Oracle Berkeley DB

Installation and Build Guide

Release 18.1
Library Version 18.1.40
# Table of Contents

- Preface ........................................................................................................................................... xiv
- Conventions Used in this Book ........................................................................................................... xiv
- For More Information ............................................................................................................................ xiv
- Contact Us ........................................................................................................................................... xiv
- 1. Introduction ................................................................................................................................. 1
  - Installation Overview ....................................................................................................................... 1
  - Supported Platforms ......................................................................................................................... 1
- 2. System Installation Notes ............................................................................................................... 3
  - File utility /etc/magic information .................................................................................................... 3
    - Magic information ......................................................................................................................... 3
    - Big-endian magic information ....................................................................................................... 4
    - Little-endian magic information .................................................................................................... 7
  - Building with multiple versions of Berkeley DB .............................................................................. 9
- 3. Debugging Applications ................................................................................................................. 11
  - Introduction to debugging ............................................................................................................... 11
  - Compile-time configuration ............................................................................................................ 11
  - Run-time error information ............................................................................................................. 12
  - Reviewing Berkeley DB log files .................................................................................................... 12
    - Augmenting the Log for Debugging ............................................................................................... 16
    - Extracting Committed Transactions and Transaction Status ...................................................... 16
    - Extracting Transaction Histories .................................................................................................. 16
    - Extracting File Histories ............................................................................................................... 16
    - Extracting Page Histories ............................................................................................................. 16
    - Other log processing tools ........................................................................................................... 17
- 4. Building Berkeley DB for Windows ............................................................................................... 18
  - Building Berkeley DB for Windows .................................................................................................. 18
    - Visual C++ .NET 2015 .................................................................................................................. 18
    - Visual C++ .NET 2013 .................................................................................................................. 18
    - Visual C++ .NET 2012 .................................................................................................................. 19
    - Visual C++ .NET 2010 .................................................................................................................. 19
    - Build results ................................................................................................................................. 19
  - Building Berkeley DB with Cygwin ................................................................................................... 20
  - Building the C++ API ....................................................................................................................... 20
  - Building the C++ STL API ............................................................................................................... 20
  - Building the Java API ...................................................................................................................... 20
  - Building the C# API ........................................................................................................................ 21
  - Building the Tcl API ....................................................................................................................... 21
  - Building Secure Sockets Layer (SSL) Support for the Replication Manager ................................ 22
  - Distributing DLLs ............................................................................................................................. 24
  - Additional build options ................................................................................................................ 24
  - Building a small memory footprint library ..................................................................................... 25
  - Running the test suite under Windows ........................................................................................... 26
    - Building the software needed by the tests .................................................................................... 26
    - Visual Studio 2010 or newer ......................................................................................................... 26
