to fix an infinite loop problem. This problem has been reported by only a few customers. The patch article file name is RDB31C_RDMSHRP_ECO19.ARTICLE.

To get this patch from the mandatory update kit, use the following command:

```
$ BACKUP <DEVICE>RDBVMS_MUPA040.A/SAV/SEL=RDB31C_RDMSHRP_ECO19.ARTICLE *.*
```

4.1.4 RDMMON ECO 1: Rdb/VMS Monitor Fails When the Last User Finishes on a Particular Database

See Section 3.1.14 for a description of this problem.

This patch resolves a problem in which the Rdb/VMS V3.1B monitor fails when the last user finishes on a database. This causes an abbreviated bugcheck dump to be written to the monitor log file. The exception is one of the following:

```
MON$UNLOCK_MPLL + 00000031
          or
MON$UNLOCK_MPLL + 00000036
          or
MON$UNLOCK_MPLL + 00000049
```

The patch is optional because it was not completed in time to be placed on the field test kit and therefore is not part of the final kit. The patch article file name is RDB31C_RDMMON_ECO01.ARTICLE.

This problem is fixed in V4.1.

To get this patch from the mandatory update kit, use the following command:

```
$ BACKUP <DEVICE>RDBVMS_MUPA040.A/SAV/SEL=RDB31C_RDMMON_ECO01.ARTICLE *.*
```

Part II

Mandatory Update for VAX Rdb/VMS Version 4.0

This part contains Chapters 5 through 8 and describes information pertaining to the mandatory update for VAX Rdb/VMS Version 4.0. This part also contains Appendix A and Appendix B.

5

Installing the Rdb/VMS Mandatory Update for Version 4.0

This chapter describes information necessary for installing the mandatory update for Rdb/VMS Version 4.0.

5.1 Before Installing the Rdb/VMS Mandatory Update Package for Rdb/VMS Version 4.0

You must install Rdb/VMS V4.0 before you install the mandatory update package (MUP). See the *VAX Rdb/VMS Installation Guide* for instructions. The following sections describe what steps you need to take to install the MUP. The installation of the MUP checks to see whether you have Rdb/VMS V3.1B or Rdb/VMS V4.0 installed on your system. It then installs either V3.1C or V4.0A. The name of the mandatory update kit is RDBVMS_MUPA040. This name appears in installation messages whether you are installing the MUP for V3.1B or V4.0.

5.1.1 Prerequisite Hardware and Software

This section discusses the hardware and software you must have installed on your system before you install the mandatory update package of Rdb/VMS.

You can install the MUP only when your system meets or exceeds the minimum hardware requirements as shown in the SPD. Table 5–1 lists the approximate system disk storage required for the installation of the MUP. Your system may require additional mass storage for backup and restore operations.

The VMS operating system Version 5.3 or higher must be installed on your VAX system if you are installing the mandatory update package for V4.0 of Rdb/VMS (V4.0A).

Rdb/VMS V4.0 must be installed before you install the MUP. If this version is not present, the installation aborts.

The installation uses the RDO.EXE image to check the existing Rdb/VMS version number. If you have deleted your RDO.EXE image, you must restore it from the saveset using the VMS BACKUP command. For example, to restore RDO.EXE from the full development kit, enter the following command:

```
$ BACKUP <device>:RDBVMSDEV040.F/SAVE/SELECT=RDO.EXE -
_$ SYS$SYSROOT:[SYSEXE]RDO.EXE
```

To restore RDO.EXE from the interactive kit, enter the following command:

```
$ BACKUP <device>:RDBVMSINT040.E/SAVE/SELECT=INTRDO.EXE *.*
```

To restore RDO.EXE from the run-time only kit, enter the following command:

```
$ BACKUP <device>:RDBVMSRTO040.B/SAVE/SELECT=RTORDO.EXE *.*
```

5.1.2 Back Up All Existing Rdb/VMS Databases

As a precaution, Digital recommends that you back up any Rdb/VMS databases, including DECtrace and CDD/Plus databases, with the RMU/BACKUP command before installing the MUP.

Before installing a new version of Rdb/VMS, Digital recommends that you perform a full RMU/BACKUP of the DECtrace administration and history databases, including DECtrace-formatted databases produced with the FORMAT command. To back up the DECtrace administration database, use the following command:

```
$ RMU/BACKUP ERC$ADMIN_DB EPC$ADMIN_DB.RBF
```

To backup the DECtrace history database, use the following command:

```
$ RMU/BACKUP EPC$HISTORY_DB EPC$HISTORY_DB.RBF
```

5.1.3 Disk Space Required to Install the MUP

Installing the MUP requires a certain amount of available disk storage space during the installation. Once the MUP is installed, it takes as much room as your previous version of Rdb/VMS. Table 5–1 summarizes the storage requirements for Rdb/VMS.

Table 5–1 Disk Space Requirements

Rdb/VMS Kit	Blocks During Installation
Full development	24,000
Interactive	24,000
Run-time	24,000

To determine the number of available disk blocks on the current system disk, enter the following command at the DCL prompt:

```
$ SHOW DEVICE SYS$SYSDEVICE
```

5.1.4 Shut Down the Rdb/VMS Monitor

The installation procedure terminates if the Rdb/VMS monitor is running. Before starting the installation, ensure that there are no active Rdb/VMS users by shutting down the Rdb/VMS monitor.

Note

If DECtrace is installed on your system, you must turn DECtrace off *before* you attempt to shut down the Rdb/VMS monitor. Turn DECtrace off using the following command:

```
$ COLLECT STOP SYSTEM/ABORT
```

Alternatively, you could stop both DECtrace and the Rdb/VMS monitor using the RMU/MONITOR STOP/ABORT=DELPRC command.

Run the RMONSTOP.COM procedure from SYS$STARTUP to shut down the monitor on all nodes in a VAXcluster system. For example:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
SYSMAN> DO @SYS$STARTUP:RMONSTOP
SYSMAN> EXIT
```

If you want to stop the Rdb/VMS monitor on only one node, enter the following command on that node:

```
$ @SYS$STARTUP:RMONSTOP
```

5.1.5 Obtain VMS Privileges Required to Install the MUP

VMSINSTAL is located in SYS$UPDATE, which is a restricted directory. To install the MUP, you must use an account that has SETPRV privilege. As one of its first actions, the VMSINSTAL command procedure grants all privileges except BYPASS to the process that invokes it. The VMSINSTAL command succeeds only if the account has SETPRV privilege.

To check the default privileges of the installing account, log in and enter this DCL command:

```
$ SHOW PROCESS/PRIVILEGES
```

If the account lacks the SETPRV privilege, you cannot install the MUP. You have two options:

- Ask your system manager to use AUTHORIZE to modify the default privileges of the account to include the SETPRV privilege.
- Run AUTHORIZE and make the changes yourself, if your account has the SYSPRV privilege:

  ```
  $ SET DEFAULT SYS$SYSTEM
  $ RUN AUTHORIZE
  UAF> MODIFY account-name/PRIVILEGES=(SETPRV)
  UAF> EXIT
  ```

  To activate the change in privileges, you must log out and log in again.

(Note that the VMSINSTAL procedure turns off the BYPASS privilege at the start of the installation.)

5.1.6 Ensure Sufficient Process Account Quotas to Install the MUP

The account you use to install the MUP must have sufficient quotas to enable you to perform the installation. Table 5–2 summarizes the minimum process quotas required to install the MUP.

Table 5–2 Process Account Quotas for the Installing Account

Account Quota	Value
ASTLM	24
BIOLM	18
BYTLM	20,480
DIOLM	18

(continued on next page)

Table 5–2 (Cont.) Process Account Quotas for the Installing Account

Account Quota	Value
ENQLM	2000
FILLM	50
PGFLQUO	20,000

User account quotas are stored in the file SYSUAF.DAT. You use AUTHORIZE to verify and change user account quotas. First set your directory to SYS$SYSTEM and then run AUTHORIZE:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF>
```

At the AUTHORIZE prompt (UAF>), use the SHOW command with an account name to check a particular account. For example, to check the SYSTEM account enter:

```
UAF> SHOW SYSTEM
```

To change a quota, use the MODIFY command at the UAF> prompt. The MODIFY command has the following syntax:

MODIFY account-name /quota-name=NNN

The following example changes the FILLM quota for the SYSTEM account and then exits from AUTHORIZE:

```
UAF> MODIFY SYSTEM /FILLM=50
UAF> EXIT
```

After you exit from the utility, the VMS system displays messages that indicate whether or not changes were made. Once the changes have been made, you must log out and log in again for the new quotas to take effect.

For more information on modifying account quotas, see the description of AUTHORIZE in the VMS documentation set.

5.1.7 Obtain System Parameter Values Required to Install the MUP

Installing the MUP requires certain system parameter settings. Table 5–3 lists the minimum required system parameter values for the installation. Depending on the kinds of programs and applications running at your site, you might need higher values for some settings.

Table 5–3 Required Minimum System Parameter Values

System Parameter	Value
CHANNELCNT	A number larger than the largest FILLM used on the system
CLISYMTBL[1]	250 pages

[1]The CLISYMTBL dynamic system parameter must be set to a minimum value of 250 pages during the installation procedure. If the current CLISYMTBL setting is less than 250 pages, you can lower the setting to its original value once the installation is finished.

(continued on next page)

Table 5–3 (Cont.) Required Minimum System Parameter Values

System Parameter	Value
GBLPAGES[2]	2078 available pages
GBLSECTIONS[2]	80 available sections
LOCKIDTBL	256 entries
LOCKIDTBL_MAX[3]	2048 entries
MAXBUF	2048 bytes
PROCSECTCNT	32 sections
RESHASHTBL	512 entries
SRPCOUNT	1024 packets
SRPCOUNTV	2048 packets
VIRTUALPAGECNT	20,000 (a number larger than largest PGFLQUOTA used on the system)

[2]For systems where you are performing a reinstallation, this number is the current value of GBLSECTIONS or GBLPAGES when the RMONSTOP command file or the RMU/MONITOR STOP command has been executed.

[3]This dynamic system parameter must be set permanently to a value equal to or greater than the value listed. Do not lower this value after the installation.

Section 5.1.7.1 through Section 5.1.7.3 show you how to check system parameter values, calculate values for the GBLPAGES and GBLSECTIONS system parameters, and change parameter values with the VMS AUTOGEN command procedure. Section 5.1.7.4 shows you how to use SYSGEN to change the values for dynamic system parameters.

5.1.7.1 Checking System Parameter Values

To check the values of your system parameters, enter the following command at the DCL prompt to invoke the VMS System Generation utility (SYSGEN):

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>
```

At the SYSGEN prompt (SYSGEN>), enter the SHOW command to display the value of a system parameter. The values displayed should equal or exceed the value of each parameter listed in Table 5–3. The following command displays the value for the LOCKIDTBL_MAX system parameter:

```
SYSGEN> SHOW LOCKIDTBL_MAX
Parameter Name    Current   Default   Minimum   Maximum Unit  Dynamic
--------------    -------   -------   -------   ------- ----  -------
LOCKIDTBL_MAX     65535     65535       200     65535 Entries     D
```

After you finish checking the parameters with the SHOW command, you can enter the EXIT command at the SYSGEN prompt to return to DCL.

5.1.7.2 Calculating the Values for GBLPAGES and GBLSECTIONS

To install and run the MUP, you must set the correct values for the GBLPAGES and GBLSECTIONS system parameters.

The 2078 value for GBLPAGES and the 80 value for GBLSECTIONS in Table 5–3 indicate that you must have at least 2078 unused pages and 80 unused sections available on your system for the installation to proceed successfully.

To see how many unused global pages and global sections your system has, enter the following DCL commands:

```
$ WRITE SYS$OUTPUT F$GETSYI ("FREE_GBLPAGES")
8900
$ WRITE SYS$OUTPUT F$GETSYI ("FREE_GBLSECTS")
90
```

Section 5.1.7.3 describes the procedures for increasing these values as well as those of other system parameters. Refer to the VMS documentation on system management and operations for more information.

5.1.7.3 Changing System Parameter Values with AUTOGEN

You use the AUTOGEN command procedure to change system parameters. The AUTOGEN procedure automatically adjusts values for parameters that are associated with the ones you set manually. To change system parameters with AUTOGEN, you must edit the SYS$SYSTEM:MODPARAMS.DAT file.

Use an editor to access the file. If you need to change a parameter value that is already in the SYS$SYSTEM:MODPARAMS.DAT file, delete the current value associated with that parameter and enter the new value.

To add a new value, add a line to the MODPARAMS.DAT file. The line contains the name of the parameter and its value. For example:

```
LOCKIDTBL_MAX = 2048
```

You can also modify incremental parameters in the MODPARAMS.DAT file. The following example increases the global page setting by 2000:

```
ADD_GBLPAGES = 2000
```

After you have made all your changes, run the AUTOGEN procedure to recalculate your system parameters. Enter the following command at the DCL prompt:

```
$ @SYS$UPDATE:AUTOGEN GETDATA REBOOT
```

AUTOGEN automatically adjusts some of the SYSGEN parameters based on the consumption of resources since the last reboot. If you do not want to take advantage of this automatic adjustment, include the NOFEEDBACK parameter at the end of the AUTOGEN command line. The AUTOGEN procedure performs an automatic system shutdown and reboots when it has finished. Rebooting your system makes the new parameter values active.

For more information about using AUTOGEN, see the instructions on modifying system parameters in the VMS documentation on system management and operations.

5.1.7.4 Setting Dynamic System Parameters

You can use SYSGEN to change the values for dynamic system parameters. The following example demonstrates this process for the CLISYMTBL system parameter. (After the installation is complete, you can reset CLISYMTBL to its previous setting or let it be reset automatically when you reboot your system.)

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> USE ACTIVE
SYSGEN> SET CLISYMTBL 250
SYSGEN> WRITE ACTIVE
SYSGEN> EXIT
```

Dynamic parameters changed with the SYSGEN WRITE ACTIVE command become active immediately without any need to reboot your system. In fact, rebooting returns dynamic system parameter values to their previous settings. Once you set values for dynamic parameters, you should complete the installation before rebooting the system.

The values for other dynamic parameters, such as LOCKIDTBL_MAX, must remain at the same level or higher than the values specified in Table 5–3.

5.1.8 Back Up Your System Disk

At the beginning of the installation, the VMSINSTAL command procedure asks if you have backed up your system disk. Digital recommends that you back up your system disk before installing any software on top of the operating system.

This precaution protects your system software. A system failure at a critical point in the installation procedure could leave unusable files. You also protect an existing version of the product, which may, if you request it, be deleted during the installation.

Use the backup procedures that have been established at your site. For details on backing up the system disk, see the section on the VMS Backup utility in the VMS documentation set.

5.1.9 Avoid Giving Users Access to HELP

When the installation inserts the Rdb/VMS Help modules into the VMS Help library, it must have sole access to the VMS Help library. If anyone uses the HELP command when the installation tries to insert the Rdb/VMS Help module, the installation fails. You can prevent other users from using the help library during the installation by either of the following methods:

- Running the installation when no one else is logged in
- Limiting access to the help library SYS$HELP:HELPLIB.HLB to the SYSTEM account:

```
$ SET PROTECTION = (S:RWED, O, G, W) SYS$HELP:HELPLIB.HLB
```

Remember to note the original protection on the library. After the installation, return the protection on the help library to the original setting. Instructions are provided in Section 5.3.3.

5.1.10 Prevent Interactive Users from Gaining Access to the System

If the installation fails for an indeterminable reason, Digital recommends that you install the MUP again, keeping all interactive users off the system during the installation procedure. You might also choose to keep interactive users off the system if you will be changing any system parameter values with the AUTOGEN command procedure. Use the DCL REPLY command to inform users of the schedule for the installation. Prevent other users from logging in by issuing the DCL SET LOGIN command:

```
$ REPLY/USER "Installation of Rdb/VMS starting in 5 minutes. Please
log out."
$ SET LOGIN/INTERACTIVE=0
```

Both of these commands require the OPER privilege.

If any batch or device jobs are running, you have two options:

- Wait until the last one finishes.
- Use the DCL DELETE/ENTRY command to stop any job still running.

5.2 Installing the Mandatory Update Package

This section describes how to install the MUP.

5.2.1 Time Required to Install the MUP

The installation of V4.0A takes approximately 25 minutes on a VAX 8800 system. This time may vary depending on your type of media, your system configuration, whether or not CDD/Plus is installed, and whether or not you need to reboot your system.

5.2.2 Invoking VMSINSTAL

To start the installation, invoke the VMSINSTAL command procedure from a privileged account, such as the SYSTEM account. The VMSINSTAL procedure is in the SYS$UPDATE directory. You use the following syntax to invoke VMSINSTAL:

```
@SYS$UPDATE:VMSINSTAL product-name device-name OPTIONS N
```

The rest of this section presents the parameters for the VMSINSTAL command line.

product-name
The installation name for the product. For the MUP, provide this parameter as follows:

RDBVMS_MUPA040

device-name
The name of the device on which you plan to mount the media. For example, MTA0: and MUA0: are device names for tape drives. It is not necessary to use the console drive for this installation. However, if you do use the console drive, you should replace any media you removed once the installation is complete.

OPTIONS N
An optional parameter that indicates you want to review the release notes question. If you include the OPTIONS N parameter, VMSINSTAL displays a menu that lets you choose between printing the release notes or displaying them on your terminal. You should always review the release notes before proceeding in case they contain new information about the installation. If you do not include the OPTIONS N parameter, VMSINSTAL does not ask you about the release notes. However, the release notes are automatically copied to SYS$HELP.

Note that there are several other options you can select when you invoke VMSINSTAL. See the VMS documentation on software installation for information on these options.

The following example displays the command to invoke VMSINSTAL to install Rdb/VMS from tape drive MTA0: and shows the system response. This example uses the OPTIONS N release note parameter.

```
$ @SYS$UPDATE:VMSINSTAL RDBVMS_MUPA040 MTA0: OPTIONS N

      VAX/VMS Software Product Installation Procedure V5.3


It is 14-MAR-1991 at 14::00.
Enter a question mark (?) at any time for help.
```

If you do not supply either the product name or the device name, VMSINSTAL prompts you for this information later in the installation procedure.

5.2.3 Steps of the Installation Procedure

This section discusses the installation process itself, presenting all the questions that appear during the installation.

Each question in the installation is marked with an asterisk (*) at the beginning of the line. Some questions show the default response in brackets, for example, [YES]. If you want to use the default response, press the Return key.

1. System backup

   The VMSINSTAL procedure asks if you are satisfied with your system backup. You should always back up your system disk before performing an installation. If you are satisfied with the backup of your system disk, press the Return key. Otherwise, enter NO to discontinue the installation. After you back up your system disk, you can start the installation again.

   ```
   * Are you satisfied with the backup of your system disk [YES]?
   ```

2. Mounting the media

   You should now mount the first distribution volume on the device you specified when you invoked VMSINSTAL. The VMSINSTAL procedure then asks you if you are ready to continue with the installation.

   If you respond YES to indicate that you are ready, VMSINSTAL displays a message that the media containing Rdb/VMS has been mounted on the specified device and that the installation has begun. For example:

   ```
   * Where will the distribution volumes be mounted: MTA0:

   Enter the products to be processed from the first distribution volume set.
   * Products: RDBVMS_MUPA040
   * Options: N
   The following products will be processed:
     RDBVMS_MUPA V4.0

       Beginning installation of RDBVMS_MUPA V4.0 at 14:00

   %VMSINSTAL-I-RESTORE, Restoring product saveset A...
   ```

   If you entered the wrong device name when you invoked VMSINSTAL and need to start the installation again, enter NO when asked if you are ready to install.

3. Release notes

   If you specified OPTIONS N when you invoked VMSINSTAL, you are now asked to choose one of the four options for reviewing the release notes.

```
    Additional Release Notes Options:

        1.  Display release notes
        2.  Print release notes
        3.  Both 1 and 2
        4.  None of the above

* Select option [2]: 2
```

The release notes are automatically copied to SYS$HELP no matter which option you choose, and whether or not you specified OPTIONS N.

The release notes are long; you might wish to print them by selecting option 2.

4. Continuing the installation

   The installation procedure now asks if you want to continue the installation. To continue, enter YES. Otherwise, press the Return key. In either case, the release notes, entitled *VAX Rdb/VMS Mandatory Update for Versions 3.1B and 4.0*, are copied to a file in the SYS$HELP directory. For example:

```
* Do you want to continue the installation [N]?: YES
%VMSINSTAL-I-RELMOVED, The product's release notes have been
moved to SYS$HELP.
```

   The release notes are located in the following file:

```
SYS$HELP:RDBVMS_MUPA040.RELEASE_NOTES
```

Note

The name of the release notes file installed by VMSINSTAL consists of the current product name and version number. Digital recommends that you keep the release notes for previous versions of Rdb/VMS.

   When you continue the installation, the following message is displayed:

```
    Installation procedures for: "VAX RDB/VMS V4.0A"


Be sure you have read the section on pre-installation steps in the
installation guide before continuing with the installation.


Checking system requirements ...
```

5. Choosing to run the Installation Verification Procedure (IVP)

   The Installation Verification Procedure (IVP) for the MUP verifies the installation. The installation asks if you want to run the IVP as part of the installation. If you respond YES, VMSINSTAL runs the IVP following the installation. It is recommended that you run the IVP to be sure that the MUP is installed correctly.

```
* Do you want to run the IVP after the installation [YES]?
```

   As part of the IVP, the MUP creates the personnel sample database in the directory specified by the logical RDM$DEMO.

   After the MUP is installed, you can run the IVP independently to verify that the software is available on your system. You might also want to run the IVP after a system failure to be sure that users can access Rdb/VMS. Online help contains instructions for running the IVP independently. More information follows in Section 5.3.2.

6. Choosing to purge files

   You have the option to purge files from previous versions of Rdb/VMS that are superseded by this installation. Purging is recommended; however, if you need to keep files from the previous version, enter NO in response to the question.

```
* Do you want to purge files replaced by this installation [YES]?
```

7. Informational messages

   At this point, the installation procedure displays a number of informational messages that report on the progress of the installation. There are no further questions. If the installation procedure has been successful up to this point, VMSINSTAL moves the new or modified files to their target directories, updates help files, and updates DCL tables, if necessary. If you asked for files to be purged, that work is done now. The following messages are displayed:

```
    There are no more questions.  The installation takes approximately
    20 minutes on a stand-alone VAX 8800.


    Beginning installation ...


    Installing under VMS V5.3     - 14-MAR-1991 14:10


%VMSINSTAL-I-RESTORE, Restoring product saveset C ...
%VMSINSTAL-I-RESTORE, Restoring product saveset D ...
%VMSINSTAL-I-RESTORE, Restoring product saveset E ...
%VMSINSTAL-I-RESTORE, Restoring product saveset G ...

   .
   .
   .

%VMSINSTAL-I-MOVEFILES, files will now be moved to their target
directories . . .
```

8. Running the IVP

   If you chose to run the IVP, VMSINSTAL runs it now. When the IVP runs successfully, you see the following display:

```
************************************

VAX Rdb/VMS V4.0A

Development

IVP COMPLETED SUCCESSFULLY

************************************
IVP completed for: VAX Rdb/VMS V4.0A
```

5.2.4 Completing the Installation Procedure

The following messages indicate that the entire installation procedure is complete:

```
Installation of RDBVMS_MUPA V4.0 completed at 14:45

VMSINSTAL procedure done at 14:45
```

You can now log out of the privileged account:

```
$ LOGOUT
SYSTEM     logged out at 14-MAR-1991 14:45:00.0
```

Note that VMSINSTAL deletes or changes entries in the process symbol tables during the installation. Therefore, if you are going to continue using the system manager's account and you want to restore these symbols, you should log out and log in again.