    - Running the test suite under Windows ......................................................................................... 26
  - Windows notes .................................................................................................................................. 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows FAQ</td>
<td>28</td>
</tr>
<tr>
<td>5. Building Berkeley DB for UNIX/POSIX</td>
<td>30</td>
</tr>
<tr>
<td>Building for UNIX/POSIX</td>
<td>30</td>
</tr>
<tr>
<td>Configuring Berkeley DB</td>
<td>30</td>
</tr>
<tr>
<td>Building a small memory footprint library</td>
<td>37</td>
</tr>
<tr>
<td>Changing compile or load options</td>
<td>39</td>
</tr>
<tr>
<td>Cross-Compiling on Unix</td>
<td>40</td>
</tr>
<tr>
<td>Installing Berkeley DB</td>
<td>41</td>
</tr>
<tr>
<td>Dynamic shared libraries</td>
<td>42</td>
</tr>
<tr>
<td>Running the test suite under UNIX</td>
<td>43</td>
</tr>
<tr>
<td>Architecture independent FAQ</td>
<td>44</td>
</tr>
<tr>
<td>AIX</td>
<td>47</td>
</tr>
<tr>
<td>FreeBSD</td>
<td>48</td>
</tr>
<tr>
<td>HP-UX</td>
<td>49</td>
</tr>
<tr>
<td>Apple iOS (iPhone OS)</td>
<td>50</td>
</tr>
<tr>
<td>IRIX</td>
<td>51</td>
</tr>
<tr>
<td>Linux</td>
<td>51</td>
</tr>
<tr>
<td>Mac OS X</td>
<td>52</td>
</tr>
<tr>
<td>SCO</td>
<td>53</td>
</tr>
<tr>
<td>Solaris</td>
<td>53</td>
</tr>
<tr>
<td>SunOS</td>
<td>55</td>
</tr>
<tr>
<td>6. Building Berkeley DB for VxWorks</td>
<td>56</td>
</tr>
<tr>
<td>Building for VxWorks 6.x and VxWorks 7</td>
<td>56</td>
</tr>
<tr>
<td>Building With Wind River Workbench using the Makefile</td>
<td>56</td>
</tr>
<tr>
<td>VxWorks notes</td>
<td>56</td>
</tr>
<tr>
<td>Building and Running the Demo Program</td>
<td>57</td>
</tr>
<tr>
<td>Building and Running the Utility Programs</td>
<td>57</td>
</tr>
<tr>
<td>Support for Replication Manager</td>
<td>57</td>
</tr>
<tr>
<td>VxWorks FAQ</td>
<td>57</td>
</tr>
<tr>
<td>7. Building Third Party APIs</td>
<td>61</td>
</tr>
<tr>
<td>Building the Go API</td>
<td>61</td>
</tr>
<tr>
<td>Install Oracle Berkeley DB</td>
<td>61</td>
</tr>
<tr>
<td>Build Go Berkeley DB (go-bdb)</td>
<td>62</td>
</tr>
<tr>
<td>Test Go Berkeley DB (go-bdb)</td>
<td>62</td>
</tr>
<tr>
<td>Additional Information</td>
<td>62</td>
</tr>
<tr>
<td>Building Node.js</td>
<td>63</td>
</tr>
<tr>
<td>Additional Information</td>
<td>63</td>
</tr>
<tr>
<td>8. Upgrading Berkeley DB 12.1.6.2 applications to Berkeley DB 18.1</td>
<td>64</td>
</tr>
<tr>
<td>Introduction</td>
<td>64</td>
</tr>
<tr>
<td>License Change</td>
<td>64</td>
</tr>
<tr>
<td>Secure Sockets Layer (SSL) Support for the Replication Manager</td>
<td>64</td>
</tr>
<tr>
<td>New Function</td>
<td>65</td>
</tr>
<tr>
<td>New Flags</td>
<td>65</td>
</tr>
<tr>
<td>Multiversion Concurrency Control (MVCC) for Replication Clients</td>
<td>66</td>
</tr>
<tr>
<td>New Replication Environment Region Objects</td>
<td>66</td>
</tr>
<tr>
<td>New Parameters</td>
<td>66</td>
</tr>
<tr>
<td>New Replication Manager Flags</td>
<td>66</td>
</tr>
<tr>
<td>New Parameter Values</td>
<td>67</td>
</tr>
<tr>
<td>New Information Available to Replicated Applications</td>
<td>67</td>
</tr>
</tbody>
</table>
New Error ..................................................................................... 67
New Replication Manager Event ......................................................... 67
New Replication Manager Statistic ....................................................... 67
New Replication Manager Site Field ..................................................... 67
New Replication Manager Site Flag ....................................................... 68
New Utility db_convert ............................................................................. 68
New Direct Persistence Layer (DPL) API .................................................. 68
New Utility Command Line Option .......................................................... 68
SQL API Changes .................................................................................... 68
  SQL User Authentication .................................................................... 68
  New PRAGMAS ................................................................................ 68
  Internal Berkeley DB Messages ........................................................... 68
  Updated SQL Version ............................................................... 68
Changed Visual Studio Support ................................................................. 69
Added VxWorks Support ........................................................................... 69
Added PHP 7 Support .............................................................................. 69
Error Message Numbers ........................................................................... 69
Removed Utility db_sql_codegen ............................................................... 69
Removed QNX Support ............................................................................ 69
Berkeley DB Library Version 18.1.40 Change Log ........................................... 69
  Changes between version 18.1.32 and version 18.1.40 ......................... 69
  Changes between version 18.1.25 and version 18.1.32 ......................... 70
  Database or Log File On-Disk Format Changes ....................................... 71
  New Features/Improvements ................................................................ 71
  Database Environment Changes ........................................................... 72
  Concurrent Data Store Changes ............................................................ 72
  Access Method Changes ..................................................................... 72
  API Changes ................................................................................... 72
  SQL-specific API Changes .................................................................. 72
  Tcl-specific API Changes .................................................................... 73
  Java-specific API Changes .................................................................. 73
  C#-specific API Changes .................................................................... 73
  Replication Changes ......................................................................... 73