5.2.5 Errors That Cause the Installation to Fail

If errors occur during the installation itself or when the IVP is running, VMSINSTAL displays failure messages. If the installation fails, you see the following message:

```
%VMSINSTAL-E-INSFAIL, The installation of RDBVMS_MUPA V4.0 has failed.
```

If the IVP fails, you see these messages:

```
The RDBVMS_MUPA V4.0 Installation Verification Procedure failed.

%VMSINSTAL-E-IVPFAIL, The IVP for RDBVMS_MUPA V4.0 has failed.
```

Errors can occur during the installation if any one of the following conditions exists:

- Incorrect Rdb/VMS version

  Unless you have V3.1B, V3.1C, V4.0, or V4.0A of Rdb/VMS installed, the installation will fail.

- Incorrect operating system version

  Unless you have the VMS Version 5.3 or higher operating system installed, the installation will fail.

- No RDO image available

  If you have deleted your RDO.EXE image, you must restore it from the saveset using the VMS BACKUP command. See Section 5.1.1 for more information.

- Insufficient privileges

  The account you use to install the MUP must have the SETPRV privilege.

- Insufficient disk space on system disk

  If the system disk does not have enough blocks available to install the MUP, purge or delete unnecessary files according to the policies of your site. When you have enough disk space, you are ready to continue the installation procedure.

  See Table 5–1 for disk space requirements.

- Insufficient system parameter values for successful installation

  You must have the necessary minimum settings for system parameters on the installing account. See Table 5–3 for more system parameter information.

- Insufficient quotas for successful installation

You must have the necessary minimum account quotas set. See Table 5–2 for process account quotas.

- VMS Help Library currently in use

  The installation must have sole access to the VMS Help library when it tries to insert the Rdb/VMS Help module into the library.

- CDD/Plus installed but not started up prior to MUP installation

  If CDD/Plus is installed on your system but not started up, the IVP will commonly fail in the COBOL precompiler tests. If this occurs, start up CDD/Plus and rerun the IVP. Use the following command to start up CDD/Plus:

  ```
  $ @SYS$STARTUP:CDDSTRTUP
  ```

5.3 After Installing the MUP

After installing the MUP, you need to perform the following tasks:

- Tailor your system.
- Return the system to original settings.

This section also explains how access the online release notes and how to run the Installation Verification Procedure (IVP) independently after the software has been installed.

5.3.1 Accessing the Online Release Notes

Once the MUP has been installed, the release notes, entitled *VAX Rdb/VMS Mandatory Update for Versions 3.1B and 4.0*, are located in SYS$HELP:RDBVMS_MUPA040.RELEASE_NOTES.

Online help also directs you to the release notes file. After the installation, you can enter the following command to find the location of the release notes:

```
$ HELP RDBVMS RELEASE_NOTES
```

5.3.2 Tailoring Your System

This section discusses steps you must take to tailor your system to run Rdb/VMS after installing the MUP.

1. If you had the RDBPRE problem with request handles, recompile your programs. This problem has been fixed in this release.
2. If you had the RDBinterpret problem with VARCHAR fields in V4.0, retest your programs. This problem has been fixed in this release.
3. If you had one of the following RDML problems, recompile your programs:
   - A CDD informational message caused RDML to abort compilation. This problem has been fixed in this release.
   - RDML was accessing the wrong field in the symbol table, causing RDML to incorrectly abort compilation and return the RDML-E-READ_ONLY message when you attempted to update read-only (COMPUTED BY) fields. This problem has been fixed in this release.
4. Reinstall SQL/Services APIs after installation of the MUP.

After the MUP is installed, you must reinstall any of the client Application Programming Interfaces (APIs) that you plan to use. Reinstallation makes available all changes made to the SQL/Services APIs for the MUP.

Refer to the Rdb/VMS Version 4.0 *VAX Rdb/VMS Installation Guide* for complete SQL/Services API installation instructions.

5. Run RMONSTART.COM manually or by means of the Installation Verification Procedure (IVP).

   If you chose not to run the IVP as part of the installation, you will have to run the RMONSTART.COM command procedure manually to start the Rdb/VMS monitor and perform other related tasks such as installing shareable images and defining necessary logical names. The RMONSTART.COM command procedure is located in SYS$STARTUP. (If you have edited RMONSTART.COM to define LNK$LIBRARY, you will have to run the IVP on the boot node as well as on the VAXcluster satellite nodes.)

   Simply running the system startup command procedure SYSTARTUP_V5.COM or the RMU/MONITOR START command does not perform all of the tasks that the RMONSTART.COM procedure does and that Rdb/VMS requires.

6. Obtain the list of files installed by Rdb/VMS.

   A file is written to your system that identifies all the Rdb/VMS files installed on your system. To obtain this list after the installation ends, print (DCL PRINT) or display (DCL TYPE) a copy of the following file:

   ```
   SYS$COMMON:[SYSMGR.VAXINFO$PRODUCTS]RDBVMS_MUPA040_FILES.DAT
   ```

7. Run the MUP IVP as a standalone procedure.

   The MUP IVP procedure can be run at any time after the successful installation of the MUP. For example, if Rdb/VMS does not appear to be running properly, you may want to verify that the correct MUP distribution kit files are present on your system.

   The account you use to run the IVP must have the TMPMBX and SYSPRV privileges. If the data dictionary is installed on the system, the account must also have BYPASS privilege or the CDD EXTEND privilege at the CDD$TOP dictionary directory. Also, the quotas for the account you use must be sufficient to run Rdb/VMS.

   To run the MUP IVP after installation:

   a. Set the default to the following directory:

      ```
      $ SET DEFAULT SYS$COMMON:[SYSTEST]
      ```

   b. The command you enter to invoke the IVP depends on whether or not you have installed the VAX Rdb/VMS full development, interactive, or run-time kit license option:

      ```
      $ @RDBIVP DEV   ! Executes full development kit IVP
      $ @RDBIVP INT   ! Executes interactive kit IVP
      $ @RDBIVP RTO   ! Executes run-time kit IVP
      ```

   The standalone IVP procedure runs in the same manner as the VMSINSTAL IVP procedure. If the IVP fails, it creates a log file, SYS$UPDATE:RDBIVP.LOG, of the failed portion of the test.

5.3.3 Returning the System to Original Settings

If you have set interactive logins to 0 or changed the protection on the help library, you must reverse these actions.

- To restore interactive logins, enter the following command:

```
$ SET LOGIN/INTERACTIVE=value
```

- To change the protection on the help library, enter the following commands:

```
$ SET DEFAULT SYS$HELP
$ SET PROTECTION=(S:RWED,O:RWED,G:RWED,W:RE) HELPLIB.HLB
```

- If the system parameter CLISYMTBL was less than 250 before the installation, you can now set it to the original setting.

6

Software Errors Fixed in the Mandatory Update for Rdb/VMS V4.0

The following sections describe problems with previous versions of the Rdb/VMS software that are fixed in the mandatory update for Version 4.0. These software problems no longer exist.

6.1 General Information

This section contains notes and problem descriptions of a general nature.

6.1.1 The VMS Sort Utility for VMS V5.1, V5.2, and V5.3 Caused Problems with Rdb/VMS Databases

When an SQL or RDO IMPORT statement was used to import a database that had a table with a placement via hashed index defined, database corruption resulted. If the table had an index with no duplicates allowed defined for it, an RDB-E-NO_DUP error message was returned. This problem was caused by the VMS Sort utility (SORT) and is similar to the problem described in Section 2.1.1. However, in this case, Rdb/VMS was using the Sort utility that was not part of Rdb/VMS to perform this IMPORT operation.

Rdb/VMS called the Sort utility to sort a file to process the placement via hashed index. However, the Sort utility returned empty (null) records. Rdb/VMS then attempted to store these records in the database. If a no duplicates index was defined for the table, then the RDB-E-NO_DUP error message was returned. Otherwise, corrupt records were stored in the database with no error returned to the user.

This Sort utility problem could also occur if you used the RMU/UNLOAD command, the DCL SORT command, and the RMU/LOAD/PLACE command to sort and reload your database tables.

This problem is fixed in VMS Version 5.4. In addition, a VMSINSTAL kit is available from your Customer Support Center to fix the problem for Version 5.1, Version 5.2, and Version 5.3.

6.2 Software Errors Fixed That Apply to All Interfaces in the Mandatory Update for Version 4.0

This section contains notes and problems fixed in all interfaces.

6.2.1 Wrong RDBINTSHR.EXE Image Was Installed for Interactive License Customers

For V4.0, the wrong RDBINTSHR.EXE image was installed for interactive license customers.

This problem is fixed in the mandatory update for V4.0. The correct RDBINTSHR.EXE image now installs for interactive license customers.

6.2.2 Active Transactions in Application Programs Could Not Recover from Network Failures

If a network failure occurred during a transaction, an RDB$_IO_ERROR error was returned and the program running was not able to do any further Rdb/VMS processing.

This problem is fixed in the mandatory update for V4.0. The program can now finish and detach from the database. If a network failure occurs or there are other errors, such as constraint errors, an error message is returned indicating why the call failed. This enables applications to continue without exiting by trapping for the RDB$_IO_ERROR and taking appropriate actions, such as finishing all attaches and reattaching later.

6.2.3 Using Event Flags Caused Conflicts with Other Software Products

Using event flags caused conflicts with other software products.

This problem is fixed in the mandatory update for V4.0. Rdb/VMS now uses the LIB$GET_EF routine to allocate event flags to avoid conflicts with other software products. Rdb/VMS now reserves the following event flags:

- Common use
- Stall/timer use
- Remote use

An error message will be returned if there are not enough event flags.

6.2.4 An Access Violation Resulted When DECdtm Services and DECnet Services Were Not Running

If the SYSGEN parameter SCSNODE was not defined on the system, then any SQL, callable SQL, RDO, callable RDO, or RMU commands failed with a SYSTEM-F-ACCVIO error message. This problem existed because DECdtm services uses the SYSGEN parameter SCSNODE as the DECnet services name of the node and requires that it be defined.

A workaround is to define SCSNODE on machines where DECnet services is not running, when VAX Rdb/VMS is being used. SCSNODE is a nondynamic SYSGEN parameter. The recommended way of defining a value for SCSNODE is by editing the file SYS$SYSTEM:MODPARAMS.DAT and modifying the following line so that the spaces are replaced by a meaningful node name.

```
SCSNODE = "        "
```

VMS places the restriction that a node name must have between 1 and 6 alphanumeric characters, and must start with an alphabetic character. For example, the following node name would work for defining SCSNODE for the node named TEST1:

```
SCSNODE = "TEST1"
```

The following line may appear multiple times in the file MODPARAMS.DAT.

```
SCSNODE + "        "
```

Comment out the extra lines (using the ! character) so that you have just one line defining SCSNODE.

Then run AUTOGEN and reboot the machine. To do this in one step, use the following DCL command

```
$ @SYS$UPDATE:AUTOGEN SAVPARAMS REBOOT NOFEEDBACK
```

Because this DCL command updates the SYSGEN parameter SCSNODE and reboots the machine, you should issue it only when a reboot is convenient for users. It is recommended that you issue this command from the SYSTEM account.

This problem is fixed in the mandatory update for V4.0.

6.2.5 If a Commit Failed During a One-Phase Commit Protocol When an Explicit Distributed Transaction Was Run, It Caused a Premature $FINISH_RMOP to DECdtm

If you ran an explicit distributed transaction and DECdtm delivered a one-phase commit message to Rdb/VMS on commit transaction and the commit operation failed, it caused a premature $FINISH_RMOP to DECdtm.

This problem is fixed in the mandatory update for V4.0.

6.2.6 The Rdb/VMS DISTRIBTRAN Privilege Was Not Available for Remote Database Access

A remote attach to a database that does not support distributed transactions can still attempt a distributed transaction. This was a security problem because the remote server did not return the DISTRIBTRAN denied bit to the local process.

This problem is fixed in the mandatory update for V4.0.

6.2.7 A Partitioned Sorted Index Stored the First Record Incorrectly

A partitioned index allows you to specify limits for partitioning the index keys across multiple storage areas, as in the following example:

```
CREATE INDEX EMPLOYEES_INDEX
    ON EMPLOYEES(LAST_NAME)
    TYPE IS SORTED
    STORE USING (LAST_NAME)
    IN AREA_1 WITH LIMIT OF ("MILLER")
    OTHERWISE IN AREA_2;
```

If a table had a sorted partitioned index and at least one other index (partitioned or single area), the index keys for the first row stored in the table were stored in the wrong storage area. This problem sometimes led to an inability to select the first row stored in the table (if the index was used), or to a bugcheck. If you have a table with a partitioned index and at least one other index (partitioned or single area), you can work around this problem by dropping the indexes on the table and re-creating them.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 1. This mandatory update *will not* fix existing indexes with this problem. You must delete and redefine your indexes to repair them. New tables loaded after applying the mandatory update for V4.0 are not subject to this problem. However, tables

that were loaded prior to Version 4.0 still require the previous workaround to correct the problem.

6.2.8 A MODIFY Operation Caused Index Corruption on Partitioned Hash Indexes

Index corruption was possible with partitioned hash indexes on SQL UPDATE or RDO MODIFY operations where the index key changed. This corruption occurred only when the table had compression disabled.

This problem occurred if all of the following were true:

- A hashed index was defined on the table.
- The hashed index was partitioned over multiple storage areas.
- The data in the table was uncompressed.
- The operation was an SQL UPDATE or an RDO MODIFY operation.
- The hash key was altered as part of the update.

If all of these conditions were true, a hash key could be stored in the wrong storage area during the update operation. This could lead to the following as a result of this corruption:

- Deleting a record or changing the key of a record could result in the PSIHASH$DELETE + 00000051 bugcheck exception.
- Searching for a record by direct key lookup (using the hashed index) could result in no records returned when records should have been returned.

When the index key changed, Rdb/VMS put the hash bucket in the wrong partition. If you have this problem, you can correct your indexes by re-creating them. The problem could be worked around by eliminating any of the previously stated conditions. For instance, the table could be unloaded and reloaded with compression enabled.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 2. This mandatory update *will not* fix indexes that already exist with this problem. You must delete and redefine your indexes to repair them.

6.2.9 Partitioned Sorted Indexes Resulted in Various Problems

Any of the following problems was possible when you used partitioned sorted indexes:

- A bugcheck occurred with an exception at LCKCCH$LOCK_RET_NOT_ OK+15.
- "%RDMS-F-NOT_READY, storage area !AC not in ready mode" error message.
- During an IMPORT operation, a bugcheck dump occurred with the following two exceptions:
  - PIO$READY+23A, %RDMS-F-AREA_CORRUPT
  - LCKCCH$LOCK_RET_NOT_OK+15, %SYSTEM-F-ACCVIO
- Incorrect results were returned from queries.
- Inconsistent data or indexes were found as a result of the previous problem.

These problems generally occurred when one or more index partitions had no index records stored in it.

There is no workaround for these problems.

These problems are fixed in the mandatory update for V4.0 by RDMSHRP ECO 7. This mandatory update *will not* fix indexes that already exist with any of these problems. You must delete and redefine your indexes to repair them.

6.2.10 You Could Not Define Views Based on System Relations

In V4.0 the following restriction was added as documented in the *VAX Rdb/VMS Release Notes*:

```
3.11.3  Do Not Add Fields to Relations, Define Indexes, Triggers, and
        Other Database Objects Based on System Relation Fields

   You should not add fields to relations based on system relation
   fields. In addition, you should not define indexes, triggers, and
   other database objects on any system relation fields.
```

This restriction prevented the definition of views based on system relations. If you attempted to define a view based on a system relation, the following error message was returned:

```
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-NOMETSYSREL, metadata may not be created/altered on system relations
```

This restriction posed a problem for some 4GL products that require the ability to define views based on system relations.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 3. The restriction has been lifted to allow only the definition of views based on system relations. The restriction is still in effect for defining any other database object based on system relations.

6.2.11 An Incorrect Value Was Stored or a Bugcheck Resulted When Using BEFORE UPDATE or BEFORE MODIFY Triggers

The following circumstances could result in incorrect results stored in a database or a bugcheck dump with an exception at RDMS$$EXE_ACTION+4F7 when a table was updated in SQL or modified in RDO:

- An SQL BEFORE UPDATE or RDO BEFORE MODIFY trigger was defined on the table.
- The SQL BEFORE UPDATE or RDO BEFORE MODIFY trigger contained an update or modify action.
- The update or modify action on the trigger subject table required a scan of another table, or the update or modify action on the trigger subject table included an ordering clause such as a sort or project operation (SQL ORDER BY or SELECT DISTINCT or RDO SORTED BY or REDUCED TO clause).

This problem occurred as a result of the trigger operating with a record that was not the subject record of the update.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 4. The subject record is now ensured to be the current record before the trigger actions begin.

6.2.12 Queries with Computed Expressions and Indexes Returned the Wrong Results

Queries that used a computed expression where all fields of the expression were contained in an index returned incorrect results. This problem also occurred with queries with computed-by fields.

An example of a query with a computed expression is given in the example shown here. The R.F2|R.F3 is the computed expression and concatenates the fields F2 and F3.

The query in this example returned incorrect results because the code that computed the concatenation expression was missing from the Boolean code. This happened because the code was generated twice for the Boolean code that involved the concatenation expression.

```
$ RDO
DEFINE DATABASE MIKE DICTIONARY IS NOT USED.
DEFINE FIELD F1 DATA TEXT SIZE 3.
DEFINE FIELD F2 DATA TEXT SIZE 5.
DEFINE FIELD F3 DATA TEXT SIZE 3.
DEFINE FIELD F4 DATA TEXT SIZE 11.
DEFINE REL R1. F1. F2. F3. F4. END.
DEFINE INDEX I1 FOR R1 DUPLICATES ARE NOT ALLOWED. F1. F2. F3. END.
COMMIT
START-TRANSACTION READ-WRITE
STORE R IN R1 USING R.F1="AAA"; R.F2="AAAAA"; R.F3="AAA"; R.F4="A123" END-STORE
STORE R IN R1 USING R.F1="AAA"; R.F2="AAAAA"; R.F3="BAA"; R.F4="A123" END-STORE
STORE R IN R1 USING R.F1="AAA"; R.F2="CCAAA"; R.F3="CAA"; R.F4="A123" END-STORE
STORE R IN R1 USING R.F1="BAA"; R.F2="AAAAA"; R.F3="AAA"; R.F4="A123" END-STORE
STORE R IN R1 USING R.F1="BAA"; R.F2="CCAAA"; R.F3="BAA"; R.F4="A123" END-STORE
STORE R IN R1 USING R.F1="CAA"; R.F2="AAAAA"; R.F3="AAA"; R.F4="A123" END-STORE
STORE R IN R1 USING R.F1="CAA"; R.F2="CCAAA"; R.F3="BAA"; R.F4="A123" END-STORE
STORE R IN R1 USING R.F1="CAA"; R.F2="CCAAA"; R.F3="CAA"; R.F4="A123" END-STORE
COMMIT
! This query should return one record
FOR R IN R1 WITH R.F1="baa" AND R.F2|R.F3 BETWEEN "CCAAABAA" AND "CCAAACAA"
  SORTED BY R.F2,R.F3 PRINT R.F4,R.F2,R.F3 END-FOR
EXIT
```

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 5.

6.2.13 Queries with Computed Expressions Returned the Wrong Results

Queries that used a computed expression or computed-by fields returned incorrect results.

The query in this example returned incorrect results:

```
INVOKE DATA FILE PERSONNEL
START_TRANSACTION READ_WRITE
DELETE INDEX EMP_EMPLOYEE_ID.
DEFINE INDEX EMP_1 FOR EMPLOYEES. EMPLOYEE_ID. END.
DEFINE INDEX EMP_4 FOR EMPLOYEES. SEX. STATE. POSTAL_CODE. EMPLOYEE_ID. END.
FOR E IN EMPLOYEES WITH E.SEX="M" AND
  E.STATE|E.POSTAL_CODE="MA03442" AND E.EMPLOYEE_ID="00174"
    PRINT E.SEX,E.STATE,E.POSTAL_CODE,E.EMPLOYEE_ID,E.MIDDLE_INITIAL
END_FOR
```

This query returned nothing when it should have returned the following:

```
SEX   STATE   POSTAL_CODE   EMPLOYEE_ID   MIDDLE_INITIAL
M     MA      03442         00174         V
```

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 17.

6.2.14 Some Update-Intensive Applications Experienced a Performance Degradation in Rdb/VMS V4.0 Compared to V3.1B

Some update-intensive applications experienced a dramatic increase in I/O activity following an upgrade to Rdb/VMS V4.0. Applications that continually updated the same set of records usually experienced this problem. This increased I/O activity could severely degrade overall performance. If, after you upgraded to V4.0, the query optimizer strategy chosen by Rdb/VMS remained the same as it was under V3.1B, but I/O activity increased (causing overall performance degradation), then your application is experiencing this problem.

Performance is improved in the mandatory update for V4.0 by RDMSHRP ECO 6 and this problem is fixed in V4.1. See Section 7.9.1 for more information.

6.2.15 Poor Performance Was Experienced While Retrieving Views by Dbkey

Queries that retrieved views by dbkey performed poorly due to optimizer strategies that did not always retrieve data from relations by dbkey. This situation especially affected DATATRIEVE users who performed FIND collections on views and then processed them. The DATATRIEVE FIND collections operation builds dbkey lists. Further processing was done by asking Rdb/VMS to retrieve the information using the dbkey lists.

Consider the following RDBPRE BASIC program. This program gets the dbkey for the first record in the CURRENT_JOB view and then saves the dbkey. The program then reads the record using the saved dbkey.

```
10      !
&rdb&   INVOKE DATABASE FILENAME "MF_PERSONNEL"
&rdb&   FOR FIRST 1 C IN CURRENT_JOB GET DBKEY$=C.RDB$DB_KEY END_GET END_FOR
        PRINT "**** RETRIEVING BY VIEW DBKEY ****"
&rdb&   FOR C IN CURRENT_JOB WITH C.RDB$DB_KEY=DBKEY$ GET A$=C.LAST_NAME
&rdb&     END_GET END_FOR
        END
```

If you run this program with the RDMS$DEBUG_FLAGS logical name defined as ''S'', the following output displays:

```
   .
   .
   .
**** Retrieving by view dbkey ****
Cross block of 2 entries
  Cross block entry 1
    Get     Retrieval by index of relation EMPLOYEES
      Index name  EMP_LAST_NAME [0:0]
  Cross block entry 2
    Conjunct        Conjunct        Firstn  Get
    Retrieval by DBK of relation JOB_HISTORY
```

As noted here, the optimizer chose retrieval by index on EMPLOYEES instead of retrieval by dbkey. If this was the retrieval on the first relation of a view that scanned over 16,000 records using sequential retrieval for every dbkey in a dbkey list, performance would be poor.

The proper strategy for the view retrieval by dbkey for this same program is as follows:

```
   .
   .
   .
**** Retrieving by view dbkey ****
Cross block of 2 entries
  Cross block entry 1
    Conjunct      Firstn  Get  Retrieval by DBK of relation JOB_HISTORY
  Cross block entry 2
    Conjunct      Conjunct     Firstn  Get
    Retrieval by DBK of relation EMPLOYEES
```

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 9.

6.2.16 Wrong Results Were Returned from Queries That Used Collating Sequences and the STARTING WITH "" Relational Operator

Queries that used the STARTING WITH "" relational operator and collating sequences returned incorrect results. The following example reproduces the problem:

```
$ RDO
DEFINE DATABASE MIKE DICTIONARY IS NOT USED.
DEFINE COLLATING_SEQUENCE DEUTSCH GERMAN.
DEFINE FIELD F1 DATATYPE IS TEXT SIZE IS 5 COLLATING_SEQUENCE IS DEUTSCH.
DEFINE RELATION R1. F1. END RELATION.
STORE R IN R1 USING R.F1 ="BBB" END_STORE
STORE R IN R1 USING R.F1 ="ooo" END_STORE
COMMIT
START_TRANSACTION READ_WRITE
FOR R IN R1 WITH R.F1 STARTING WITH "" PRINT R.* END_FOR
DEFINE INDEX F1_IND FOR R1 DUPLICATES NOT ALLOWED TYPE IS SORTED. F1. END.
FOR R IN R1 WITH R.F1 STARTING WITH "" PRINT R.* END_FOR
ROLLBACK
```

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 10.

6.2.17 Recovery-Unit Journal (.RUJ) Files Could Not Be Created Using Angle Brackets (< >)

Recovery-unit journal (.RUJ) files could not be created in the default .RUJ directory (DISK:[RDM$RUJ]) when the current directory had been set using angle brackets (< >). An example of the error message output from this problem is shown here:

```
%RDB-F-SYS_REQUEST, error from system services request
-RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-FILACCERR, error creating recovery-unit journal file
    DISK4:<RDM$RUJ>MIKE$009452BD0BCF2B00.RUJ;
-RMS-E-PRV, insufficient privilege or file protection violation
```

This problem occurred when .RUJ files were created in the default .RUJ directory after a user had set the default directory using angle brackets. If the default .RUJ directory did not exist, then a user would receive a directory-not-found error. If the default .RUJ directory had been previously created by another user, then the current user would receive a privilege-violation error, as shown in the sample output.

Note that this problem was only relevant to sites that use angle brackets (< >) in the directory specification.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 11.

6.2.18 A Bugcheck Sometimes Resulted When a Sorted Index Rebalanced Itself

Occasionally, a sorted or B-tree index in the process of rebalancing itself caused a bugcheck with the following exception: PSIINDEX$JOIN_SCR + 9A.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 12.

6.2.19 NOWAIT Transactions Started During a Recovery Process Caused an RDMS-F-AREABUSY Fatal Error

The following messages could result after repeated attempts to start a NOWAIT transaction:

```
%RDB-E-LOCK_CONFLICT, request failed due to locked resource; no-wait
    parameter specified for transaction
-RDMS-F-AREABUSY, usage of storage area <rdb$system> conflicts with a
    co-user
```

This problem generally occurred after a recovery process had been started to roll back a terminated process. The user process that attempted to start the NOWAIT transaction gave up the freeze lock because of the recovery process. As part of the execution of the start transaction statement, Rdb/VMS attempted to access the RDB$SYSTEM storage area and in doing so, attempted to regain the freeze lock when the recovery was done. Rdb/VMS could not regain the freeze lock due to a problem, and so returned an error indicating that it could not access the RDB$SYSTEM storage area.

This problem can be reproduced on the PERSONNEL database by using four processes. Three processes are used to update the database, and the fourth process is used to stop the job. The function of each process and its time sequence is as follows:

```
(1) Process A: START_TRANSACTION READ_WRITE NOWAIT

(2)   Process B: START_TRANSACTION READ_WRITE NOWAIT

(3) Process A: STORE C IN CANDIDATES USING C.LAST_NAME="A" END_STORE
               COMMIT

(4)   Process B: STORE C IN CANDIDATES USING C.LAST_NAME="B" END_STORE

(5)     Process C: START_TRANSACTION READ_WRITE NOWAIT
                       STORE C IN CANDIDATES ...

(6)       Process D: STOP/PROC/ID=PROCESS C

    **** Wait for recovery to complete (pause for a minute) ****

(7) Process A: START_TRANSACTION READ_WRITE NOWAIT
```

The user process did not regain the freeze lock due to a problem in the Rdb/VMS software.

A workaround is to start a WAIT transaction or detach from the database with a FINISH statement.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 13.

6.2.20 The Query Optimizer Caused Various Bugchecks When Queries Were Run

The following bugcheck exceptions were observed from Rdb/VMS V4.0 queries:

```
PSIISCAN$GET_NEXT_UNIQUE + 0FF
RDMS$$EXE_LEAF_CLOSE + 36
PSIISCAN$GET_NEXT_DUPLICATE + 26
PSIISCAN$END_SCAN + 26
KOD$GET_VM + 16A
KOD$FREE_VMLIST + 37
PIOFETCH$WITHIN_DB + 142
```

This problem was caused by the query optimizer writing a bitmap into an incorrect data area. This occurred only when the same query (or the same leaf execution node) was executed more than once and when the index bitmap was created during the previous query execution.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 14.

6.2.21 Shared Write Queries Consumed More Memory Than Expected

Some queries consumed up to two times as much memory on Rdb/VMS V4.0 as the same query running on Rdb/VMS V3.1B.

This problem is fixed in the mandatory update for V4.0. Digital recommends that, when possible, you use a protected write transaction whenever the transaction modifies a large percentage of the rows in the table. This is because shared write transactions need more main memory to keep track of all the rows that have been accessed or updated by the transaction.

A protected write transaction, on the other hand, needs to keep track of only the rows that have been updated. This can substantially reduce memory requirements for these data structures.

6.2.22 Locking Protocol Problem Caused Bugchecks

An error in the locking protocols created the rare scenario that the recovery process and after-image journaling would interact incorrectly, causing errors in writing to the .AIJ file and generating bugcheck dumps for either the user or the recovery process.

There are no workarounds to this problem. However, this problem did not result in any database corruption. When the bugcheck occurred in a user process, recovery completed normally. When the bugcheck occurred in the DBR process, the database was shut down to protect itself.

This problem is fixed in the mandatory update for V4.0.

6.2.23 Deleting and Then Creating a Logical Area and Accessing the Schema Caused a Page Checksum Bugcheck

When a logical area was deleted and another logical area was then created followed by a simple query of the database (SQL SELECT statement) to cause the schema to be read again, a page checksum bugcheck resulted. There was a problem that corrupted some entries on an AIP page in the database.

This problem is fixed in the mandatory update for V4.0.

6.2.24 Rdb/VMS Behavior Had Changed so That Buffers Were Emptied on Rollback

For V4.0, a ROLLBACK statement caused all buffers to be emptied. This meant that the next transaction would reread the same buffer again, resulting in unnecessary I/O activity. This was especially bad for VAX RALLY and other applications that use rollback for read-only transactions. A workaround is to commit the read-only transactions.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 19. The behavior is now the same as it was in versions previous to V4.0.

6.2.25 Lock-Related Looping Problem

In some cases, the database application went into an infinite loop or a bugcheck resulted while processing some lock information. The infinite loop condition consumed all of the CPU.

This problem is fixed in the mandatory update for V4.0.

6.2.26 Problem with SPAM Thresholds in a Recover Operation

SPAM thresholds that were not properly updated in a recover operation caused performance problems.

This problem is fixed in the mandatory update for V4.0.

6.2.27 Query Returned Records in Wrong Order with the SQL ORDER BY <column-name> DESCENDING or the RDO SORTED BY DESCENDING <field> Clauses

Queries using the SQL ORDER BY <column-name> DESCENDING clause or the RDO SORTED BY DESCENDING <field> clause returned rows in incorrect order. Consider the following SQL defined database and query that reproduces this problem:

```
CREATE SCHEMA FILENAME MIKE;
CREATE TABLE T1 (F1 CHAR(2),F2 CHAR(3),F3 DATE,F4 SMALLINT,F5 INTEGER);
CREATE TABLE T2 (F1 CHAR(2),F2 CHAR(3),F3 DATE,F4 SMALLINT,F6 DATE);
COMMIT;
CREATE UNIQUE INDEX T1_I ON T1 (F3,F5,F2,F1,F4);
CREATE UNIQUE INDEX T2_I ON T2 (F3,F4,F2,F1);
COMMIT;
INSERT INTO T1 VALUES('NW','VIC','1-OCT-1990',1,80531);
INSERT INTO T1 VALUES('NW','VIC','29-OCT-1990',1,80531);
INSERT INTO T2 VALUES('NW','VIC','1-OCT-1990',1,'1-JAN-1990');
INSERT INTO T2 VALUES('NW','VIC','29-OCT-1990',1,'1-JAN-1990');
COMMIT;
SELECT P.F3, P.F6 FROM T2 P, T1 I
        WHERE I.F1="NW" AND I.F2="VIC" AND I.F5=80531 AND P.F1="NW" AND
              P.F2="VIC" AND I.F4=P.F4 AND I.F3=P.F3 AND P.F3 <= "29-OCT-1990"
        ORDER BY P.F3 DESC;
```

This query returned P.F3 not sorted in descending order:

```
 P.F3                      P.F6
  1-OCT-1990 00:00:00.00    1-JAN-1990 00:00:00.00
 29-OCT-1990 00:00:00.00    1-JAN-1990 00:00:00.00
2 rows selected
```

The query should have returned P.F3 sorted in descending order:

```
 P.F3                        P.F6
 29-OCT-1990 00:00:00.00     1-JAN-1990 00:00:00.00
  1-OCT-1990 00:00:00.00     1-JAN-1990 00:00:00.00
2 rows selected
```

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 18.

6.2.28 The Predicate CONTAIN Uppercased the Second Byte of Some Two-Octet Characters Incorrectly

Users who used text strings in two-octet character sets, that is, KANJI, HANZI, HANYU or HANGUL have to define the logical name RDB$CHARACTER_ SET. In this case, the predicate CONTAIN uppercased the second byte of some two-octet characters incorrectly. When the second byte was equivalent to the code of an ASCII lowercase character, the predicate uppercased it to a code of a corresponding ASCII uppercase character. The problem was that this uppercasing was not meaningful for the two-octet character sets.

Users who did not use any character sets besides ASCII or MCS were not affected.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 16. The predicate CONTAIN does not uppercase the second byte of a two-octet character.

6.2.29 Negate Operator Incorrectly Propagated the NULL Bit While Processing a Record Stream

In previous versions of Rdb/VMS, the negate operator incorrectly propagated the NULL bit while processing a record stream.

For instance, of the six values returned from a query, the third was NULL. If the negate operator was used, then the third and subsequent values were also returned as NULL.

In RDO, RDBPRE, and RDML the symptoms were zeros (the default MISSING_ VALUE) returned for the subsequent column values.

In V4.0, (also in V3.0 and V3.1) the following example demonstrates this problem:

```
SQL> CREATE DATABASE FILENAME FOO;
SQL> CREATE TABLE T(A INTEGER, B INTEGER);
SQL> INSERT INTO T VALUES (1, 10);
1 row inserted
SQL> INSERT INTO T VALUES (2, 10);
1 row inserted
SQL> INSERT INTO T VALUES (3, NULL);
1 row inserted
SQL> INSERT INTO T VALUES (4, 10);
1 row inserted
SQL> INSERT INTO T VALUES (5, 10);
1 row inserted
SQL> INSERT INTO T VALUES (6, 10);
1 row inserted
SQL> COMMIT;
```

```
SQL> SELECT A, B FROM T;
           A             B
           1            10
           2            10
           3          NULL
           4            10
           5            10
           6            10
6 rows selected
SQL> SELECT A, -B FROM T;
           A
           1           -10
           2           -10
           3          NULL
           4          NULL   <-- should be -10
           5          NULL   <-- should be -10
           6          NULL   <-- should be -10
6 rows selected
SQL> COMMIT;
```

A workaround is to subtract the value from zero, instead of using the negate operator.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 23. The following query shows the correct results:

```
SQL> SELECT A, 0 - B FROM T;
           A
           1                    -10
           2                    -10
           3                   NULL
           4                    -10
           5                    -10
           6                    -10
6 rows selected
```

6.2.30 An UPDATE Operation Stored Incorrect Results

Updates to tables where one or more of the columns updated is contained in a sorted index for the table caused incorrect results to be stored in the table. This is due to rows being updated multiple times when they should have been updated only once.

The following SQL example shows this problem:

```
DECLARE SCHEMA FILENAME PERSONNEL;
CREATE INDEX TIDX ON SALARY_HISTORY (EMPLOYEE_ID,SALARY_AMOUNT);
SELECT EMPLOYEE_ID, SALARY_AMOUNT, DBKEY FROM SALARY_HISTORY
  WHERE EMPLOYEE_ID BETWEEN '00201' AND '00202' ORDER BY DBKEY;

    EMPLOYEE_ID   SALARY_AMOUNT                  DBKEY
    00201            $15,897.00               41:701:4
    00201            $21,259.00               41:701:5
   .
   .
   .
    18 rows selected

UPDATE SALARY_HISTORY SET SALARY_AMOUNT = SALARY_AMOUNT + 500 WHERE
  EMPLOYEE_ID BETWEEN '00201' AND '00202';
    69 rows updated     ! This number will probably be 23 on your system.

SELECT EMPLOYEE_ID, SALARY_AMOUNT, DBKEY FROM SALARY_HISTORY
  WHERE EMPLOYEE_ID BETWEEN '00201' AND '00202' ORDER BY DBKEY;
```

```
    EMPLOYEE_ID   SALARY_AMOUNT                  DBKEY
    00201            $20,397.00                41:701:4
    00201            $21,759.00                41:701:5
  .
  .
  .
   18 rows selected
```

As shown, the first SALARY_HISTORY row has been updated nine times with the SALARY_AMOUNT = SALARY_AMOUNT + 500 statement.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 21.

6.2.31 An UPDATE Operation Caused a Bugcheck

Updates to a table where two or more indexes are used to retrieve rows to be updated caused a bugcheck. This occurred after the RDMSHRP ECO 21 patch was applied, which caused multiple temporary tables to be created as part of the query strategy.

The following SQL example shows the problem:

```
DECLARE SCHEMA FILENAME PERSONNEL;

UPDATE DEGREES SET DEGREE = NULL
WHERE EMPLOYEE_ID = '00200' OR
      COLLEGE_CODE = 'MIT';
17 rows updated
```

Note that the problem only occurs if patch RDMSHRP ECO 21 has been applied. RDMSHRP ECO 29, along with RDMSHRP ECO 21, corrects the original problem with table updates. See Section 6.2.30 for more information about RDMSHRP ECO 21.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 29.

6.2.32 Query Using Descending Indexes Returned Incorrect Results

When a query was performed using descending indexes, the query returned incorrect results. The following example shows the problem:

```
CREATE INDEX J_IDX ON JOB_HISTORY (EMPLOYEE_ID DESC);
CREATE INDEX E_IDX ON EMPLOYEES (EMPLOYEE_ID DESC);
!
CREATE VIEW V AS
        SELECT E.EMPLOYEE_ID FROM
                EMPLOYEES E, JOB_HISTORY J
                        WHERE E.EMPLOYEE_ID=J.EMPLOYEE_ID;
!
! The following query is fine.
!
SELECT * FROM V ORDER BY EMPLOYEE_ID;
!
! **** The following query returned incorrect results ****
!
SELECT * FROM V ORDER BY EMPLOYEE_ID DESC;
!
```

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 25.

6.2.33 Query with SQL LIKE Returned Incorrect Results

When a query was run using the SQL LIKE predicate, it returned inconsistent results. When the same query was run and the LIKE predicate was replaced with the BETWEEN predicate, consistent results were returned. The following example shows the inconsistent results when the query was run, rolled back, and repeated 10 times:

```
$ SQL
DECLARE SCHEMA FILENAME RDB_DATABASE;
    SELECT STK_CODE,BRIEF_DESC,RESTRICTED_STATUS
      FROM STK_RPT_STRUCTURE,STK_MASTER
      WHERE ITEM_CODE = STK_CODE
    AND
      ((PROD_WAREHOUSE_1 LIKE 'AA')
      OR (PROD_WAREHOUSE_2 LIKE 'AA')
      OR (PROD_WAREHOUSE_3 LIKE 'AA'))
    AND EXISTS
      (SELECT * FROM WAREHOUSE_STK WHERE STK_CODE = ITEM_CODE
         AND WAREHOUSE LIKE "__"
           AND (ON_HAND <> 0
           OR IN_TRANSIT <> 0
           OR ORDERS <> 0))
    ORDER BY RPT_SEQ;

97 rows selected
97 rows selected
97 rows selected
92 rows selected
97 rows selected
97 rows selected
97 rows selected
97 rows selected
97 rows selected
96 rows selected
```

A workaround is to run the same query and replace the LIKE predicate with the BETWEEN predicate as follows:

```
$ SQL
DECLARE SCHEMA FILENAME RDB_DATABASE;
   SELECT STK_CODE,BRIEF_DESC,RESTRICTED_STATUS
     FROM STK_RPT_STRUCTURE,STK_MASTER
     WHERE ITEM_CODE = STK_CODE
   AND
     ((PROD_WAREHOUSE_1 LIKE 'AA')
     OR (PROD_WAREHOUSE_2 LIKE 'AA')
     OR (PROD_WAREHOUSE_3 LIKE 'AA'))
   AND EXISTS
     (SELECT * FROM WAREHOUSE_STK WHERE STK_CODE = ITEM_CODE
        AND WAREHOUSE BETWEEN 'AA' AND 'ZZ'
          AND (ON_HAND <> 0
          OR IN_TRANSIT <> 0
          OR ORDERS <> 0))
   ORDER BY RPT_SEQ;
ROLLBACK;
EXIT;
```

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 24.

6.2.34 Query with Compressed Indexes Returned Incorrect Results

When a query was run that used a sorted index with a WITH LIMITS clause for compressing the index, and the query performed an equijoin operation, incorrect results were returned. The following example shows the problem:

```
CREATE SCHEMA FILENAME TEST
CREATE STORAGE AREA RDB$SYSTEM;
!
CREATE TABLE TABLE_A (I INTEGER);
CREATE TABLE TABLE_B (I_1 INTEGER, I_2 INTEGER);
!
! Initial test using uncompressed indexes
!
CREATE INDEX INDEX_A ON TABLE_A(I);
CREATE INDEX INDEX_B ON TABLE_B(I_1,I_2);
COMMIT;
INSERT INTO TABLE_A VALUES (1);
INSERT INTO TABLE_A VALUES (2);
INSERT INTO TABLE_A VALUES (3);
INSERT INTO TABLE_A VALUES (4);
INSERT INTO TABLE_A VALUES (5);
INSERT INTO TABLE_A VALUES (6);
INSERT INTO TABLE_A VALUES (7);
INSERT INTO TABLE_A VALUES (8);
INSERT INTO TABLE_A VALUES (9);
INSERT INTO TABLE_A VALUES (0);
COMMIT;
INSERT INTO TABLE_B VALUES (1,1);
INSERT INTO TABLE_B VALUES (1,2);
INSERT INTO TABLE_B VALUES (1,3);
INSERT INTO TABLE_B VALUES (1,4);
INSERT INTO TABLE_B VALUES (1,5);
INSERT INTO TABLE_B VALUES (2,1);
INSERT INTO TABLE_B VALUES (2,2);
INSERT INTO TABLE_B VALUES (2,3);
INSERT INTO TABLE_B VALUES (2,4);
INSERT INTO TABLE_B VALUES (2,5);
COMMIT;
!
! First SELECT should retrieve 10 rows and does
!
SELECT A.I, B.I_1, B.I_2 FROM TABLE_A A, TABLE_B B
WHERE A.I = B.I_2;
!
! Second SELECT should retrieve 2 rows and does
!
SELECT A.I, B.I_1, B.I_2 FROM TABLE_A A, TABLE_B B
WHERE A.I = B.I_2 AND B.I_1 = 1 AND B.I_2 BETWEEN 1 AND 2;
COMMIT;
!
! Drop indexes and replace them with compressed indexes
! (compression range, like row counts and table degrees, is vastly
! reduced for this example).
!
DROP INDEX INDEX_A;
DROP INDEX INDEX_B;
!
CREATE INDEX INDEX_A ON TABLE_A(I MAPPING VALUES 0 TO 750);
CREATE INDEX INDEX_B ON TABLE_B(I_1 MAPPING VALUES 0 TO 750,
    I_2 MAPPING VALUES 0 TO 750);
!
COMMIT;
!
! First SELECT continues to retrieve 10 rows
```

```
!
SELECT A.I, B.I_1, B.I_2 FROM TABLE_A A, TABLE_B B
WHERE A.I = B.I_2;
!
! Second SELECT should retrieve 10 rows, but does not;
! therefore, an incorrect result was returned.
!
SELECT A.I, B.I_1, B.I_2 FROM TABLE_A A, TABLE_B B
    WHERE A.I = B.I_2
        AND B.I_1 = 1 AND
        B.I_2 BETWEEN 1 AND 2;
```

A workaround retrieves the correct number of rows, but the access strategy is not optimal (MATCH with multiple SORT) and adversely affects the application performance by several orders of magnitude. The workaround is as follows:

```
SELECT A.I, B.I_1, B.I_2 FROM TABLE_A A, TABLE_B B
    WHERE A.I = B.I_2
        AND B.I_1 BETWEEN 1 AND 1
        AND B.I_2 BETWEEN 1 AND 2;
```

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 26.

6.2.35 Query Returned Incorrect Results

The following query returned incorrect results:

```
SELECT A.EMPLOYEE_ID,B.EMPLOYEE_ID FROM EMPLOYEES A,JOB_HISTORY B
WHERE A.EMPLOYEE_ID='00195' AND
  ((A.EMPLOYEE_ID=B.EMPLOYEE_ID AND
    A.STATUS_CODE='0') OR
   (A.EMPLOYEE_ID=B.EMPLOYEE_ID AND
    A.STATUS_CODE='1'));
```

A workaround is to rephrase the query as follows when this is possible:

```
SELECT A.EMPLOYEE_ID,B.EMPLOYEE_ID FROM EMPLOYEES A,JOB_HISTORY B
WHERE A.EMPLOYEE_ID='00195' AND
    A.EMPLOYEE_ID=B.EMPLOYEE_ID AND
   (A.STATUS_CODE='0' OR
    A.STATUS_CODE='1');
```

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 27.

6.2.36 Poor Performance Was Observed with Queries Using Dynamic OR Optimization Within the Leaf Retrieval

Queries that specify several ORed ranges on an index executed much more slowly even though the query cost was considerably less. For example, a query using two ranges based on a key with three segments using the following format (as described in Section 17.1.1 of the *VAX Rdb/VMS Guide to Database Maintenance and Performance*) showed this problem:

```
(S1 = X AND S2 = Y AND S3 = Z) OR (S1 = X AND S2 = Y1 AND S3 = Z1)
```

The problem was that the leaf bypassed the key skip mechanism to pass over a key gap to skip to the next range. As a result, keys were scanned sequentially from the begining of the first range, which was abandoned by the leaf and finally replaced by a sequential table scan.

A workaround is to rephrase the query as follows:

```
(S1 = X AND S2 = Y AND (S3 = Z OR S3 = Z1)
```

A second workaround is to add an ORDER BY S1, S2, S3 clause to the query. This should enable the use of a regular indexed retrieval instead of leaf retrieval and thus avoid the problem.

This problem is fixed for read-only transactions in the mandatory update for V4.0 by RDMSHRP ECO 28.

6.2.37 SPAM Pages Were Not Updated Correctly

SPAM pages showed space to store records when in fact pages were full. A full verification (RMU/VERIFY/ALL) of the database produced error messages indicating that the SPAM pages were not correctly updated.

A workaround is to use the RMU/REPAIR command to periodically correct the problem by rebuilding the SPAM pages so SPAM pages display the correct available space on the data pages.

This problem is fixed in the mandatory update for V4.0.

6.2.38 Global Section Was Corrupted When a User Had Multiple Attaches

When a process with multiple streams or attaches received a blocking AST while it was switching streams, the global section was corrupted. This could cause data corruption and other unpredictable results.

This problem is fixed in the mandatory update for V4.0.

6.2.39 Under Certain Circumstances a Committed Update Was Not Completely Written to the .AIJ File

Under certain circumstances with a specific combination of buffer flushing, a committed update to the database was not completely written to the .AIJ file. As a consequence, if an attempt was made to recover the database, the database was corrupted and a bugcheck resulted or the corruption was not detected.

For example, when rows were added or deleted and in the process of rebalancing an index node for a sorted index on a system relation along with a specific combination of buffer-flushing, a line change was not written to the after-image journal (.AIJ) file. As a consequence, if an attempt was made to recover the database, the database was corrupted and a bugcheck resulted.

This problem is fixed in the mandatory update for V4.0.

6.3 IMPORT/EXPORT Problems Fixed in the Mandatory Update for Version 4.0

This section contains notes and problems fixed in the SQL and RDO IMPORT/EXPORT operations.

6.3.1 Importing a Database with Tables Containing Lists (Segmented Strings) Failed

Importing a database with tables containing lists (segmented strings) failed with one of the following symptoms:

- The ''RDB-F-BAD_SEGSTR_ID, invalid segmented string identifier'' error message was returned.
- A bugcheck resulted with an exception at RDMS$$BLOB_ASSIGN + 0EC.

Users observed these problems while importing databases that contained lists in tables that used placement via hashed indexes. However, these problems could also occur at other times when users were storing records with lists.

The only workaround is to remove the PLACEMENT VIA INDEX clause from the tables affected and perform the EXPORT/IMPORT operation again.

This problem is fixed in the mandatory update for V4.0 by RDMSHRP ECO 8.

6.4 SQL Problems Fixed in the Mandatory Update for V4.0

This section contains notes and problems fixed in the SQL interface.

6.4.1 SQL$STARTUP.COM Startup File Contained an Error in the SQL/Services Startup Logical Name

In Rdb/VMS Version 4.0, the SQL startup file contained a commented out line for starting SQL/Services that incorrectly specified the SYS$SYSTARTUP logical name instead of SYS$STARTUP.

This problem is fixed in the mandatory update for V4.0.

6.4.2 Opening a Cursor That Was Already Opened Caused the Cursor to Lose Its State

In a precompiled or module language program that uses a cursor, the cursor is first opened, fetched, and opened once again. With previous versions of SQL, the second OPEN statement incorrectly repositioned the cursor back to the first record rather than returning an error.

This problem is fixed in the mandatory update for V4.0. Recompile any modules associated with the programs that have this error condition and that contain the cursor statements. Example 6–1 illustrates the problem and correct program behavior.

Example 6–1 Cursor Losing Its State

```
/*
    Do not lose cursor state on subsequent opens of a cursor.
*/
#include stdio
main()
{
int SQLCODE;
int fetch_status;
EXEC SQL DECLARE SCHEMA FILENAME 'date';

struct {
    char    pnum[6],pname[20],color[6];
    short   weight;
    char    city[15];
    } c_rec;

EXEC SQL DECLARE P_CURSOR CURSOR FOR SELECT * FROM P ;

printf("OPEN CURSOR FIRST TIME:\n");
EXEC SQL OPEN P_CURSOR;
if (SQLCODE != 0)
    printf("Expected Success on first error, but got an error\n" );

EXEC SQL OPEN P_CURSOR;
if (SQLCODE != 0)
    printf("Expected an error and got it.\n" );
```

(continued on next page)

Example 6–1 (Cont.) Cursor Losing Its State

```
/*
    Print using the C record
*/
EXEC SQL FETCH P_CURSOR INTO :c_rec;
fetch_status = SQLCODE;
EXEC SQL OPEN P_CURSOR;
if (SQLCODE != 0)
    printf("Expected an error and got it.\n" );

while (fetch_status == 0)
    {
    printf("\n%s %s %s %d %s",
        c_rec.pnum, c_rec.pname, c_rec.color, c_rec.weight, c_rec.city);
    EXEC SQL FETCH P_CURSOR INTO :c_rec;
    fetch_status = SQLCODE;
    EXEC SQL OPEN P_CURSOR;
    if (SQLCODE != 0)
        printf("Expected an error and got it.\n" );

        }
EXEC SQL CLOSE P_CURSOR;

EXEC SQL ROLLBACK WORK;
}
```

6.4.3 Executing the ROLLBACK Statement with OPEN LIST Cursors Left List Cursors in an Unusable State

If a ROLLBACK statement was executed when an INSERT ONLY LIST CURSOR was open, the cursor was implicitly closed; however, the cursor was left in a state where subsequent OPEN statements caused an invalid segmented string handle error. This problem is fixed in the mandatory update for V4.0.

Example 6–2 shows a host C program that illustrates the problem and proper program behavior.

Example 6–2 Executing ROLLBACK with LIST CURSORS in a Host C Program

```
#include <stdio.h>
#include <string.h>

typedef struct {
        short int   size;
        char        body[10];
        }                   varchar_t;

void    insert_function (
        int             col_a,
        varchar_t      *col_b);
```

(continued on next page)

Example 6–2 (Cont.) Executing ROLLBACK with LIST CURSORS in a Host C Program

```
    main ()

    {
    static  varchar_t   list_data;

    printf ("first function call...list_data has data\n");
    list_data.size = 10;
    memcpy (list_data.body, "ABCDEFGHIJ", 10);
    insert_function (0, &list_data);

    printf ("second function call...list_data is null\n");
    list_data.size = 0;
    insert_function (1, &list_data);

    printf ("third function call...list_data has data\n");
    list_data.size = 10;
    memcpy (list_data.body, "KLMNOPQRST", 10);
    insert_function (2, &list_data);
    }

void    insert_function (
            int             col_a,
            varchar_t      *col_b)
    {
    int                     sqlcode;
    int                     a,b;
    char                    db_name[256];

    strncpy (db_name, "date1.rdb", 255);

    tab_ins_cur_open (
        &sqlcode,
        db_name);
    if (sqlcode != 0)
        {
        printf ("open cursor failed\n");
        sql$signal();
        abort (
            &sqlcode,
            db_name);
        return;
        }

    tab_ins_cur_insert (
        &sqlcode,
        db_name,
        &col_a);
    if (sqlcode != 0)
        {
        printf ("insert cursor failed\n");
        sql$signal ();
        abort (
            &sqlcode,
            db_name);
        return;
        }
```

(continued on next page)

Example 6–2 (Cont.) Executing ROLLBACK with LIST CURSORS in a Host C Program

```
/* insert data into list cursor */

tab_list_ins_cur_open (
        &sqlcode,
        db_name);
if (sqlcode != 0)
    {
    printf ("open cursor for list failed\n");
    sql$signal ();
    abort (
        &sqlcode,
        db_name);
    return;
    }

if (col_b->size != 10)
    {
    printf ("Error.  List data should be 10 bytes, instead is %d bytes\n",
        col_b->size);
    abort (
        &sqlcode,
        db_name);
    return;
    }

tab_list_ins_cur_insert (
    &sqlcode,
    db_name,
    col_b);
if (sqlcode != 0)
    {
    printf ("insert cursor for list failed\n");
    sql$signal ();
    abort (
        &sqlcode,
        db_name);
    return;
    }

/* close all cursors */

tab_list_ins_cur_close (
    &sqlcode,
    db_name);
if (sqlcode != 0)
    {
    printf ("close cursor for list failed\n");
    sql$signal();
    abort (
        &sqlcode,
        db_name);
    return;
    }
```

(continued on next page)

Example 6–2 (Cont.) Executing ROLLBACK with LIST CURSORS in a Host C Program

```
    tab_ins_cur_close (
        &sqlcode,
        db_name);
    if (sqlcode != 0)
        {
        printf ("close cursor failed\n");
        sql$signal();
        abort (
            &sqlcode,
            db_name);
        return;
        }

    commit (
        &sqlcode,
        db_name);
    if (sqlcode != 0)
        {
        printf ("commit failed\n");
        sql$signal();
        abort (
            &sqlcode,
            db_name);
        return;
        }

    printf ("record stored\n");

    }
```

Example 6–3 shows an SQL module language program that illustrates the problem and proper program behavior.

Example 6–3 Executing ROLLBACK with LIST CURSORS in SQL Module Language

```
MODULE sqlmod LANGUAGE GENERAL AUTHORIZATION ABC

DECLARE LOCAL abc SCHEMA AUTHORIZATION FOR
    COMPILETIME FILENAME 'date'
    RUNTIME FILENAME 'date'

DECLARE tab_ins_cur INSERT ONLY TABLE CURSOR FOR
    SELECT col_a, col_b
    FROM abc.tab

DECLARE tab_list_ins_cur INSERT ONLY LIST CURSOR FOR
    SELECT col_b
    WHERE CURRENT OF tab_ins_cur
```

(continued on next page)

Example 6–3 (Cont.) Executing ROLLBACK with LIST CURSORS in SQL Module Language

```
procedure tab_ins_cur_open
    SQLCODE
    SCHEMA                  char (255);
OPEN tab_ins_cur;

procedure tab_ins_cur_insert
    SQLCODE
    SCHEMA                  char (255)
    col_a                   integer;
INSERT INTO CURSOR tab_ins_cur (col_a)
    VALUES (col_a);

procedure tab_ins_cur_close
    SQLCODE
    SCHEMA                  char (255);
CLOSE tab_ins_cur;

procedure tab_list_ins_cur_open
    SQLCODE
    SCHEMA                  char (255);
OPEN tab_list_ins_cur;

procedure tab_list_ins_cur_insert
    SQLCODE
    SCHEMA                  char (255)
    col_b                   varchar (10);
INSERT INTO CURSOR tab_list_ins_cur
    VALUES (col_b);

procedure tab_list_ins_cur_close
    SQLCODE
    SCHEMA                  char (255);
CLOSE tab_list_ins_cur;

procedure commit
    SQLCODE
    SCHEMA                  char (255);
commit;

procedure abort
    SQLCODE
    SCHEMA                  char (255);
rollback;
```

6.4.4 Executing the COMMIT Statement with OPEN LIST Cursors Did Not Commit the Newly Created Lists

In Rdb/VMS Version 4.0, if a COMMIT statement was executed when an INSERT ONLY LIST CURSOR was open, the cursor was implicitly closed, but the list created by the cursor was not committed. An error in the semantics of the CREATE statement caused this problem.

This problem is fixed in the mandatory update for V4.0. Recompile any programs that exhibit behavior of this type.

Example 6–4 shows an embedded C program that illustrates the problem and the corrected behavior.

Example 6–4 Executing COMMIT with OPEN LIST Cursors

```
main()
{

#define SQLCODE_SUCCESS 0
#define SQLCODE_EOS 100
#define SQLCODE_RDBERR -1

        int SQLCODE;
        extern struct {
                int RDB$LU_NUM_ARGUMENTS;
                int RDB$LU_STATUS;
                int RDB$LU_ARGUMENTS[18]; } RDB$MESSAGE_VECTOR;

        char    name[26];
        char    element[ 30 ];
        char    seg[9];
        short   seg_ind;
        short   name_ind;
        short   ind1;
        short   element_ind;
/*
*/
EXEC SQL WHENEVER SQLERROR GOTO PRINT_ERROR;
EXEC SQL WHENEVER SQLWARNING GOTO PRINT_ERROR;
EXEC SQL DECLARE SCHEMA FILENAME DATE;
EXEC SQL DECLARE TCURSOR INSERT ONLY TABLE CURSOR FOR SELECT T1, BLOB FROM TMP;
EXEC SQL DECLARE FCURSOR TABLE CURSOR FOR SELECT T1, BLOB FROM TMP;

EXEC SQL DECLARE LCURSOR INSERT ONLY LIST CURSOR FOR
        SELECT BLOB WHERE CURRENT OF TCURSOR;

EXEC SQL DECLARE A1 READ ONLY LIST CURSOR FOR SELECT BLOB
                        WHERE CURRENT OF FCURSOR;
EXEC SQL SET TRANSACTION READ WRITE;

EXEC SQL OPEN TCURSOR;
EXEC SQL INSERT INTO CURSOR TCURSOR (T1) VALUES (:name);

EXEC SQL OPEN LCURSOR;
EXEC SQL INSERT INTO CURSOR LCURSOR VALUES ('1111111111 ');
EXEC SQL INSERT INTO CURSOR LCURSOR VALUES ('XXXXXXXXXX ');
EXEC SQL INSERT INTO CURSOR LCURSOR VALUES ('@@@@@@@@@@ ');
EXEC SQL INSERT INTO CURSOR LCURSOR VALUES ('########## ');
EXEC SQL INSERT INTO CURSOR LCURSOR VALUES ('%%%%%%%%%% ');
EXEC SQL INSERT INTO CURSOR LCURSOR VALUES ('ZZZZZZZZZZ ');
/*
    Close all open cursors and commit
*/
EXEC SQL COMMIT;

EXEC SQL OPEN FCURSOR;
        ind1 = 0;
        EXEC SQL FETCH FCURSOR INTO :name:name_ind,:seg:seg_ind;
        while (SQLCODE != SQLCODE_EOS) {
            printf( "t1: (%d) '%s'\n", name_ind, &name);
```

(continued on next page)

Example 6–4 (Cont.) Executing COMMIT with OPEN LIST Cursors

```
            printf( "List elements for Blob field \n" );
            exec sql open a1;
            if (SQLCODE != SQLCODE_SUCCESS)
                goto PRINT_ERROR;
            if (seg_ind == -1) {
                printf( "NULL LIST\n" );
            }
            exec sql FETCH A1 INTO :element:element_ind;
            while (SQLCODE != SQLCODE_EOS) {
                if (element_ind != -1) {
                    printf( "seg: '%s' segind: %d\n", element, element_ind);
                } else {
                    printf( "NULL LIST ELEMENT\n" );
                }
                exec sql fetch a1 into :element:element_ind;
            }
            /*
             * Make sure the operations on the table cursor interact
             * properly with the list cursor.
             */
        EXEC SQL CLOSE A1;
        EXEC SQL FETCH FCURSOR INTO :name:name_ind,:seg:seg_ind;
        }
EXEC SQL CLOSE FCURSOR;

EXEC SQL ROLLBACK;
goto DONE;

PRINT_ERROR:
        printf("sql error number=> %d\n",SQLCODE);
        exec sql rollback;
        SQL$SIGNAL(RDB$MESSAGE_VECTOR);
DONE:
;
}
```

6.4.5 The OPEN Statement of an INSERT TABLE CURSOR Did Not Properly Return Error Status

In Rdb/VMS Version 4.0, if the OPEN statement of an INSERT ONLY TABLE CURSOR statement was the first SQL statement executed in a program, and the database referenced by the program did not exist, the OPEN statement did not return the correct error information.

This problem is fixed in the mandatory update for V4.0. Recompile any programs that exhibit this behavior to correct the problem.

Example 6–5 shows an embedded C program that illustrates the problem and the corrected behavior.

Example 6–5 OPEN Statement Not Returning Error Information

```
#include <stdio.h>
#include <string.h>

void    insert_function ();

int     main ()

    {
    printf ("first function call...\n");
    insert_function ();
    printf ("second function call...\n");
    insert_function ();
    }

void    insert_function ()
    {
    int                     sqlcode;
    int                     a,b;
    char                    db_name[256];

    strncpy (db_name, "database_file_that_does_not.exist", 255);

    a = 1;
    a = 2;

    tab_ins_cur_open (
        &sqlcode,
        db_name);
    if (sqlcode != 0)
        {
        printf ("open cursor failed\n");
        sql$signal();
        abort (
            &sqlcode,
            db_name);
        return;
        }

    tab_ins_cur_insert (
        &sqlcode,
        db_name,
        &a,
        &b);
    if (sqlcode != 0)
        {
        printf ("insert cursor failed\n");
        sql$signal ();
        abort (
            &sqlcode,
            db_name);
        return;
        }
```

(continued on next page)

Example 6–5 (Cont.) OPEN Statement Not Returning Error Information

```
    tab_ins_cur_close (
        &sqlcode,
        db_name);
    if (sqlcode != 0)
        {
        printf ("close cursor failed\n");
        sql$signal();
        abort (
            &sqlcode,
            db_name);
        return;
        }

    commit (
        &sqlcode,
        db_name);
    if (sqlcode != 0)
        {
        printf ("commit failed\n");
        sql$signal();
        abort (
            &sqlcode,
            db_name);
        return;
        }

    printf ("record stored\n");

    }
```

Example 6–6 shows an SQL module language program that illustrates the problem and the corrected behavior.

Example 6–6 OPEN Statement Not Returning Error Information in SQL Module Language

```
MODULE sqlmod LANGUAGE GENERAL AUTHORIZATION ABC

DECLARE LOCAL abc SCHEMA AUTHORIZATION FOR
    COMPILETIME FILENAME date
    RUNTIME FILENAME schema

DECLARE tab_ins_cur INSERT ONLY CURSOR FOR
    SELECT col_a, col_b
    FROM abc.tab

procedure tab_ins_cur_open
    SQLCODE
    SCHEMA                  char (255);
OPEN tab_ins_cur;

procedure tab_ins_cur_insert
    SQLCODE
    SCHEMA                  char (255)
    col_a                   integer
    col_b                   integer;
INSERT INTO CURSOR tab_ins_cur (col_a, col_b)
    VALUES (col_a, col_b);
```

(continued on next page)

Example 6–6 (Cont.) OPEN Statement Not Returning Error Information in SQL Module Language

```
procedure tab_ins_cur_close
    SQLCODE
    SCHEMA                  char (255);
CLOSE tab_ins_cur;

procedure commit
    SQLCODE
    SCHEMA                  char (255);
COMMIT;

procedure abort
    SQLCODE
    SCHEMA                  char (255);
ROLLBACK;
```

6.4.6 Records Included from the Data Dictionary in the C Preprocessor Did Not Null Terminate Character Strings

The ANSI/ISO standard dictates that SQL interpret C character strings as being terminated with the NULL character. SQL does so for definitions explicitly defined in the user's embedded C programs; however, SQL for Rdb/VMS Version 4.0 did not make the same interpretation for records included from the data dictionary. SQL's handling of records from the data dictionary was incorrect and in conflict with the Rdb/VMS Version 4.0 documentation.

This problem is fixed in the mandatory update for V4.0. SQL now interprets records included from the data dictionary as being terminated with the NULL character, as the documentation states. To correct the problem, recompile the programs that are affected by this behavior.

Example 6–7 shows an embedded C program that illustrates the problem and the corrected behavior.

Example 6–7 Records from the Data Dictionary Not Terminated with the NULL Character

```
#include stdio
main()
{
int SQLCODE;
EXEC SQL DECLARE SCHEMA PATHNAME "cdd$default.date";
EXEC SQL INCLUDE FROM DICTIONARY 'CDD$DEFAULT.DATE.RDB$RELATIONS.P';
struct p p_rec;
EXEC SQL DECLARE P_CURSOR CURSOR FOR SELECT * FROM P;

EXEC SQL OPEN P_CURSOR;
EXEC SQL FETCH P_CURSOR INTO :p_rec;
while (SQLCODE == 0)
    {
    printf("\n%s %s %s %d %s",
        p_rec.pnum, p_rec.pname, p_rec.color, p_rec.weight, p_rec.city);
    EXEC SQL FETCH P_CURSOR INTO :p_rec;
    }
EXEC SQL CLOSE P_CURSOR;
}
```

6.4.7 Triggers Created with Long Source Text Strings Were Improperly Displayed

When a trigger is created in a program, the source text of the trigger is saved and then can be displayed by the SHOW TRIGGER command. In Rdb/VMS Version 4.0, the SHOW TRIGGER command displayed the saved source text as if it had been created and saved by interactive SQL. How programs and interactive SQL save trigger source text differs. Thus, because triggers created in programs do not conform to interactive SQL format, the SHOW TRIGGER command truncated the source text to the first 256 characters.

This problem is fixed in the mandatory update for V4.0. Interactive SQL now properly handles the printing of program-generated triggers, as shown in Example 6–8.

Example 6–8 Triggers Properly Displayed

```
SQL> SHOW TRIG BIGTRG
     BIGTRG
 Source:
 BIGTRG BEFORE DELETE ON EMPLOYEES ( DELETE FROM DEGREES
     DEGREESP_GRADEN_DIPLOMAS_XXXX WHERE
     DEGREESP_GRADEN_DIPLOMAS_XXXX . EMPLOYEE_ID = EMPLOYEES .
     EMPLOYEE_ID ) FOR EACH ROW ( DELETE FROM JOB_HISTORY
     JOBHISTORY_HELE_GESCHIEDENIS WHERE JOBHISTORY_HELE_GESCHIEDENIS
     . EMPLOYEE_ID = EMPLOYEES . EMPLOYEE_ID ) FOR EACH ROW ( DELETE
     FROM SALARY_HISTORY SALARHISTORY_HELE_GESCHIEDENIS WHERE
     SALARHISTORY_HELE_GESCHIEDENIS . EMPLOYEE_ID = EMPLOYEES .
     EMPLOYEE_ID ) FOR EACH ROW ( DELETE FROM RESUMES
     RESUMEES_WAT_JE_MAAR_VERZINT WHERE RESUMEES_WAT_JE_MAAR_VERZINT
     . EMPLOYEE_ID = EMPLOYEES . EMPLOYEE_ID ) FOR EACH ROW ( UPDATE
     DEPARTMENTS DEPARTMNT_MET_EIGEN_MINISTER SET
     DEPARTMNT_MET_EIGEN_MINISTER . MANAGER_ID = NULL WHERE
     DEPARTMNT_MET_EIGEN_MINISTER . MANAGER_ID = EMPLOYEES .
     EMPLOYEE_ID ) FOR EACH ROW
```

6.4.8 Triggers Created from Programs Had Their Source Text Truncated by a Word

In Rdb/VMS Version 4.0, the last word of the text saved for a trigger was truncated for triggers created from programs.

This problem is fixed in the mandatory update for V4.0. To reveal the missing word, you must rerun the program that creates the triggers after the Rdb/VMS Version 4.0 mandatory update kit is installed. Note that the trigger definition is correct; only the source text that SQL displayed for the trigger text was wrong.

Example 6–9 displays a trigger in which the example's text is truncated to leave out the word ''ROW.''

Example 6–9 Trigger Text Truncated

```
SQL> SHOW TRIG BIGTRG
     BIGTRG
 Source:
 BIGTRG BEFORE DELETE ON EMPLOYEES ( DELETE FROM DEGREES
     DEGREESP_GRADEN_DIPLOMAS_XXXX WHERE
     DEGREESP_GRADEN_DIPLOMAS_XXXX . EMPLOYEE_ID = EMPLOYEES .
     EMPLOYEE_ID ) FOR EACH ROW ( DELETE FROM JOB_HISTORY
     JOBHISTORY_HELE_GESCHIEDENIS WHERE JOBHISTORY_HELE_GESCHIEDENIS
     . EMPLOYEE_ID = EMPLOYEES . EMPLOYEE_ID ) FOR EACH ROW ( DELETE
     FROM SALARY_HISTORY SALARHISTORY_HELE_GESCHIEDENIS WHERE
     SALARHISTORY_HELE_GESCHIEDENIS . EMPLOYEE_ID = EMPLOYEES .
     EMPLOYEE_ID ) FOR EACH ROW ( DELETE FROM RESUMES
     RESUMEES_WAT_JE_MAAR_VERZINT WHERE RESUMEES_WAT_JE_MAAR_VERZINT
     . EMPLOYEE_ID = EMPLOYEES . EMPLOYEE_ID ) FOR EACH ROW ( UPDATE
     DEPARTMENTS DEPARTMNT_MET_EIGEN_MINISTER SET
     DEPARTMNT_MET_EIGEN_MINISTER . MANAGER_ID = NULL WHERE
     DEPARTMNT_MET_EIGEN_MINISTER . MANAGER_ID = EMPLOYEES .
     EMPLOYEE_ID ) FOR EACH
```

6.5 SQL/Services Problems Fixed in the Mandatory Update for V4.0

This section contains notes and problems fixed in SQL/Services.

6.5.1 Reinstalling SQL/Services APIs After Installation of Mandatory Update Kit for Rdb/VMS Version 4.0

After the mandatory update kit for V4.0 is installed, you must reinstall any of the client Application Programming Interfaces (APIs) that you plan to use. Reinstallation makes available all changes applied to the SQL/Services APIs in the mandatory update for V4.0.

Refer to the Rdb/VMS V4.0 *VAX Rdb/VMS Installation Guide* for complete SQL/Services API installation instructions.

6.5.2 SQL/Services Sample Application ULTRIX API Compiled with a Syntax Error

In Rdb/VMS V4.0, the SQL/Services SQLSRV$DYNAMIC sample application received a syntax error when compiled on the ULTRIX operating system.

This syntax error is fixed in the mandatory update for V4.0.

6.5.3 SQL/Services MS–DOS IVP Failed with a –2003 and 9 Error Status Codes, Indicating That Numbers Were Not Being Allowed Within Server Node Names

In Rdb/VMS V4.0, the SQL/Services Installation Verification Program (IVP) for the MS–DOS API failed with the following message:

```
VAX SQL Services sample program failed

SQLCA: SQLCODE: -2003  SQLERRD[0]: 9 SQLERRD[2] 0
```

This error occurred when the server node name specified in the IVP contained a character other than an alphabetic character (a–z, A–Z).

This problem is fixed in the mandatory update for V4.0. SQL/Services now allows numeric as well as alphabetic characters in server node names.

6.5.4 SQL/Services Length Packet Split Problem

For some isolated cases in Rdb/VMS V4.0, the SQL/Services API was unable to read large server messages because of a network packet split problem.

This problem is fixed in the mandatory update for V4.0.

6.5.5 SQL/Services ULTRIX API Was Not Freeing Network Connections

In Rdb/VMS V4.0, the SQL/Services Application Programming Interface (API) sqlsrv_release routine did not free network connections for an ULTRIX SQL/Services application. This problem occurred in applications that called the sqlsrv_associate routine multiple times within one session.

This problem is fixed in the mandatory update for V4.0.

6.5.6 SQL/Services Communication Server Did Not Report Error Status

In Rdb/VMS V4.0, if the SQL/Services communication server received an unexpected error (for example, ''No privilege to create a mailbox''), the server process exited without writing the error to the log file.

This problem is fixed in the mandatory update for V4.0. The startup file has been changed to enable logging of communication server errors. SQL/Services creates log files in the following locations:

- SYS$MANAGER:SQLSRV$.LOG
- SYS$MANAGER:SQLSRV$.ERR

Also, to enable this change, a new file is created on your system during installation of the mandatory update for Rdb/VMS V4.0:

```
SYS$STARTUP:SQLSRV$RUN.COM
```

SQL/Services automatically executes this command procedure at startup.

6.5.7 SQL/Services Shutdown Procedure Hung, Causing the Subsequent Startup Procedure Not to Work

In Rdb/VMS V4.0, if unexpected errors occurred in the communication server and the server exited without deleting the mailbox used to communicate with the SQL/Services Shutdown utility, then the Shutdown utility did not work properly. In fact, SQLSRV$UTL.EXE could become hung, ultimately causing the RMONSTOP procedure to hang if SQLSRV$SHUTDOWN was called from within this command procedure.

Further, if the user subsequently tried to start up another server, because the mailbox still existed, the user got an error from the startup command procedures, indicating ''SQLSRV server has been previously started''. Thus, in this situation, neither the startup nor the shutdown procedure worked properly.

These problems are fixed in the mandatory update for V4.0. The Shutdown utility now does not require a response from the communication server during shutdown. Also, the startup procedure has been changed (as noted in Section 6.5.8), as has the shutdown procedure.

6.5.8 SQL/Services Startup File Changes

SQL/Services includes a number of changes in its startup file for the mandatory update for Rdb/VMS V4.0. The startup file:

- Enables the logging of communication server errors. Refer to Section 6.5.6 for details.
- Checks for additional privileges required by the communication server:

  ```
  CMKRNL, SYSPRV, OPER, NETMBX, PRMMBX, SYSNAM, DETACH
  ```

- Increases to 256 the I/O buffer count for the communication server.
- Has corrected an error in the parsing of the SQLSRV$SERVER account information. On some systems, SQL/Services extracted the [owner] string instead of the UIC information that was needed.

6.5.9 SQL/Services Macintosh API Code Fixes

This section describes several software fixes that have been made to the SQL/Services Macintosh API for the mandatory update for Rdb/VMS V4.0.

6.5.9.1 SQL/Services SQLSRV$Volume Installation Volume Could Not Be Accessed on the Macintosh

In Rdb/VMS V4.0, during installation of the SQL/Services Macintosh API, the SQLSRV$Volume installation volume could be successfully added by the file server; however, the volume might not have been accessible from your Macintosh. For example, the volume might have been visible in the list of available volumes but dimmed out, indicating that it could not be accessed. This error occurred because the backup file used to create the installation volume on the file server contained an extraneous file (MSAF$VOLUME.MSAF$AFP) that caused the file server to mount the volume incorrectly.

This problem is fixed in the mandatory update for V4.0.

6.5.9.2 PATHWORKS DECtask Tool Name Changed

In Rdb/VMS V4.0, the SQL/Services Macintosh API did not work with the DECnet transport mechanism because of a name mismatch with the DECnet Tool provided by PATHWORKS for Macintosh V1.0.

In the Rdb/VMS V4.0 mandatory update, this problem is fixed. The SQL/Services resource file now looks for the ''DECnet Tool'' instead of ''DECtask Tool''. The ''DECnet Tool'' should exist in the Communications Folder of the System Folder on your Macintosh system after installation of PATHWORKS for Macintosh.

6.5.9.3 SQL/Services Macintosh API Fix for Macintoshes Based on the Motorola 68000 Chip

For Rdb/VMS Version 4.0, an application calling the SQL/Services Macintosh API for MPW encountered an ''illegal instruction'' error when running the application on Macintosh systems based on the Motorola 68000 chip. Macintosh systems based on the 68000 include the Macintosh P/Cs, Macintosh SE, and Macintosh portables.

This problem is fixed in the mandatory update for V4.0.

6.6 RDO, Callable RDO, RDBPRE, and RDML Problems Fixed in the Mandatory Update for V4.0

This section contains notes and problems fixed in the RDO, Callable RDO, RDBPRE, and RDML interfaces.

6.6.1 RDBPRE Generated Incorrect Code for Request Handles

The following run-time error could occur in RDBPRE programs that used request handles:

```
%RDB-F-BAD_REQ_HANDLE, invalid request handle
%TRACE-F-TRACEBACK, symbolic stack dump follows
module name     routine name                     line       rel PC    abs PC
                                                          00054A23  00054A23
----- above condition handler called with exception 0138803C:
%RDB-F-BAD_REQ_HANDLE, invalid request handle
----- end of exception message
RDB$MIKE        RDB$START_1_942272_82CD84                 0000003A  0000074A
MIKE$MAIN       MIKE$MAIN                          40     0000003B  0000063B
```

This problem was caused by RDBPRE, which generated incorrect MACRO code from the source program. Consider the following RDBPRE BASIC program:

```
10      !
&rdb&   INVOKE DATABASE FILENAME "MF_PERSONNEL"
&rdb&   DECLARE_STREAM (REQUEST_HANDLE DIDDLE%) EMP_STREAM
&rdb&   DECLARE_STREAM (REQUEST_HANDLE DIDDLE%) EMP_STREAM
&rdb&   USING EMP IN EMPLOYEES WITH EMP.EMPLOYEE_ID > A$
&rdb&   START_STREAM EMP_STREAM
&rdb&   ON ERROR
            GOTO ERROR_HANDLER
&rdb&   END_ERROR
        STOP
error_handler:
        CALL LIB$STOP(RDB$STATUS)
        END
```

If you compile this program with RDBPRE, save the MACRO code by defining the logical name RDMS$KEEP_PREP_FILES as ''Y'', and inspect the MACRO code in the .MAR file, you will find the following code section:

```
.ENTRY  Rdb$START_1_942272_82CD84,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
MOVAB   G^Rdb$L_TRANSACTION_HANDLE,R6
MOVL    8(AP),R5                ; Request handle
MOVAL   G^RDB$DBHANDLE,R4               ; DBhandle
MOVAB   G^RDMS$GX_BLR_1_942272_82CA76,R8        ; BLR
MOVL    #<RDMS$GX_BLR_1_942272_82CA76_E-RDMS$GX_BLR_1_942272_82CA76>,
    R7 ; BLR length
MOVL    8(AP),R11       ; Message buffer
MOVL    #MESSAGE$1_2_LENGTH,R10
MOVL    #2,R9
$BSBW   Rdb$DEF_START_TXN_R2_TO_R11
BLBS    R0,10000$
CALLG   @4(AP),G^LIB$STOP
```

Note that the lines with ''Request handle'' and ''Message buffer'' comments both refer to 8(AP). In this case, the request handle should be 12(AP).

This problem is fixed in the mandatory update for V4.0 by RDBPRE ECO 1.

6.6.2 A CDD/Plus Informational Message Caused RDML to Abort Compilation

A CDD/Plus informational message caused RDML to abort compilation. This problem can be avoided by using the CDO CLEAR NOTICE command.

This problem is fixed in the mandatory update for V4.0.

6.6.3 An RDML-E-READ_ONLY Error Was Returned When Attempting to Update COMPUTED BY Fields

When RDML attempted to warn users about updating read-only (COMPUTED BY) fields, it incorrectly aborted compilation and reported an RDML-E-READ_ ONLY error.

This problem is fixed in the mandatory update for V4.0.

6.6.4 Problem with Callable RDO and Varying String Descriptors

In V4.0, passing VS descriptor strings (from PL/I) caused RDB$INTERPRET to return illegal characters or to fail with syntax errors. The errors returned included the following messages:

```
-RDO-F-LOOK_FOR, syntax error, looking for commit_tail, found SALHIST instead
%TRACE-E-TRACEBACK, symbolic stack dump follows
   .
   .  NOTE: where the above syntax error is on a string which appears valid
   .
Unexpected error - terminating program
%PLI-E-ERROR, PL/I ERROR condition.
```

or

```
%PLI-F-ERROR, PL/I ERROR condition.
-RDO-F-ILLCHAR, illegal character detected
```

The problem occurred on CALLABLE$RDO programs that used a varying string to hold the command string. For example, in PL/I the following example failed with the errors previously mentioned:

```
DECLARE RDB_COMMAND        CHARACTER(1024) varying;

RDB_COMMAND = 'DATABASE FILENAME "PERSONNEL"';
RDB_STATUS = RDB$INTERPRET(DESCRIPTOR (RDB_COMMAND));
    IF (^RDB_STATUS_SUCCESS) THEN
        CALL HANDLE_ERROR;
```

A workaround is to drop the word varying.

This problem is fixed in the mandatory update for V4.0 by RDBINTSHR ECO 1 and allows command strings for RDB$INTERPRET to be VS descriptors.

In V4.1, RDB$INTERPRET will fully support VS descriptors for !VAL placeholders.

6.7 RMU Problems Fixed in the Mandatory Update for V4.0

This section contains notes and problems fixed in the RMU interface.

6.7.1 RMU/VERIFY Returned Spurious Errors Involving Fragmented Records

When you performed an RMU/VERIFY operation, the following errors could be incorrectly returned:

```
%RMU-W-INVRELID, invalid relation id at dbkey 61:9198:4
          expected relation id 28, found 12848

%RMU-W-BADFRALEN, area ANNIE_AREA, page 9543, line 24
          storage record UNKNOWN, bad expanded fragment length
          expected: 34, found: 25,
%RMU-I-FRACHNPOS, pointed to by fragment on page 9198, line 21

%RMU-W-BADFRAPTR, area ANNIE_AREA, page 9195, line 19
          storage record UNKNOWN, bad fragment chain pointer
          expected 1 through 15009, found page number 1605660.
```

```
%RMU-W-BADFRAEND, area ANNIE_AREA, page 304, line 8
          storage record UNKNOWN, bad last fragment pointer
          expected: 304:8, found: 132864:1.
```

In addition, %SYSTEM-W-ROPRAND errors and bugcheck dumps could result from this problem.

This problem was caused by a buffer flushing problem in the RMU/VERIFY command when RMU/VERIFY was processing record fragments.

A workaround is to restructure your database to avoid record fragmentation.

This problem is fixed in the mandatory update for V4.0.

6.7.2 RMU/CONVERT Failed with a Default Collating Sequence Defined

If an Rdb/VMS V3.1, V3.1A, or V3.1B database had a default collating sequence, and RMU/CONVERT was used to convert the database, no error messages were returned. However, attempts to attach to the database failed with the following messages, and the database was unusable:

```
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-TABNOTDEF, relation RDBVMS$STORAGE_MAPS is not defined in database
```

Because V3.1, V3.1A, and V3.1B databases return an access violation when you attempt to export them with default collating sequences defined, there is no workaround to this problem.

This problem is fixed in the mandatory update for V4.0. A V3.1, V3.1A, or V3.1B database with a default collating sequence defined can now be successfully converted to a V4.0 database using the RMU/CONVERT command.

6.7.3 The /INTERVAL Qualifier of the RMU/BACKUP/AFTER_JOURNAL Command Miscalculated a Specified Interval Value

If a specified /INTERVAL qualifier value was greater than 400 seconds in the RMU/BACKUP/AFTER_JOURNAL/INTERVAL command, it was calculated to be the remainder of a division by 400. That is, a value of 500 seconds was interpreted as 100 seconds.

This problem is fixed in the mandatory update for V4.0.

6.7.4 Problem with RMU/SHOW USERS and RMU/SHOW SYSTEM Commands and VMS WORLD Privileges

If RMU was installed with VMS WORLD privilege, the RMU/SHOW SYSTEM and RMU/SHOW USERS commands were not available for use.

This problem is fixed in the mandatory update for V4.0 by RMU ECO 5.

6.7.5 RMU/REPAIR Command Caused Database Corruption–Problem I

When the RMU/REPAIR command was performed on a database, it processed pages that contained deleted records incorrectly and caused the database to become corrupt. When a subsequent RMU/VERIFY/ALL command was performed, it returned errors indicating that a dbkey no longer pointed to a record, then produced a bugcheck dump, and the process disappeared.

This problem is fixed in the mandatory update for V4.0.

6.7.6 RMU/REPAIR Command Caused Database Corruption–Problem II

The RMU/REPAIR command improperly updated the ABM pages, resulting in database corruption.

This problem is fixed in the mandatory update for V4.0.

6.7.7 RMU/DUMP and RMU/CLOSE Commands Required VMS SYSPRV Privilege

The RMU/DUMP and RMU/CLOSE commands no longer require the VMS SYSPRV privilege, even if the RMU.EXE image is installed with the VMS SYSPRV privilege. This was a restriction in V4.0.

A workaround is to use the OPEN IS AUTOMATIC clause when that is feasible, or designate a trusted person to perform these operations on request.

This problem is fixed in the mandatory update for V4.0. The new RMU behavior is as follows:

- If the user has VMS SYSPRV privilege, then allow access.
- Otherwise, attach to the database to check the user's privileges.
- If the attach to the database fails, abort.
- Otherwise, allow access if the user has SQL DBADM (RDO ADMINISTRATOR) privilege.

7

Known Problems, Restrictions, and Other Notes for the Mandatory Update for Rdb/VMS V4.0

This chapter describes known problems and restrictions relating to Rdb/VMS V4.0, and includes workarounds where appropriate. It also contains other information not discussed in the preceding chapters.

7.1 General Problems, Restrictions, and Notes

This section contains general problems, restrictions, and other notes.

7.1.1 VMS Lock Remastering Changed in VMS V5.4

VMS changed the lock remastering algorithm in VMS Version 5.4. This may result in the lock trees being mastered on one node to start with and then the lock tree may migrate to another node due to the lock remastering. If the new node (where the locks get remastered) is a slower node, lock requests may take more time to be processed. You can force the lock manager to avoid certain nodes as candidates for remastering by setting the value for LCKDIRWT to 0 during SYSGEN. See the VMS documentation set for more information.

7.2 Problems, Restrictions, and Notes for All Interfaces

This section contains problems, restrictions, and other notes that pertain to all interfaces.

See also the Restrictions chapter of the V4.0 *VAX Rdb/VMS Release Notes* for other restrictions that still apply.

7.2.1 Using Quoted Threshold Values for Binary Data Types for Partitioning Data or Indexes Results in Data or Index Corruption

Data or index corruption can result from using quotes around threshold values for the signed quadword, signed longword, signed word, and signed byte data types when you partition data and indexes, shown in the following example:

```
RDO>
DEFINE DATABASE MIKE DICTIONARY IS NOT USED
  DEFINE STORAGE AREA ST1 FILENAME ST1 END ST1 STORAGE AREA
  DEFINE STORAGE AREA ST2 FILENAME ST2 END ST2 STORAGE AREA
  DEFINE STORAGE AREA ST3 FILENAME ST3 END ST3 STORAGE AREA
  DEFINE STORAGE AREA ST4 FILENAME ST4 END ST4 STORAGE AREA
  DEFINE STORAGE AREA ST5 FILENAME ST5 END ST5 STORAGE AREA
  DEFINE STORAGE AREA ST6 FILENAME ST6 END ST6 STORAGE AREA
  DEFINE STORAGE AREA ST7 FILENAME ST7 END ST7 STORAGE AREA.
DEFINE FIELD BILL DATA SIGNED QUADWORD.
DEFINE FIELD ACCT_NO DATA TEXT SIZE 5.
DEFINE RELATION ACCT. BILL. ACCT_NO. END.
COMMIT
```

```
DEFINE INDEX I1 FOR ACCT DUPLICATES ARE NOT ALLOWED
  STORE USING BILL WITHIN
  IST1 WITH LIMIT OF "01";IST2 WITH LIMIT OF "03";IST3 WITH LIMIT OF "05";
  IST4 WITH LIMIT OF "07";IST5 WITH LIMIT OF "09";IST6 WITH LIMIT OF "11";
  IST7.
  BILL. ACCT_NO. END.
COMMIT
DEFINE STORAGE MAP ACCT_MAP FOR ACCT RELATION
  STORE USING BILL WITHIN
  ST1 WITH LIMIT OF "01";ST2 WITH LIMIT OF "03";ST3 WITH LIMIT OF "05";
  ST4 WITH LIMIT OF "07";ST5 WITH LIMIT OF "09";ST6 WITH LIMIT OF "11";
  ST7 END.
COMMIT
START_TRANSACTION READ_WRITE
STOR A IN ACCT USING A.BILL=1;A.ACCT_NO="1" END_STORE
STOR A IN ACCT USING A.BILL=11;A.ACCT_NO="11" END_STORE
STOR A IN ACCT USING A.BILL=3;A.ACCT_NO="3" END_STORE
STOR A IN ACCT USING A.BILL=5;A.ACCT_NO="5" END_STORE
STOR A IN ACCT USING A.BILL=15;A.ACCT_NO="15" END_STORE
STOR A IN ACCT USING A.BILL=7;A.ACCT_NO="7" END_STORE
STOR A IN ACCT USING A.BILL=9;A.ACCT_NO="9" END_STORE
COMMIT
FOR A IN ACCT WITH A.BILL EQ 5 PRINT A.* END_FOR
FOR A IN ACCT WITH A.BILL LE 5 PRINT A.* END_FOR
```

If you have data and index records stored in the wrong partition for a table, reload the table with a storage map that does not use quotes around the threshold values for the signed quadword, signed longword, signed word, and signed byte data types. Note that quotes are permitted around the TEXT data type values and these work correctly.

7.2.2 Problem with SQL LIKE and RDO MATCHING Clauses

The RDO MATCHING clause, the SQL and RDO CONTAINING and STARTING WITH clauses, and the SQL LIKE clause work as expected on character string and integer data types, but all work a little differently with the DATE data types.

In each case, the syntax is accepted, but unexpected results are returned if a character expression is used because of the conversion of dates to text strings. When a number is used in the MATCHING expression or LIKE predicate, correct results are returned, but if a character expression is used, no results are returned, as shown in these examples.

```
RDO> FOR E IN EMPLOYEES WITH E.BIRTHDAY MATCHING "*1954*"
cont> PRINT E.BIRTHDAY END_FOR
 BIRTHDAY
 20-MAR-1954 00:00:00.00
 13-MAR-1954 00:00:00.00
 21-NOV-1954 00:00:00.00
 15-MAY-1954 00:00:00.00
 20-JUL-1954 00:00:00.00
RDO>
RDO> FOR E IN EMPLOYEES WITH E.BIRTHDAY MATCHING "*mar*"
cont> PRINT E.BIRTHDAY END_FOR
```

The same behavior is seen in SQL with the LIKE clause.

```
SQL> SELECT BIRTHDAY FROM EMPLOYEES WHERE BIRTHDAY LIKE "%1954%";
 BIRTHDAY
 20-Mar-1954
 13-Mar-1954
 21-Nov-1954
 15-May-1954
 20-Jul-1954
5 rows selected
SQL> SELECT BIRTHDAY FROM EMPLOYEES WHERE BIRTHDAY LIKE "%Mar%";
0 rows selected
SQL> SELECT BIRTHDAY FROM EMPLOYEES WHERE BIRTHDAY LIKE "%MAR%";
0 rows selected
SQL>
```

The reason for this apparent inconsistent behavior is as follows. The RDO MATCHING clause, the SQL and RDO CONTAINING and STARTING WITH clauses, and the SQL LIKE clause, all require TEXT as input. Therefore, the dates will be converted to text strings that have the YYYYNNDDHHMMSSCC format described in the *VAX Rdb/VMS SQL Reference Manual*. The match will be performed on all digit text strings of the date ("MAR" will never be seen because the month value will have been converted to "03"), and then the binary values are returned to SQL or RDO for printing in the format shown.

Use the RDO MATCHING clause with "*03*" or the SQL LIKE clause with "%03%" to get the results you are expecting.

7.2.3 RDB$REMOTE Account Has SYSTEM as Owner

If V4.0 is installed over an older version of Rdb/VMS that has SYSTEM as the owner of its directory, RDB$SERVER cannot create the NETSERVER.LOG file and fails on attach or on the first transaction. A workaround is to use the SET OWNER command to set the owner to RDB$REMOTE.

7.2.4 An Arithmetic Exception Results When Joining Integer Columns

The following error is possible when a join operation is performed using integer columns of different sizes:

```
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SYSTEM-F-INTOVF, arithmetic trap, integer overflow at PC=xxxxxx,PSL=01C00000
```

The problem occurs only when the columns being joined are each part of an index, and when the larger join column contains data that exceeds the maximum size of the smaller join column. The following example shows the problem.

```
$ RDO
DEFINE DATABASE MIKE DICTIONARY IS NOT USED.
DEFINE FIELD W1 DATA SIGNED WORD.
DEFINE FIELD L1 DATA SIGNED LONG.
DEFINE RELATION R1. W1. END.
DEFINE RELATION R2. L1. END.
DEFINE INDEX I1 FOR R1. W1. END.
DEFINE INDEX I2 FOR R2. L1. END.
STORE R IN R1 USING R.W1=1 END_STORE
STORE R IN R1 USING R.W1=2 END_STORE
STORE R IN R1 USING R.W1=3 END_STORE
STORE R IN R1 USING R.W1=4 END_STORE
STORE R IN R2 USING R.L1=100000 END_STORE
STORE R IN R2 USING R.L1=100000 END_STORE
STORE R IN R2 USING R.L1=100000 END_STORE
COMMIT
FOR A IN R1 CROSS B IN R2 WITH A.W1=B.L1 PRINT A.*,B.* END-FOR
ROLLBACK
EXIT
```

A workaround is to use the larger data type for both columns of the join operation. Note that indexes may have to be redefined as a workaround to this problem.

7.2.5 Collating Sequences That Use Two-to-Two Character Mapping Can Bugcheck

When you define a collating sequence that uses two-to-two character mapping, for example Thai collating sequences, a bugcheck can occur with the following exception:

```
EXCEPTION AT 003C2BC1
RDMS$$MCS$NCS_RECORD_8 + 000000401
```

Two-to-two character mapping is not supported in V4.0.

The following example duplicates the problem.

```
DEFINE DATABASE AX DICTIONARY IS NOT USED.
DEFINE COLLATING_SEQUENCE TEST TEST.
DEFINE FIELD TEST DATATYPE TEXT SIZE 2 COLLATING_SEQUENCE IS TEST.
DEFINE RELATION MIKE.
TEST.
END RELATION.
COMMIT
START_TRANSACTION READ_WRITE
STORE T IN MIKE USING T.TEST = "AA" END_STORE
STORE T IN MIKE USING T.TEST = "AA" END_STORE
STORE T IN MIKE USING T.TEST = "ZZ" END_STORE
STORE T IN MIKE USING T.TEST = "A " END_STORE
STORE T IN MIKE USING T.TEST = "Z " END_STORE
STORE T IN MIKE USING T.TEST = "BA" END_STORE
STORE T IN MIKE USING T.TEST = "AB" END_STORE
STORE T IN MIKE USING T.TEST = "C " END_STORE
FOR T IN MIKE PRINT T.* END_FOR
FOR T IN MIKE SORTED BY T.TEST PRINT T.* END_FOR
SHOW FIELD TEST
COMMIT
```

There is no workaround to this problem. This problem is fixed in V4.1.

7.2.6 Query with Keys Scans the Index Instead of Using Direct Tree Lookup

When a range list index retrieval is used in one or more of the indexes that dynamic leaf optimization uses, the key skip mechanism is not used. This causes many unwanted index records to be read, resulting in poor performance.

With range list index retrieval, several key ranges (using OR logic) on the same index are scanned in order. The index node is opened and the first key range is scanned. After the first key range is processed, a key skip is performed on the next key range. The key skip process is repeated for the remaining key ranges. This is how range list retrieval should work.

The following query reproduces this problem:

```
SELECT FIELD1, FIELD2 FROM TABLE1 WHERE KEYED_FIELD = "XXX" OR
KEYED_FIELD = "YYY";
```

A workaround is to use ORDER BY KEYED_FIELD in the previous query.

This problem is fixed in V4.1.

7.2.7 Synchronization Problem for an Empty Sorted Index

When two simultaneous attaches to the same database access an empty table via a sorted index, one attach may fail to see a row created by the other attach. Note that this problem occurs only in tables whose indexes reside in a single storage area; it does not occur in partitioned indexes nor does it occur in relations whose indexes are unmapped and reside by default in the RDB$SYSTEM logical area. Furthermore, as stated earlier, this problem occurs only if the table is empty when both processes attach to the database. This problem was originally fixed in V3.1B; however, when two released patches (LCKCCH31B.PAT and SORT_FIX_B.PAT) are applied, the problem recurs and produces the following exception:

```
(Exception at 00211A5C : RDMS$$KOD_REMOVE_TREE + 0000046B,
(%RDMS-F-BUGCHECK, fatal, unexpected error detected)
```

The problem still occurs in V4.0. The following example shows this problem:

```
SQL> CREATE SCHEMA FILENAME TEST_DB;
SQL> FINISH;

SQL> DECLARE SCHEMA FILE TEST_DB;
SQL> CREATE TABLE CANDIDATES
cont> (LAST_NAME CHAR (15),
cont> FIRST_NAME CHAR (15));
SQL> COMMIT;
SQL> CREATE INDEX CANDIDATES_INDEX ON CANDIDATES
cont> (LAST_NAME, FIRST_NAME) TYPE IS SORTED;
SQL> COMMIT;
SQL> FINISH;

- Table CANDIDATES is empty.

- User #1 attaches to the database.

SQL> DECLARE SCHEMA FILE TEST_DB;
SQL> SET TRANS READ ONLY;
SQL> SEL * FROM CANDIDATES;
0 rows selected
SQL> COMMIT;

- User #2 attaches to the database and inserts a row.

SQL> DECLARE SCHEMA FILE TEST_DB;
SQL> SET TRANSACTION READ WRITE;
SQL> INSERT INTO CANDIDATES (LAST_NAME, FIRST_NAME)
cont> VALUES ("Doe", "John");
1 row inserted
SQL> COMMIT;

- User #1 wants to delete the row
SQL> SET TRANSACTION READ WRITE;
SQL> DELETE FROM CANDIDATES;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file $2$DUA1:[DB]RDSBUGCHK.DMP;1
%SYSTEM-F-BREAK, breakpoint fault at PC=00000000, PSL=00000000
```

There is a workaround to this problem. After creating any new B-tree index defined on an empty table, store data in the table so that the index fields are used and, if you have a partitioned index, store at least one record in each partition. Once the data is stored, erase the rows. Note that any index defined on a table that has data does not have this problem.

7.2.8 Rdb/VMS Does Not Accept the Database File Specification in a Logical Name

Rdb/VMS V4.0 does not accept the database file specification in a logical name in a remote attach if the colon is missing, as shown in the following example:

```
On node A:
$ DEFINE/SYSTEM LOGICALNAME DB$DISK:[RDB]MF_PERSONNEL

On node B:
$ DIR A::LOGICALNAME:

Directory DB$DISK:[RDB]

MF_PERSONNEL.RDB;1

In SQL:
$ SQL
SQL> DECLARE SCHEMA FILENAME A::LOGICALNAME;

%RDB-E-BAD_DB_FORMAT, LOGICALNAME.RDB; does not reference a database known to Rdb
-RMS-E-FNF, file not found

In RDO:
$ RDO
RDO> INVOKE DATABASE FILENAME A::LOGICALNAME

%RDB-E-BAD_DB_FORMAT, LOGICALNAME.RDB; does not reference a database known to Rdb
-RMS-E-FNF, file not found
```

This is a known problem.

A workaround is to define your logical name with a directory specification and include the colon at the end of the logical name, as the following example shows:

```
$ DEFINE/SYSTEM LOGICALNAME DB$DISK:[RDB]

In SQL:
$ SQL
SQL> DECLARE SCHEMA FILENAME LOGICALNAME:MF_PERSONNEL;

In RDO:
$ RDO
RDO> INVOKE DATABASE FILENAME LOGICALNAME:MF_PERSONNEL
```

7.2.9 Query Optimizer Does Not Choose Index-Only Retrieval When the Dbkey Is Selected

Queries that return dbkeys, but do not reference table columns, can avoid the use of beneficial indexes. Consider the following query in which the EMPLOYEES table has a sorted index with the EMPLOYEE_ID column as the first segment:

```
FOR E IN EMPLOYEES PRINT E.RDB$DB_KEY END_FOR
```

Rdb/VMS chooses to access the EMPLOYEES table data to solve the query, even though Rdb/VMS could have solved the query by accessing only the sorted index.

In a similar query that references a table column (E.EMPLOYEE_ID) in which the EMPLOYEES table has a sorted index with the EMPLOYEE_ID column as the first segment, Rdb/VMS chooses to access the sorted index using index-only retrieval.

```
FOR E IN EMPLOYEES PRINT E.EMPLOYEE_ID,E.RDB$DB_KEY END_FOR
```

There is no suitable workaround for this problem.

The following example shows the problem.

```
$ DEFINE RDMS$DEBUG_FLAGS S
$ RDO
RDO> INVOKE DATABASE FILENAME PERSONNEL
RDO> FOR E IN EMPLOYEES
CONT>  PRINT E.RDB$DB_KEY
CONT>  END_FOR
RDO> FOR E IN EMPLOYEES
CONT>  PRINT E.EMPLOYEE_ID, E.RDB$DB_KEY
CONT>  END_FOR
```

RDMSHRP ECO 15 fixes this problem but is not part of this mandatory update for V4.0 because the patch does not change the cost information. This problem is fixed in V4.1.

7.2.10 Rdb/VMS Hangs on a SELECT Statement When a Column Data Type Is Changed from INTEGER to CHARACTER to DATE

When changing a column data type from INTEGER to CHARACTER to DATE, and then trying to select a row from the affected table, Rdb/VMS hangs.

When you store a value of zero (0) in a table with a column of INTEGER data type and then try to modify this column to a DATE data type, this operation fails (as expected) with the following error message:

```
%SQL-F-NUM_TO_DATE,  Numeric data in column INTEGER_FIELD cannot be
                     converted to a date data type.
```

When the same column is altered to a CHARACTER data type, CHAR(1), Rdb/VMS gives the following error message, and does perform the alter:

```
%SQL-W-CHR_TOO_SMA,  The string length of column INTEGER_FIELD is too small
```

When a SELECT statement is performed on the table, the desired row is returned.

When you alter the data type of the same column and make it a DATE data type, the following error is returned:

```
%SQL-W-INC_DAT_TYP,  Altering column INTEGER_FIELD to an incompatible data
                     type may cause data loss
```

Now when you perform a SELECT statement on the table, it causes Rdb/VMS to hang for this process, and does not allow anyone to attach to the database. The process must be deleted from the system to allow users to attach to the database again.

This is a known problem for V4.0. When you perform metadata changes, Rdb/VMS generates a new version number on each new row inserted. When the SELECT statement tries to read an old version, it restructures that version to the latest (current) version of the row. In this case, the restructuring tries to convert an INTEGER to DATE.

The problem described here is the result of incorrect error processing within Rdb/VMS. While trying to output a message of the following format, Rdb/VMS went into an infinite loop:

```
%RDB-F-WISH_LIST, feature has not been implemented
```

Converting from an INTEGER to a DATE data type is not defined by Rdb/VMS and should result in a conversion error.

As a workaround you should modify all the INTEGER fields stored in old rows using an UPDATE statement before altering the domain to a DATE data type.

This problem is fixed in V4.1.

7.2.11 Rdb/VMS Monitor Fails When the Last User Finishes on a Particular Database

If the Rdb/VMS monitor process, RDMMON, fails when the last user finishes from a database, an abbreviated bugcheck dump is written to the monitor log file. The exception is one of the following:

```
MON$UNLOCK_MPLL + 00000031
           or
MON$UNLOCK_MPLL + 00000036
           or
MON$UNLOCK_MPLL + 00000049
```

The secondary error message is either a SYSTEM-F-ACCVIO or SYS$SYSTEM-F-INVLOCKID.

This exception is caused by the way in which Rdb/VMS allocates virtual memory. Customers who see this problem generally have storage areas (live and snapshot) that number in the hundreds (the actual number may vary from database to database).

This is a known problem in V3.1B and V4.0. The problem will *not* cause a database to become corrupt; however, if your Rdb/VMS monitor process is failing with the errors just cited, you should apply the optional patch. This problem can be fixed in V4.0 by applying the optional patch supplied on the mandatory update kit. See Section 8.1.2 for more information.

7.2.12 Multisegmented Index Is Not Selected When a Not-Equal Predicate Is Specified

A query that uses the not-equal predicate referring to a column within the most selective index may cause the query optimizer to chose a different index. When more than one useful multisegmented index exists, the query optimizer chooses a less selective index. This occurs when the more selective index has a not-equal (<>) predicate on its non-first segment. For example, assume MY_TABLE has columns A, B, C, D, E, F with two 3-segment indexes: INDEX_A_B_C on columns A, B, C and INDEX_A_D_E on columns A, D, E. For the following query a less selective index INDEX_A_D_E is used:

```
SELECT * FROM MY_TABLE
WHERE A = 'A' AND B = 'B' AND C <> 'C';

Leaf#01 FFirst MY_TABLE Card=100
  BgrNdx1 INDEX_A_D_E [1:1]
```

Although the index INDEX_A_B_C is a better index, it is not used because a not-equal predicate (C <> 'C') appears on the third segment of this index. This problem is fixed in V4.1.

A possible workaround for this problem is to modify the query to replace the not-equal predicate with a Boolean expression. For example, in the previous query the expression AND C <> 'C' can be replaced with AND ((C < 'C') OR (C > 'C')).

This is a known problem in V4.0. This problem can be fixed in V4.0 by applying the optional patch supplied on the mandatory update kit. See Section 8.1.3 for more information.

7.2.13 Triggers That Affect Subject Table Rows Can Cause Loops or Inconsistent Results

Triggers that update rows of the trigger subject table or add rows to the trigger subject table can cause infinite loops or inconsistent results to be returned. For example, consider the following two conditions:

- A BEFORE UPDATE trigger on table X that inserts a row into table X
- An UPDATE statement affecting all the rows in table X

Given these two conditions, the UPDATE statement will loop until all resources are consumed because for each row updated, a new row will be added, which in turn will be updated, and so forth.

If subject table rows are being retrieved using an index, a triggered action operating on the same table could affect the index (by changing index key values or adding new keys) such that the triggering statement behaves in a different manner than when no trigger is involved.

To avoid this problem, construct any such triggers to operate only on rows that are either the current subject table row, or that will never be selected by the triggering statement. A more difficult avoidance method is to restructure triggering statements so that they never select a row that could have been updated or added by a trigger action. Some circumstances will require a combination of these methods.

7.2.14 Singleton Subselect Statement Returns Incorrect Results

The following query in an SQL precompiled C program shows an Rdb/VMS problem in which a query involving a singleton subselect can return incorrect results.

The correct result from interactive SQL is as follows:

```
SELECT Y.A,Y.C,(SELECT SUM(X.B) FROM X WHERE X.A=1) FROM Y WHERE Y.A=1;
           A             C
           1            10                      3
1 row selected
```

The result from the precompiled C program is the following second sum, which is incorrect:

```
 SQLCODE = 0
----> sum = 3
 SQLCODE = 0
----> sum = 0 (incorrect result)
```

A workaround is to store the result of the subselect in a temporary variable as is done in the first query in the precompiled C program shown in the following example. The second query in this precompiled C program shows the problem.

```
CREATE SCHEMA FILENAME PROBLEM;
CREATE TABLE X (A INTEGER, B INTEGER);
CREATE TABLE Y (A INTEGER, C INTEGER);
INSERT INTO X VALUES (1,1);
INSERT INTO X VALUES (1,2);
INSERT INTO X VALUES (2,1);
INSERT INTO Y VALUES (1,10);
INSERT INTO Y VALUES (2,20);
COMMIT;
SELECT Y.A, Y.C, (SELECT SUM(X.B) FROM X WHERE X.A = 1) FROM Y WHERE Y.A = 1;
```

```
---------------------- Precompiled C Program To Reproduce
----------------------
#include <stdio.h>
#define check_sqlcode printf(" SQLCODE = %d\n", SQLCODE)
main()
{       int SQLCODE, a, c, sum;
        /* This query works */
exec sql declare schema filename problem;
exec sql select sum(x.b) into :sum from x where x.a = 1;
        check_sqlcode;
        printf("----> sum = %d\n", sum);
        /* This query returns the wrong result */
exec sql select y.a, y.c, (select sum(x.b) from x where x.a = 1)
        into :a, :c, :sum from y where y.a = 1;
        check_sqlcode;
        printf("----> sum = %d\n", sum);}
```

This is a known problem in V4.0. This problem can be fixed in V4.0 by applying the optional patch supplied on the mandatory update kit. See Section 8.1.4 for more information.

7.2.15 Query with a FOR Loop with a MODIFY Statement Followed by a PRINT Statement Can Return Incorrect Results

The following query with a FOR loop with a MODIFY statement followed by a PRINT statement can return incorrect results as follows:

```
FOR E IN R1 CROSS F IN R2 WITH (E.UCTO_BH = F.UCTO_B)
  SORTED BY E.CION
  MODIFY E USING E.CAP = '1'  END_MODIFY
  PRINT E.CION,E.UCTO_BH,F.UCTO_B END_FOR
```

The following results are incorrect.

```
 119          PRO003          PRO003
 119          PRO003          PRO003
 119          PRO003          PRO003
 119          PRO003          PRO003
 :       :       :
 :  63 TIMES THE SAME RECORD IS RETURNED INSTEAD OF THE 63 RECORDS ****
 :       :       :
```

The following results are correct.

```
 101          PRO001          PRO001
 102          PRO001          PRO001
 103          PRO001          PRO001
 104          PRO002          PRO002
```

7.2.16 Query with a Computed-By Field and OR Logic Returns Incorrect Results

The following query returns incorrect results:

```
FOR R IN R2 WITH (R.M_OVED="0023" AND R.YYYYMM="199101") OR
        (R.M_KARTIS="0018" AND R.YYYYMM >= "199101") PRINT R.* END_FOR
```

The yyyymm field is a computed-by field.

If the query is slightly reworded as follows, it produces the correct results:

```
FOR R IN R2 WITH (R.M_KARTIS="0018" AND R.YYYYMM >= "199101") OR
        (R.M_OVED="0023" AND R.YYYYMM="199101") PRINT R.* END_FOR
```

This is a known problem in V3.1B and V4.0. The problem was caused by the incorrect optimization of reusing the result of evaluation of a common subquery (in this case, the computed-by expression used in both OR legs.)

This problem is fixed in V4.1.

7.2.17 Defining a View Causes a Bugcheck When a Sorted Index Was Previously Defined

When defining a view in interactive SQL, the following bugcheck is generated when a sorted multisegmented index was previously defined.

```
*****Exception at  RDMS$$RSS$ASN_FOR_RSS_NDX
                   fatal unexpected error detected
```

The view definition is as follows:

```
CREATE VIEW TEST_VIEW
      AS SELECT
            A.ASSET_ID,
            A.ASSET_DSC
      FROM ASSETS A, VEHICLES V, BUILDINGS B
      WHERE
      A.ASSET_TYPE_CD = "V" AND
      A.ASSET_ID      = V.ASSET_ID
      OR
      A.ASSET_TYPE_CD = "B" AND
      A.ASSET_ID      = B.ASSET_ID;
```

The sorted multisegmented index is as follows:

```
CREATE INDEX ASETS_KEY_NDX ON ASSETS
(ASSET_ID, ASSET_TYPE_CD)
 TYPE IS SORTED;
```

This is a known problem in V4.0. This is a problem in the optimization of certain queries containing OR logic. During optimization, the wrong strategy may be chosen when trying to optimize OR logic, resulting in incorrect assumptions about the order in which record retrieval operations are to be carried out.

A workaround is to delete the indexes before creating the view and then create the indexes again. Another workaround is to change the selection clause within the view as follows:

```
SELECT
       A.ASSET_ID,
       A.ASSET_DSC
 FROM ASSETS A
 WHERE
 A.ASSET_TYPE_CD = "V" AND
 EXISTS ( SELECT V.ASSET_ID FROM VEHICLES V WHERE A.ASSET_ID = V.ASSET_ID)
 OR
 A.ASSET_TYPE_CD = "B" AND
 EXISTS ( SELECT B.ASSET_ID FROM BUILDINGS B WHERE A.ASSET_ID = B.ASSET_ID)
```

This problem is fixed in V4.1.

7.2.18 Problem When Database Is Defined as Remote

When a database is defined as remote, the database is attached and activity occurs on the remote node as determined by running the RMU/SHOW STATISTICS command. However, when the remote node finishes, the following error message is returned:

```
%RDB-F-SYS_REQUEST, error from system services request
-LIB-F-BADBLOADR, bad block address
```

The process on the local node is running in a loop and must be stopped using a Ctrl/Y.

The program runs well in a local environment but returns an error in a remote environment.

This is a known problem. There is no workaround to this problem.

7.2.19 An Incompatible Change for RDO Applications: New Update Rules Will Be Enforced by Default in V4.1

Starting in Rdb/VMS V4.1, new update rules will be enforced by default. With new update rules, it will no longer be possible to modify or delete rows from a table that is directly joined with other tables. However, rows from a table can still be modified or deleted if the table is joined with other tables that are in a subquery. Because SQL does not have syntax that allows rows from a table to be modified or deleted and concurrently allow that table to be joined with other tables, these rules will have no affect on SQL applications. However, those RDO applications that contain join update queries (the update queries that modify or delete rows from a table that is joined with other tables) will be affected and will have to be fixed.

Rdb/VMS V4.1 will give an error diagnostic for the following join update query:

```
FOR E IN EMPLOYEES CROSS D IN DEGREES OVER EMPLOYEE_ID
     WITH D.DEGREE = 'MA'
   ERASE E
END_FOR

%RDMS-E-JOIN_CTX_UPD, relation EMPLOYEES is part of a join, cannot be updated
```

In this previous update query if an employee has two MA degrees, the same employee row would be joined to two different degree rows. Therefore, Rdb/VMS will try to delete the same row twice. Or, if instead of using the ERASE verb, the previous update query used a MODIFY verb on the EMPLOYEES table, then Rdb/VMS might modify the row more than once.

In versions of Rdb/VMS prior to V4.1, join update queries similar to the previous query worked correctly, produced an error diagnostic trying to delete the same row more than once, or, even worse, produced a bugcheck.

The previous update query can be reworded as follows:

```
FOR E IN EMPLOYEES WITH (ANY D IN DEGREES WITH D.EMPLOYEE_ID = E.EMPLOYEE_ID
                              AND D.DEGREE = 'MA')
   ERASE E
END_FOR
```

Because the EMPLOYEES table is no longer directly joined to the DEGREES table, in V4.1, the rows can be erased. The modified update query guarantees that an employee row will not be deleted more than once.

To ease the transition for applications that depend on the old update rules, Rdb/VMS will support either new or old update rules in V4.1, and in the next version after V4.1. By default, Rdb/VMS will enforce new update rules. To continue to use the old update rules, you will be able to define the following logical name, RDMS$USE_OLD_UPDATE_RULES to ''1'' in V4.1. The support of both new and old update rules allows users two Rdb/VMS versions to change their applications (if necessary) to conform to the new update rules.

After these two releases, the support for old update rules will be removed from Rdb/VMS.

The following shows another example of how to change a join update query so that it works with the new update rules:

```
! Give a 10% salary raise to all managers who have an MA degree.
FOR S IN SALARY_HISTORY CROSS D IN DEGREES CROSS DP IN DEPARTMENTS
   WITH S.EMPLOYEE_ID = D.EMPLOYEE_ID AND
        S.EMPLOYEE_ID = DP.MANAGER_ID AND
        S.SALARY_END MISSING          AND
        D.DEGREE = 'MA'
   MODIFY S USING S.SALARY_AMOUNT = S.SALARY_AMOUNT * 1.1
END_FOR
```

The previous join update query will not work with the new update rules. Also this update query modifies some salary history rows more than once and gives multiple salary raises to some managers! This query can be reworded using a subquery as follows:

```
FOR S IN SALARY_HISTORY
   WITH S.SALARY_END MISSING AND
        (ANY D IN DEGREES CROSS DP IN DEPARTMENTS
         WITH S.EMPLOYEE_ID = D.EMPLOYEE_ID AND
         S.EMPLOYEE_ID = DP.MANAGER_ID AND
         D.DEGREE = 'MA')
   MODIFY S USING S.SALARY_AMOUNT = S.SALARY_AMOUNT * 1.1
END_FOR
```

This latter query will work with the new as well as the old update rules; it will ensure that each qualified manager gets a single salary raise.

7.2.20 Relation Name Must Match Dictionary Record Name

If you define a relation using a CDD/Plus path name, the relation name must match the record name. For example, the following statement contains an error. (The statement will be processed; however, problems will occur later.)

```
DEFINE RELATION AAAA FROM PATHNAME "CDD$TOP.TEST.XXXX" END.
```

The correct form of the statement is as follows:

```
DEFINE RELATION AAAA FROM PATHNAME "CDD$TOP.TEST.AAAA" END.
```

This is a known restriction that was first documented in the V3.0 *VAX Rdb/VMS Release Notes*.

7.2.21 NOWAIT Transactions Have Their Buffers Invalidated at COMMIT

Programs that use NOWAIT transactions have their buffers invalidated at commit time. This forces Rdb/VMS to read the data again, as can be observed by higher than expected DIO rates. This is a known problem in V3.1B and V4.0.

A workaround is to use WAIT transactions.

7.3 SQL Problems, Restrictions, and Notes

This section describes problems, restrictions, and other information of interest to users of the SQL interface.

See also the Restrictions chapter of the V4.0 *VAX Rdb/VMS Release Notes* for other restrictions that still apply.

7.3.1 SQL Deprecated Features and Incompatible Changes for VAX Rdb/VMS Version 4.1

Portions of the SQL language are changing for Version 4.1. The changes are required for SQL conformance to the ANSI/ISO standard and to the standard emerging from ANSI SQL2. Where possible, the current language syntax and semantics are preserved and merely deprecated; however, in a few cases, incompatible changes will be made to achieve ANSI conformance. The types of changes envisioned for Version 4.1 include:

- Deprecated feature

  Syntax that is marked as **deprecated** will work as it did previously but its usage may not be supported in a future Rdb/VMS release.

- Incompatible change

  A change to syntax, referred to as an **incompatible change**, identifies syntax that does not work as it did in the previous version or does not work at all.

In anticipation of these changes, Rdb/VMS Version 4.0 deprecated some SQL language features; however, some language features deprecated in Rdb/VMS Version 4.0 will be changed in a future version of Rdb/VMS. You can make changes to your applications now to ease the transition. The following list describes the language features that will change in a future version of Rdb/VMS and suggests how to minimize application impact:

- Schemas will be objects within an Rdb/VMS database.

  Currently, an Rdb/VMS database is called a schema in SQL. To make SQL fully ANSI/ISO compliant, schemas will become objects within an Rdb/VMS database. Version 4.1 will correctly interpret both existing SQL applications and ANSI-compliant applications in all but two cases:

  1. In the first case, an application contains a CREATE SCHEMA statement that uses ANSI-compliant syntax, but expects an Rdb/VMS database to be created. You can resolve this ambiguity by adding an Rdb/VMS specific clause (such as FILENAME) to the CREATE SCHEMA statement. For example, your module might contain the following procedure:

     ```
     PROCEDURE create_my_schema
         SQLCODE
         CREATE SCHEMA AUTHORIZATION my_auth
             CREATE TABLE T1 (A INT);
     ```

     In Rdb/VMS Version 4.0, this statement creates an Rdb/VMS database. In Version 4.1, this statement will create a schema within a database. To resolve this ambiguity, change the procedure as follows:

     ```
     PROCEDURE create_my_schema
         SQLCODE
         CREATE SCHEMA AUTHORIZATION my_auth
             FILENAME my_auth
             CREATE TABLE T1 (A INT);
     ```

  2. In the second case an application contains a DROP SCHEMA AUTHORIZATION statement. You must change the DROP SCHEMA AUTHORIZATION statement to either DROP SCHEMA PATHNAME or DROP SCHEMA FILENAME. SQL will correctly interpret your changed applications in both Rdb/VMS Version 4.0 and in Version 4.1.

- Use single quotes (') instead of double quotes (") for string literals.

  In Rdb/VMS Version 4.0, the use of the double quote character for string literals is deprecated. ANSI/ISO dictates that string literals be formed using the single quote character. The ANSI/ISO standard does not specify the double quote character but the ANSI SQL2 draft standard *does* use the double quote character for identifiers. In Version 4.1, SQL will implement both the ANSI SQL2 draft standard quoting rules and the quoting rules supported by Rdb/VMS Version 4.0 and earlier. Users will be able to choose which quoting rules to use, with the default following Version 4.1 quoting rules of using single quotes.

  Rdb/VMS SQL is moving customers toward the ANSI SQL2 draft standard quoting rules. The quoting default will most likely change to the ANSI SQL2 draft standard rules in a future release after Version 4.1. To smooth this transition, Digital recommends that you develop all new applications using the single quote character with character string literals and the double quote character with identifiers.

- DROP TABLE will default to RESTRICT semantics.

  Currently, the DROP TABLE statement first drops all entities dependent on the table. The keywords CASCADE and RESTRICT are optional qualifiers to the DROP TABLE statement in Rdb/VMS Version 4.0. With the CASCADE qualifier, all dependent entities are dropped along with the table. With the RESTRICT qualifier, no entities are dependent on the table.

  In Rdb/VMS Version 4.0, if neither CASCADE nor RESTRICT is specified, the DROP TABLE statement is interpreted as DROP TABLE CASCADE. This is consistent with previous versions of SQL. In Version 4.1, the interpretation will be changed to DROP TABLE RESTRICT. You can make applications containing DROP TABLE statements upwardly compatible by adding the CASCADE keyword to their DROP TABLE statements.

7.3.2 SQL to Support Error Code Values in Rdb/VMS Version 4.1

In Rdb/VMS Version 4.1, SQL will add error code values. Some of these values describe specific error conditions with no specific SQLCODE value in Rdb/VMS Version 4.0 and earlier. If your programs test for SQLCODE –1 as a possible error value, they may not correctly trap error conditions.

The *VAX Rdb/VMS Guide to Using SQL* states that values that are less than zero are error values. Therefore, applications should check for SQLCODE less than zero rather than equal to –1 for error status.

Using this test ensures that your programs will always be upwardly compatible between SQL versions.

7.3.3 Using the IGNORE CASE Option of the LIKE Clause Sometimes Results in a Query That Incorrectly Returns No Rows

In certain SELECT statements in SQL, the LIKE clause works correctly unless an IGNORE CASE option is added to it. The IGNORE CASE option causes the query to return no rows. This seems to occur when the WHERE clause involves more than one condition connected with an AND Boolean operator.

The following command procedure illustrates this problem. The first SELECT statement correctly returns the inserted record. Adding an IGNORE CASE option to one of the items in the WHERE clause results in the return of no rows. Adding another IGNORE CASE option to a different item in the WHERE clause seems to fix the problem.

```
$ SQL
DECLARE SCHEMA FILE PERSONNEL;
CREATE TABLE XXX (F1 CHAR(5), F2 CHAR(3), F3 CHAR(3));
INSERT INTO XXX (F1, F2, F3) VALUES ("ABC", "ABC", "ABC");
SELECT * FROM XXX WHERE
      F1 LIKE "AB%" AND
      F2 LIKE "ABC" AND
      F3 LIKE "ABC";
SELECT * FROM XXX WHERE
      F1 LIKE "AB%" AND
      F2 LIKE "ABC" IGNORE CASE AND
      F3 LIKE "ABC";
SELECT * FROM XXX WHERE
      F1 LIKE "AB%" AND
      F2 LIKE "ABC" IGNORE CASE AND
      F3 LIKE "ABC" IGNORE CASE;
ROLLBACK;
```

This is a known problem in V3.1, V3.1A, V3.1B, and V4.0.

A workaround is to add additional IGNORE CASE clauses to the WHERE clause as previously mentioned, as this seems to correct this problem.

7.3.4 An SQL SELECT Statement Results in an Invalid BLR Error

The following SQL SELECT statement causes an invalid BLR error:

```
SELECT ( SELECT GRP2.FIRST_NAME FROM EMPLOYEES GRP2 WHERE
                GRP2.EMPLOYEE_ID = SST.EMPLOYEE_ID AND
                GRP2.SEX = "m" )  FROM DEGREES SST ORDER BY 1;
```

Ths invalid BLR is as follows:

```
%RDB-E-INVALID_BLR, request BLR is incorrect at offset 142
```

This is a known problem. If the inner SELECT statement is an aggregate then the BLR is correctly parsed.

This problem is fixed in V4.1.

7.4 SQL/Services Problems, Restrictions, and Notes

This section describes problems, restrictions, and other information of interest to users of SQL/Services.

See also the Restrictions chapter of the V4.0 *VAX Rdb/VMS Release Notes* for other restrictions that still apply.

7.4.1 SQL/Services VMS API Shipped with the Rdb/VMS Run-Time Kit

In the mandatory update for Version 4.0, the SQL/Services VMS API is included in the Rdb/VMS run-time kit. With the Rdb/VMS run-time kit installed, you can execute previously developed SQL/Services client applications; however, you cannot develop SQL/Services client applications. Development of SQL/Services client applications is possible only with the Rdb/VMS full development kit.

7.4.2 VMS API Installation Without Rdb/VMS

In Rdb/VMS Version 4.0, the VMS API IVP fails when building the SQL/Services IVP executables because the linking operation cannot find the SQL$INT.EXE and RDBSHR.EXE images.

In the mandatory update for V4.0, you can correct this problem by copying both the SQL$INT.EXE and RDBSHR.EXE images from a system running the mandatory update for Rdb/VMS Version 4.0 and SQL/Services to the target installation system's SYS$LIBRARY directory.

The images are required for IVP linking only. SQL/Services does not access the files during execution.

7.4.3 Trailing Characters on SQL/Services Sample Program Error Messages

In Rdb/VMS Version 4.0, the SQL/Services sample program occasionally displays trailing characters at the end of error messages. Terminating ASCII_STRING data type values with a NULL character causes this error to occur.

This problem will be fixed in a future release of Rdb/VMS.

7.5 RDO, RDBPRE, and RDML Problems, Restrictions, and Notes

This section describes problems, restrictions, and other information of interest to users of the RDO, RDBPRE, and RDML interfaces.

See also the Restrictions chapter of the V4.0 *VAX Rdb/VMS Release Notes* for other restrictions that still apply to users of RDO, RDBPRE, and RDML.

7.5.1 RDO IMPORT Does Not Save All SQL Defined Attributes

An RDO IMPORT operation does not save all SQL defined attributes, such as the default-value defined for columns. Use the SQL IMPORT statement. This is a problem for run-time only (RTO) systems. There is no workaround to this problem for RTO systems.

7.5.2 RDO CONVERT on V3.0 Databases Causes Database Corruption When the Database Is Converted to V4.0

The RDO CONVERT operation for V3.0x reorders the fields of some system relations so they no longer have the correct on-disk structure. Consequently, if you try to integrate the database into the data dictionary (CDD/Plus), export it, or do a SHOW TABLE RDB$FIELDS statement, a bugcheck results with the following exception:

```
PIOFETCH$WITHIN_DB + 142
```

Because normal DML access to these relations continues to work, the corruption usually is not noticed with V3.0x or even if the database is converted to V3.1x. However, the corruption causes the database to be incompatible with V4.0. After using the RMU/CONVERT command to convert the V3.0x or V3.1 database to V4.0, you can no longer attach to the database.

The solution is to perform an EXPORT/IMPORT operation after converting to V3.0x and before converting to V4.0. Alternately, the database can be converted by using an EXPORT/IMPORT operation rather than by using the RMU/CONVERT command.

Only databases converted from V2.x to V3.0x using the RDO CONVERT statement and never imported since the conversion are affected.

7.6 Rdb/VMS Management Utility (RMU) Problems, Restrictions, and Notes

This section contains problems, restrictions, and other notes that pertain to the Rdb/VMS Management Utility (RMU).

See also the Restrictions chapter of the V4.0 *VAX Rdb/VMS Release Notes* for other restrictions that still apply.

7.6.1 Do Not Delete After-Image Journal (.AIJ) Backup Files If the AIJ Backup Fails or Is Terminated

If an AIJ backup process fails or is terminated prematurely, one action the user might take would be to discard the resultant .AIJ backup file because the backup operation was not completed. However, all .AIJ backup files, including those produced by a failed backup process, are necessary to recover a database. If an .AIJ backup file of a failed backup process was discarded, the database would not be recoverable from that point forward. This is especially important if you use magnetic tapes as the AIJ backup media; in this case, preserve this magnetic tape and do *not* reuse it.

When an AIJ backup process, especially one running in continuous (/CONTINUOUS) mode, writes to the .AIJ backup file, it is possible for the transferred data to be deleted from the database .AIJ file. If the backup process subsequently fails or is prematurely terminated (for example with Ctrl/Y or the STOP command), it might not be possible to retransfer the data to the subsequent .AIJ backup file because the data was deleted from the active database .AIJ file.

Therefore, it is extremely important that you preserve *all* .AIJ backup files, even those produced by failed or terminated backup processes. If the resultant .AIJ backup file is discarded, the next .AIJ backup file could contain a ''gap'' in transactions, so that no transactions would ever be rolled forward from that point on.

Note

If this problem occurs, the database is *not* inconsistent or corrupt. Rather, the database cannot be rolled forward past the discarded .AIJ backup file.

The solution to this problem is to preserve all .AIJ backup files to ensure that a database can be completely recovered.

If you have discarded an .AIJ backup file, perform a complete database backup immediately to ensure that the database can be recovered up to the current transaction.

7.6.2 Concealed Logicals Are Supported but No Longer Recommended for Use After V4.0

For Rdb/VMS V3.1 and earlier, some maintenance operations either needed the use of concealed logical names or were simplified or more efficient when they were used. So the use of concealed logical names was recommended and supported.

With V4.0 and higher versions of Rdb/VMS, RMU provides an alternate effective means to perform the same maintenance operations without concealed logical names. Concealed logical names are an additional complication, and their use encourages the use of VMS file utilities. Both of these factors may lead to

problems with database maintenance operations. Therefore, use of concealed logical names for database files is no longer recommended, although it is supported. The use of VMS file utilities on database files, however, is no longer supported if RMU provides an alternative. So the use of concealed logical names is redundant, and should be discouraged for new databases.

Therefore, the use of concealed logical names *is* supported, but is *not* recommended after V4.0.

7.6.3 Warnings from an RMU/VERIFY Operation

Please ignore the following warning messages if you receive them from an RMU/VERIFY operation:

```
%RMU-W-AIPENTMBZ, entry (n) in area inventory page (m) has never
                  been used, but is not empty. It should contain all zeroes

%RMU-W-PGSPAMENT, area (n), page (m)
                  the fullness value for this data page does not match
                  the threshold value in the space management page
```

These are known problems. Neither problem affects the integrity of the database.

7.6.4 RMU/VERIFY/INDEX or RMU/VERIFY/ALL Command Causes a Bugcheck If You Have Hashed Indexes Defined

The RMU/VERIFY/INDEX or the RMU/VERIFY/ALL command may return a bugcheck with the following exceptions if you have hashed indexes defined:

```
***** Exception at 0003F399 : BUGCHKAST$BUGCHECK + 000004D0
and
RMUVERNDX$VFY_ONE_PAGE_HINDEX
```

This is a known problem with the RMU/VERIFY command, not with hashed indexes. The problem is fixed in V4.1.

7.7 Notes and Restrictions Related to CDD/Plus

This section describes problems and restrictions relating to the use of Rdb/VMS with CDD/Plus.

See also the Restrictions chapter of the V4.0 *VAX Rdb/VMS Release Notes* for other restrictions that still apply.

7.7.1 Restrictions Lifted by CDD/Plus Version 4.3

The restrictions listed in the Rdb/VMS V4.0 Release Notes, Section 3.8, have been lifted by VAX CDD/Plus Version 4.3.

7.8 DECtrace Problems, Restrictions, and Notes

This section describes problems and restrictions relating to the use of Rdb/VMS with DECtrace.

7.8.1 Rdb/VMS Version Number Used for DECtrace Will Remain at V4.0

The Rdb/VMS version number used for DECtrace will remain at V4.0. After installing the mandatory update for Rdb/VMS V4.0, use the DECtrace definition from the Rdb/VMS V4.0 kit for manual insertion into the DECtrace library.

7.9 Rdb/VMS Documentation Errors and Omissions in V4.0

This section describes errors or omissions in the Rdb/VMS manuals and documents.

7.9.1 Buffer Management Changes for V4.0

In Rdb/VMS V3.1, one queue of buffer control blocks was maintained in a least recently used (LRU) fashion. The least recently used buffer in the queue (the one at the tail) was chosen as the buffer to be used to read another buffer of pages, whether or not the buffer was marked.

One disadvantage to this scheme was the need to do a synchronous write operation when a marked (updated) buffer had to be cleaned out to read a new buffer. Synchronous write operations force the disk head to move to the appropriate cylinder on disk and also make the user process stall during the write operation. Asynchronous write operations are better than synchronous write operations because the user process is not stalled. In addition, if a number of write operations are issued asynchronously at the same time, Rdb/VMS reduces the disk head movement by performing the write operations in the optimal order. Because the time needed to move the disk head is the most significant expense in an I/O operation, this factor is important. For these reasons, the buffer management scheme was changed in Rdb/VMS V4.0.

Rdb/VMS V4.0 has two queues of buffer control blocks. One queue corresponds to a set of marked buffers and the other queue corresponds to a set of unmarked buffers. Whenever a buffer is marked (updated), it is moved to the head of the marked queue. Similarly, whenever a buffer is unmarked (written to disk), it is moved to the head of the unmarked queue. Therefore, each queue is managed in an LRU fashion, but there is no global LRU function for the entire buffer pool.

Whenever a buffer is to be chosen, the unmarked queue is searched first from the end (in an LRU fashion) and then the marked queue is searched. By postponing the write operations, Rdb/VMS can gather a good batch and flush the batch asynchronously at commit time. Although this change usually removes the disadvantages previously described for V3.1, occasionally the problem remains (see Section 6.2.14).

The following examples demonstrate situations in which this two-buffer queue management system improves performance and situations in which it does not. For these examples, assume that there is only one page per buffer.

The first user is updating 50 pages, and is running an application with 100 buffers. In V3.1, this could have resulted in 50 synchronous write operations. The user would have stalled 50 times, for the duration of 50 I/O operations in total. In V4.0, the 50 marked buffers remain unmarked until the end of the transaction, when they are written in one batch during commit processing. Because the batch write operation lets Rdb/VMS optimize the write operations in V4.0, the user stalls only once for a total time corresponding to approximately 10 buffers. The two-buffer queue management system provided with V4.0 improves performance for this user.

The second user is also running an application with 100 buffers. This user is updating records on 100 pages, and then reading the records repeatedly. In V3.1, this could have resulted in 100 synchronous write operations. However, because the 50 pages are soon cached in the buffer pool, they involve no read operations. In V4.0, after the 100 pages are updated, all the buffers are marked. Then for every read operation, only one buffer is used. The two-buffer queue management

system drastically increases the number of data file read operations and adversely affects performance for this user.

To clarify the explanation, consider another case. A third user is updating 100 pages and then reading 100 pages. In V3.1, this would have resulted in 100 synchronous write operations and then 100 read operations. In V4.0, after the first 100 update operations, all buffers are marked. Then Rdb/VMS uses only one buffer to read the 100 buffers. Thus, V4.0 still performs 100 read I/O operations, but submits the 100 asynchronous write operations as a batch at commit time. The two-buffer queue management system improves performance for this user.

In general, the two-queue buffer management system improves overall system performance and response time for users. However, the performance will be worse when *both* of the following conditions are true:

- When the number of pages being updated is greater than the number of buffers
- When a number of pages are read repeatedly after being updated

There are also more complicated scenarios in which this two-queue buffer management scheme can result in poorer performance. Whether your application is affected adversely by the two-queue buffer management scheme depends on the pattern of read and write operations to the data pages.

The mandatory update for V4.0 improves the poor performance of the V4.0 two-queue buffer management scheme (see Section 6.2.14). However, performance is still slower than it was under V3.1; this is fixed in V4.1.

7.9.2 Incorrect Reference in V4.0 *VAX Rdb/VMS SQL Reference Manual*, Chapter 3

There is an incorrect reference in the V4.0 *VAX Rdb/VMS SQL Reference Manual*. On page 3–46, the last paragraph before the note says to see Table 6–8 for more information about the LIB$DT_INPUT_FORMAT logical name.

Table 6–8 does not contain any information about LIB$DT_INPUT_FORMAT. You should refer to the *VMS RTL Library (LIB$) Manual* for the appropriate information.

7.9.3 Printing Error in V4.0 *VAX Rdb/VMS SQL Reference Manual*, Chapter 4

On page 4–24 of the *VAX Rdb/VMS SQL Reference Manual*, the following text, which should have preceded Table 4–2, was omitted:

```
Table 4-2 shows the VMS data types that SQL requires for actual
parameters when you declare formal parameters for each SQL data type.
```

7.9.4 Documentation Error in V4.0 *VAX Rdb/VMS SQL Reference Manual*, Appendix D.4

In Appendix D.4, Table D–2, in the *VAX Rdb/VMS SQL Reference Manual*, footnote one states that, ''If the parameter marker or select list item data type allows null values, the SQLTYPE field value returned is the base value plus one.'' This statement is in error. All versions of the Rdb/VMS SQL interface since and including VAX SQL V1.0 have not included the functionality stated in footnote one. Such a feature has never been supported in any released version of the SQL interface for Rdb/VMS.

The documentation for V4.1 no longer contains this footnote.

The only known solution for finding the state of the null vector is to query the metadata and then set a program flag that indicates the 'null value allowed' state.

7.9.5 SQL/Services Error Documentation

You can find SQL/Services client and server error documentation in a number of locations:

- Client errors

  Client errors are documented in the *VAX Rdb/VMS Guide to Using SQL/Services*. For Rdb/VMS errors (SQLSRV error –1), look in the Rdb/VMS documentation. For network errors (SQLSRV –2003), see the appropriate platform-specific documentation. Table 7–1 contains a partial list of error information on a platform-specific basis.

  Before listing these file names, look at your own platform-specific documentation for more information on the secondary error code resulting from a network error.

Table 7–1 SQL/Services Network Errors

Platform	File Location
VMS	sys$library:ssdef.h
DOS	\decnet\src\errno.h
ULTRIX	/usr/include/errno.h
MPW	MPW:Interfaces:CIncludes:CMIntf.h

- Server errors

  VMS server errors are logged in the following locations:

  – The SYS$MANAGER:SQLSRV$.LOG file contains communication server error information.

  – The default directory of the SQLSRV$SRV account contains execution server log files, which are named SQLSRV$RUNEXE.LOG.

7.10 SQL/Services Troubleshooting Suggestions

This section describes the SQL/Services installation failures most commonly observed during previous Rdb/VMS installations and offers a solution for each failure. Errors resulting from these failures, however, are not exclusively installation errors but general SQL/Services errors that can arise under a variety of other circumstances. This section also describes incompatibilities among various versions of SQL/Services.

7.10.1 Common SQL/Services Network Errors

This section describes a set of secondary status codes that are displayed when the SQL/Services IVP fails with a –2003 error, indicating an error that is issued from the network. SQL/Services displays the status codes placed in the SQLCA SQLERRD[0] field in a message like the one that follows:

```
SQLCA: SQLCODE: -2003 SQLERRD[0]: error-status-code SQLERRD[2]0
```

The following list includes a set of error status codes and a description of how to correct each error condition:

- 8356 - SS$_NOSUCHOBJ

  The SQL/Services communications server is not active.

  Invoke the SQL/Services startup command procedure on the server system to restart the SQL/Services server:

  ```
  $ @SYS$STARTUP:SQLSRV$STARTUP
  ```

  Refer to Section 6.5.8 for information about the changes that have been made to the SQL/Services startup command procedure for the mandatory update for Rdb/VMS V4.0.

  After starting SQL/Services, issue a DCL SHOW SYSTEM command to see if the communication server is active. If the SQLSRV$SERVER process does not exist, the communication server encountered a problem during startup.

  The SYS$MANAGER:SQLSRV$.LOG file contains the error message text describing why the communication server did not start. Send this file to your local support center if the error does not directly identify the problem.

- 8436 - SS$_LINKEXIT
  8348 - SS$_INVLOGIN

  The SQL/Services IVP for Rdb/VMS Version 4.0 fails with either the 8436 - SS$_LINKEXIT or 8348 - SS$_INVLOGIN status code if SQLSRV is defined as a DECnet object.

  With Rdb/VMS Version 4.0, the SQL/Services communication server automatically declares itself as a network object. This differs from the approach taken in Rdb/VMS Version 3.1, where the SQL$STARTUP.COM procedure defines SQLSRV as a DECnet object.

  To correct this condition, delete the DECnet object from the DECnet database by entering the following command at the VMS Network Control Program (NCP) prompt:

  ```
  NCP> CLEAR OBJECT SQLSRV ALL
  ```

7.10.2 Common SQL/Services Fatal Execution Server Errors

The SQL/Services IVP for Rdb/VMS Version 4.0 fails with the following error message:

```
VAX SQL Services sample program failed

SQLCA: SQLCODE: 61312708  SQLERRD[0]: 0 SQLERRD[2] 0
SQLERRM.SQLERRMC %SQLSRV-F-FTLEXEERR,
Fatal execute server error,
sqlsrv$runexe.log contains more information
```

The communication server was not able to start an execution server process successfully. A log file, SQLSRV$RUNEXE.LOG, is created within the SQLSRV$SRV account's default directory for every execution server. This log file contains the error text that caused the execution server process to fail.

A majority of these failures are caused by incorrect values for the SQLSRV$SRV account. (All execution server processes are started within the context of the SQLSRV$SRV VMS account.) Examples of incorrect account values are:

- The account's password has already expired.
- The default device or directory is incorrect.

- The process account ENQLM quota value is set too low. Set this parameter to 1000, and try the IVP again after shutting down the server and restarting it.
- The process account PGFLQUO quota value is too low. Set this parameter to 25000 and try the IVP again after shutting down the server and restarting it.

You may need to experiment further with the ENQLM and PGFLQUO quota values. Make sure to shut down and then restart the server each time you change either value.

Use the VMS Authorize utility to modify the SQLSRV$SRV account if an incorrect value for the SQLSRV$SRV account is identified. Send the SQLSRV$RUNEXE.LOG file to your local support center if the error in the log file does not help identify the problem.

7.10.3 Common SQL/Services API Installation Failures

This section provides a list of common problems encountered during SQL/Services client Application Programming Interface (API) installations. The list suggests a solution or refers you to a section in this V4.0 mandatory update document that contains a full description of the problem.

- A common installation failure that affects all API installations

  Default DECnet access must be enabled on the server.

  If the initial transfer of the client installation file fails, consult with your system manager to ensure that your default DECnet account is correctly configured for default file access. In addition, have your system manager use the VMS Network Control Program (NCP) to ensure that the server node allows nonprivileged access.

- A common SQL/Services MS–DOS API installation failure

  Refer to Section 6.5.3. The note describes an SQL/Services MS–DOS IVP error in which the IVP fails with –2003 and 9 error status codes when numbers are used within node names. The problem is fixed in the mandatory update for V4.0. SQL/Services now allows numbers as well as alphabetic characters in server node names.

- Common Macintosh API installation failures

  - Renaming of PATHWORKS tool from ‘‘DECtask Tool’’ to ‘‘DECnet Tool’’

    Refer to Section 6.5.9.2. The note provides information about a problem in which the SQL/Services Macintosh API did not work with the DECnet transport mechanism because of a name mismatch with the DECnet tool provided by PATHWORKS for Macintosh Version 1.0. The problem is fixed for the V4.0 mandatory update.

  - Selecting the Apple-Digital Gateway with the Macintosh Chooser

    If you plan to use the AppleTalk network as the SQL/Services transport mechanism, use the Chooser to select the correct Apple-Digital Gateway.

  - Renaming the System Folder

    SQL/Services looks for control panel information in the ‘‘System Folder’’ folder. SQL/Services fails with a network error, –2003, and a specific error code of –1 if the system folder has been renamed or named differently for international products.

7.10.4 SQL/Services Compatibility Issues

This section describes the differences among various versions of SQL/Services that affect software compatibility.

7.10.4.1 SQL/Services V4.0 Server Uses Proxy-Like and Default Access to Authorize V3.0 or V3.1 Client Applications

Explicit access authorization is never granted to an SQL/Services API client application (Rdb/VMS Version 3.1) seeking access to an SQL/Services server (Rdb/VMS Version 4.0); however, the SQL/Services communication server (Version 4.0) does authorize access to incoming SQL/Services client application (Version 3.1) requests using either the proxy-like or default access method.

For the SQL/Services server (Version 4.0) to grant explicit access, you must upgrade your Rdb/VMS Version 3.0 or 3.1 system to Rdb/VMS Version 4.0. As an interim measure, should you choose to upgrade Rdb/VMS at a later date, set up an SQLSRV$PROXY.DAT file on the server system to link incoming user names with local server system accounts. Refer to the Rdb/VMS Version 4.0 *VAX Rdb/VMS Guide to Using SQL/Services* for further information.

7.10.4.2 SQL/Services V4.0 Server Error –2031 Returned to V3.1 Client APIs

The SQL/Services Version 3.1 Application Programming Interface (API) receives the SQLSRV_APINOTSUP (–2031) error mnemonic when it cannot interpret information returned by the SQL/Services Rdb/VMS Version 4.0 communication server. For example, the SQL/Services Rdb/VMS Version 3.1 API receives the SQLSRV_APINOTSUP error when a Version 3.1 API application attempts to access list cursor data.

7.10.4.3 Queue Manager Must Be Started for the SQL/Services IVP to Work

The queue manager must be started in order for the SQL/Services IVP to work. If the VMS queue manager is not started when the SQL/Services IVP runs, the IVP fails with the following error:

```
SQLCA: SQLCODE: -2003  SQLERRD[0]: x20a4 SQLERRD[2] 0
```

8

Optional ECO Patches for the Mandatory Update for VAX Rdb/VMS V4.0

The .A saveset of this mandatory update kit contains optional ECO patches for the mandatory update for V4.0 that you can install after the mandatory update for V4.0 is installed. These patches are optional for one or more of the following reasons:

- The patch has reported side effects.
- The patch changes the query optimizer behavior and some customers may not want the changed behavior.
- The patch will be needed by only a few customers, and the problem does not involve data corruption or incorrect results.
- The patch was not completed in time for field test and is therefore not part of the final kit.

If you do not need the patch (that is, you do not have the problem), then probably you should not install the optional patch. Please read each patch article and then follow the specific instructions for any patch you decide to apply.

8.1 Optional ECO Patches That Can Be Applied to the Mandatory Update for Rdb/VMS V4.0

This section contains the optional ECO patches that can be applied to the mandatory update for Rdb/VMS V4.0.

8.1.1 RDMSHRP ECO 30: Poor OR Optimization Performance on Read/Write Transactions

See Section 6.2.36 for a description of this problem.

This patch resolves a problem with dynamic OR optimization on read/write transactions by changing the query optimizier to use static OR optimization instead of dynamic OR optimization. The patch is optional as it may help some customers (mainly those who want the V3.1B behavior), while hurting others (those who are happy with dynamic OR optimization on V4.0). The patch article file name is VD2A_RDMSHRPV040A$ECO30.ARTICLE.

This problem is fixed V4.1.

To get this patch from the mandatory update kit, use the following command:

```
$ BACKUP <DEVICE>RDBVMS_MUPA040.A/SAV/SEL=VD2A_RDMSHRPV040A$ECO30.ARTICLE *.*
```

8.1.2 RDMMON ECO 01: Rdb/VMS Monitor Fails When the Last User Finishes on a Particular Database

See Section 7.2.11 for a description of this problem.

This patch resolves a problem with the Rdb/VMS V4.0 monitor failing when the last user finishes from a database. This causes an abbreviated bugcheck dump to be written to the monitor log file. The exception is one of the following:

```
MON$UNLOCK_MPLL + 00000031
          or
MON$UNLOCK_MPLL + 00000036
          or
MON$UNLOCK_MPLL + 00000049
```

The patch is optional because it was not completed in time to be placed on the field test kit and therefore is not part of the final kit. The patch article file name is VD2A_RDMMONV040A$ECO01.ARTICLE.

This problem is fixed in V4.1.

To get this patch from the mandatory update kit, use the following command:

```
$ BACKUP <DEVICE>RDBVMS_MUPA040.A/SAV/SEL=VD2A_RDMMONV040A$ECO01.ARTICLE *.*
```

8.1.3 RDMSHRP ECO 31: Multisegmented Index Is Not Selected When a Not-Equal Predicate Is Specified

See Section 7.2.12 for a description of this problem.

This patch resolves a problem in which a query that uses the not-equal predicate referring to a column within the most selective index may cause the query optimizer to chose a different index.

The patch is optional because it was not completed in time to be placed on the field test kit and therefore is not part of the final kit. The patch article file name is VD2A_RDMSHRPV040A$ECO31.ARTICLE.

This problem is fixed in V4.1.

To get this patch from the mandatory update kit, use the following command:

```
$ BACKUP <DEVICE>RDBVMS_MUPA040.A/SAV/SEL=VD2A_RDMSHRPV040A$ECO31.ARTICLE *.*
```

8.1.4 RDMSHRP ECO 32: Singleton Subselect Statement Returns Incorrect Results

See Section 7.2.14 for a description of this problem.

This patch resolves a problem in which a query in an SQL precompiled C program involving a singleton subselect can return incorrect results.

The patch is optional because it was not completed in time to be placed on the field test kit and therefore is not part of the final kit. The patch article file name is VD2A_RDMSHRPV040A$ECO32.ARTICLE.

This problem is fixed in V4.1.

To get this patch from the mandatory update kit, use the following command:

```
$ BACKUP <DEVICE>RDBVMS_MUPA040.A/SAV/SEL=VD2A_RDMSHRPV040A$ECO32.ARTICLE *.*
```

8.1.5 RDMSHRP ECO 33: Query with a FOR Loop with a MODIFY Statement Followed by a PRINT Statement Can Return Incorrect Results

See Section 7.2.15 for a description of this problem.

This patch resolves a problem in which a query with MODIFY and PRINT statements returns incorrect results.

The patch is optional because it was not completed in time to be placed on the field test kit and therefore is not part of the final kit. The patch article file name is VD2A_RDMSHRPV040A$ECO33.ARTICLE.

This problem is fixed in V4.1.

To get this patch from the mandatory update kit, use the following command:

```
$ BACKUP <DEVICE>RDBVMS_MUPA040.A/SAV/SEL=VD2A_RDMSHRPV040A$ECO33.ARTICLE *.*
```

A

Sample V3.1C Installation

```
$ @SYS$UPDATE:VMSINSTAL RDBVMS_MUPA040 SYS$LOGIN: OPTIONS N

        VAX/VMS Software Product Installation Procedure V5.3

It is 30-APR-1991 at 00:05.

Enter a question mark (?) at any time for help.

* Are you satisfied with the backup of your system disk [YES]?

The following products will be processed:

  RDBVMS_MUPA V4.0

        Beginning installation of RDBVMS_MUPA V4.0 at 00:05

%VMSINSTAL-I-RESTORE, Restoring product saveset A ...

    Release notes included with this kit are always copied to SYS$HELP.

    Additional Release Notes Options:

        1.  Display release notes
        2.  Print release notes
        3.  Both 1 and 2
        4.  None of the above

* Select option [2]:
* Do you want to continue the installation [NO]? Y
%VMSINSTAL-I-RELMOVED, Product's release notes have been successfully moved
to SYS$HELP.

        Installation procedures for: "VAX Rdb/VMS V3.1C-0"

Please ignore any analyze messages

%ANALYZE-I-ERRORS, VMI$ROOT:[SYSEXE]RDO.EXE;5
0 errors


    Be sure you have read the section on pre-installation steps in the
    installation guide before continuing with the installation.


    Checking system requirements ...

* Do you want to run the IVP after the installation [YES]?
* Do you want to purge files replaced by this installation [YES]?

    There are no more questions.  The installation takes
    approximately 20 minutes on a stand-alone VAX 8800.
    License type: DEV

    Beginning installation ...

    Installing under VMS V5.3     - 30-APR-1991 00:06
```

```
%VMSINSTAL-I-RESTORE, Restoring product save set B ...
%VMSINSTAL-I-RESTORE, Restoring product save set F ...

Please ignore any analyze messages

%ANALYZE-I-ERRORS, VMI$ROOT:[SYSLIB]RDMSHRP.EXE;2
0 errors

    ************************************************************

    The Rdb/VMS Installation Verification Procedure (IVP) has
    been provided in SYS$COMMON:[SYSTEST].

    It is invoked using the commands:

        $ SET DEFAULT SYS$COMMON:[SYSTEST]
        $ @RDBIVP DEV

    ************************************************************


    ************************************************************
        The release notes for Rdb/VMS are available in the file
              SYS$HELP:RDBVMS_MUPA040.RELEASE_NOTES
    ************************************************************

%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...

Executing IVP for: VAX Rdb/VMS V3.1C-0

Rdb/VMS monitor (RDMS_MONITOR) started
SQL: Assigning System-wide SQL Logicals
Building the test database.

%RDO-W-NOCDDUPDAT, database invoked by filename, the CDD will not be updated
Program: Loading EMPLOYEES
Program: EMPLOYEES Loaded. Normal End-of-Job
Program: Loading JOBS
Program: JOBS Loaded. Normal End-of-Job
Program: Loading DEPTS
Program: DEPTS Loaded. Normal End-of-Job
%RDO-W-NOCDDUPDAT, database invoked by filename, the CDD will not be updated

Beginning Installation Verification Tests.


Running the after-image journaling test.
    Test completed successfully

Running the remote database test.
    Test completed successfully

Running the interpreter test.
    Test completed successfully

Running the BASIC precompiler test.
    Test completed successfully

Running the COBOL precompiler test.
    Test completed successfully

Running the FORTRAN precompiler test.
    Test completed successfully

Running the RDML/C preprocessor test.
    Test completed successfully

Running the RDML/PASCAL preprocessor test.
    Test completed successfully

Running the SQL/Services tests.
 Running SQL/Services D_FLOAT test
        You must have proxy enabled to run the sql services ivp program.
        See the Rdb/VMS installation guide for more information.
        All of the sql services files are in their proper directories.
```

```
 SQL/Services tests completed successfully.

Building the SQL test database.

Program: Loading EMPLOYEES
Program: EMPLOYEES Loaded. Normal End-of-Job
Program: Loading JOBS
Program: JOBS Loaded. Normal End-of-Job
Program: Loading DEPTS
Program: DEPTS Loaded. Normal End-of-Job

Running the Interactive SQL test.
    Test completed successfully

Running the Dynamic SQL test.
    Test completed successfully

Running the COBOL precompiler test.
    Test completed successfully

Running the FORTRAN precompiler test.
    Test completed successfully

Running the VAX C precompiler test.
    Test completed successfully

Running the VAX Ada precompiler test.
    Test completed successfully

Running the VAX SQL MODULE LANGUAGE test.
    Test completed successfully

    ************************************

    VAX Rdb/VMS V3.1C

    Development

    IVP COMPLETED SUCCESSFULLY

    ************************************

IVP completed for: VAX Rdb/VMS V3.1C-0

        Installation of RDBVMS_MUPA V4.0 completed at 00:25

        VMSINSTAL procedure done at 00:25
```

B

Sample V4.0A Installation

```
ROLLS - Unauthorized Access is Prohibited

Username: SYSTEM
Password:

ROLLS - Property of Digital Equipment Corporation

    Last interactive login on Wednesday,  1-MAY-1991 07:53
    Last non-interactive login on Wednesday,  1-MAY-1991 07:33
$ @SYS$STARTUP:SQLSRV$SHUTDOWN
$ @SYS$MANAGER:RMONSTOP
Shutting down the Rdb/VMS monitor (RDMS_MONITOR)
$ @SYS$UPDATE:VMSINSTAL RDBVMS_MUPA040 OPTIONS N

        VAX/VMS Software Product Installation Procedure V5.4


It is 1-MAY-1991 at 07:54.

Enter a question mark (?) at any time for help.
* Are you satisfied with the backup of your system disk [YES]?
* Where will the distribution volumes be mounted: SYS$LOGIN:

The following products will be processed:
  RDBVMS_MUPA V4.0
Beginning installation of RDBVMS_MUPA V4.0 at 07:54

%VMSINSTAL-I-RESTORE, Restoring product saveset A ...


    Release notes included with this kit are always copied to SYS$HELP.

    Additional Release Notes Options:

        1.  Display release notes
        2.  Print release notes
        3.  Both 1 and 2
        4.  None of the above

* Select option [2]:
* Do you want to continue the installation [NO]? Y
%VMSINSTAL-I-RELMOVED, Product's release notes have been moved to SYS$HELP.


        Installation procedures for: "VAX Rdb/VMS V4.0A"


Please ignore any analyze messages

%ANALYZE-I-ERRORS, VMI$ROOT:[SYSEXE]RDO.EXE;11
0 errors


    Be sure you have read the section on pre-installation steps in the
    installation guide before continuing with the installation.


    Checking system requirements ...

* Do you want to run the IVP after the installation [YES]?
* Do you want to purge files replaced by this installation [YES]?
```

```
    There are no more questions.  The installation takes
    approximately 20 minutes on a stand-alone VAX 8800.
    License type: DEV

    Beginning installation ...

    Installing under VMS V5.4     -  1-MAY-1991 07:54

%VMSINSTAL-I-RESTORE, Restoring product save set C ...
%VMSINSTAL-I-RESTORE, Restoring product save set D ...
%VMSINSTAL-I-RESTORE, Restoring product save set E ...
%VMSINSTAL-I-RESTORE, Restoring product save set G ...
    ************************************************************

    The Rdb/VMS Installation Verification Procedure (IVP) has
    been provided in SYS$COMMON:[SYSTEST].

    It is invoked using the commands:

        $ SET DEFAULT SYS$COMMON:[SYSTEST]
        $ @RDBIVP DEV

    ************************************************************


    ************************************************************
        The release notes for Rdb/VMS are available in the file
              SYS$HELP:RDBVMS_MUPA040.RELEASE_NOTES
    ************************************************************

%VMSINSTAL-I-MOVEFILES, files will now be moved to their target directories...

Executing IVP for: VAX Rdb/VMS V4.0A

Rdb/VMS monitor (RDMS_MONITOR) started
SQL: Assigning System-wide SQL Logicals
Building the test database.

%RDO-W-NOCDDUPDAT, database invoked by filename, the CDD will not be updated
Program: Loading EMPLOYEES
Program: EMPLOYEES Loaded. Normal End-of-Job
Program: Loading JOBS
Program: JOBS Loaded. Normal End-of-Job
Program: Loading DEPTS
Program: DEPTS Loaded. Normal End-of-Job
%RDO-W-NOCDDUPDAT, database invoked by filename, the CDD will not be updated

Beginning Installation Verification Tests.


Running the after-image journaling test.
    Test completed successfully

Running the remote database test.
    Test completed successfully

Running the interpreter test.
    Test completed successfully

Running the BASIC precompiler test.
    Test completed successfully

Running the COBOL precompiler test.
    Test completed successfully

Running the FORTRAN precompiler test.
    Test completed successfully

Running the RDML/C preprocessor test.
    Test completed successfully
```

```
Running the RDML/PASCAL preprocessor test.
    Test completed successfully

Running the SQL/Services tests.
 Running SQL/Services D_FLOAT test
Assigning System-wide SQLSRV Logicals
Starting the SQL Services Server


SQL/Services IVP succeeded

    Test completed successfully.
 Running SQL/Services G_FLOAT test


SQL/Services IVP succeeded

    Test completed successfully.

 SQL/Services tests completed successfully.

Building the SQL test database.

Program: Loading EMPLOYEES
Program: EMPLOYEES Loaded. Normal End-of-Job
Program: Loading JOBS
Program: JOBS Loaded. Normal End-of-Job
Program: Loading DEPTS
Program: DEPTS Loaded. Normal End-of-Job

Running the Interactive SQL test.
    Test completed successfully

Running the Dynamic SQL test.
    Test completed successfully

Running the COBOL precompiler test.
    Test completed successfully

Running the FORTRAN precompiler test.
    Test completed successfully

Running the VAX C precompiler test.
    Test completed successfully

Running the VAX Ada precompiler test.
    Test completed successfully

Running the VAX SQL MODULE LANGUAGE test.
    Test completed successfully

    ************************************

    VAX Rdb/VMS V4.0A

    Development

    IVP COMPLETED SUCCESSFULLY

    ************************************


IVP completed for: VAX Rdb/VMS V4.0A

        Installation of RDBVMS_MUPA V4.0 completed at 08:17

        VMSINSTAL procedure done at 08:17

$ LOGOUT
  SYSTEM       logged out at  1-MAY-1991 08:20:29.23
```

Reader's Comments

VAX Rdb/VMS
Mandatory Update for
Versions 3.1B and 4.0
August 1991

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

Do Not Tear – Fold Here and Tape

digital™

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Information Products
NUO1–1/G10
55 NORTHEASTERN BLVD.
NASHUA, NH 03062–9934

Do Not Tear – Fold Here