  Locking Subsystem Changes ............................................................... 74
  Logging Subsystem Changes ............................................................... 74
  Memory Pool Subsystem Changes ......................................................... 74
  Mutex Subsystem Changes ................................................................. 74
  Transaction Subsystem Changes ........................................................... 74
  Test Suite Changes .......................................................................... 74
  Utility Changes ............................................................................... 75
  Configuration, Documentation, Sample Apps, Portability and Build Changes ........................................................................... 75
  Example Changes ............................................................................ 76
  Miscellaneous Bug Fixes .................................................................... 76
  Deprecated Features ........................................................................... 76
  Known Bugs ................................................................................ 76

9. Upgrading Berkeley DB 12.1.6.1 applications to Berkeley DB 12.1.6.2 ...................... 78
   Introduction ......................................................................................... 78
   Database Slices .................................................................................. 78
   New Functions .................................................................................... 78
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Error</td>
<td>78</td>
</tr>
<tr>
<td>New Parameter Values</td>
<td>78</td>
</tr>
<tr>
<td>Renaming the BLOB file feature to External Files</td>
<td>79</td>
</tr>
<tr>
<td>Functions</td>
<td>79</td>
</tr>
<tr>
<td>New Parameter Values</td>
<td>79</td>
</tr>
<tr>
<td>New Statistics</td>
<td>79</td>
</tr>
<tr>
<td>Configurable Region Directory</td>
<td>79</td>
</tr>
<tr>
<td>New Functions</td>
<td>80</td>
</tr>
<tr>
<td>Replication Write Forwarding</td>
<td>80</td>
</tr>
<tr>
<td>New Parameter Values</td>
<td>80</td>
</tr>
<tr>
<td>Replication Manager IPv6 Support</td>
<td>80</td>
</tr>
<tr>
<td>New Function</td>
<td>80</td>
</tr>
<tr>
<td>Global Message Prefix</td>
<td>80</td>
</tr>
<tr>
<td>New Functions</td>
<td>80</td>
</tr>
<tr>
<td>Message Call Function Signature Change</td>
<td>81</td>
</tr>
<tr>
<td>Reindex SQL Databases</td>
<td>81</td>
</tr>
<tr>
<td>Changes to SQL Behavior</td>
<td>81</td>
</tr>
<tr>
<td>Updated SQLite Version</td>
<td>81</td>
</tr>
<tr>
<td>Support for Java 8</td>
<td>81</td>
</tr>
<tr>
<td>Support for arm64</td>
<td>81</td>
</tr>
<tr>
<td>JDBC Build Changes</td>
<td>81</td>
</tr>
<tr>
<td>varargs.h Not Supported</td>
<td>81</td>
</tr>
<tr>
<td>Removed Java Methods</td>
<td>81</td>
</tr>
<tr>
<td>Dropped Support</td>
<td>82</td>
</tr>
<tr>
<td>Berkeley DB Library Version 12.1.6.2 Change Log</td>
<td>82</td>
</tr>
<tr>
<td>Database or Log File On-Disk Format Changes</td>
<td>82</td>
</tr>
<tr>
<td>New Features/Improvements</td>
<td>82</td>
</tr>
<tr>
<td>Database Environment Changes</td>
<td>83</td>
</tr>
<tr>
<td>Concurrent Data Store Changes</td>
<td>83</td>
</tr>
<tr>
<td>Access Method Changes</td>
<td>83</td>
</tr>
<tr>
<td>API Changes</td>
<td>84</td>
</tr>
<tr>
<td>SQL-specific API Changes</td>
<td>84</td>
</tr>
<tr>
<td>Tcl-specific API Changes</td>
<td>85</td>
</tr>
<tr>
<td>Java-specific API Changes</td>
<td>85</td>
</tr>
<tr>
<td>C#-specific API Changes</td>
<td>85</td>
</tr>
<tr>
<td>Replication Changes</td>
<td>85</td>
</tr>
<tr>
<td>Locking Subsystem Changes</td>
<td>85</td>
</tr>
<tr>
<td>Logging Subsystem Changes</td>
<td>86</td>
</tr>
<tr>
<td>Memory Pool Subsystem Changes</td>
<td>86</td>
</tr>
<tr>
<td>Mutex Subsystem Changes</td>
<td>86</td>
</tr>
<tr>
<td>Transaction Subsystem Changes</td>
<td>86</td>
</tr>
<tr>
<td>Test Suite Changes</td>
<td>86</td>
</tr>
<tr>
<td>Utility Changes</td>
<td>86</td>
</tr>
<tr>
<td>Configuration, Documentation, Sample Apps, Portability and Build Changes</td>
<td>87</td>
</tr>
<tr>
<td>Example Changes</td>
<td>87</td>
</tr>
<tr>
<td>Miscellaneous Bug Fixes</td>
<td>88</td>
</tr>
<tr>
<td>Deprecated Features</td>
<td>89</td>
</tr>
<tr>
<td>Known Bugs</td>
<td>89</td>
</tr>
<tr>
<td>10. Upgrading Berkeley DB 12.1.6.0 applications to Berkeley DB 12.1.6.1</td>
<td>90</td>
</tr>
</tbody>
</table>
Introduction ......................................................................................... 90
Failchk Broadcast Notifications .......................................................... 90
   New Parameter Value ....................................................................... 90
   New Events ................................................................................... 90
New Logging Option ............................................................................. 90
   New Parameter Value ....................................................................... 90
Replication Manager Preferred Master Mode ........................................... 91
   New Parameter Values ......................................................................... 91
Replication Manager Incoming Queue Size Limit .................................... 91
   New Functions ................................................................................ 91
   New Event .................................................................................... 91
New Database Format for BLOBs .......................................................... 91
Updated SQLite Version ......................................................................... 91
   Upgrading SQL databases to releases after 6.1.19 ........................... 91
      Upgrading SQL databases on POSIX/UNIX Systems .................... 92
      Upgrading SQL databases on Windows Systems .......................... 92
Removed C# Class ............................................................................... 92
Dropped Support .................................................................................. 93
Berkeley DB Library Version 12.1.6.1 Change Log ................................ 93
   Database or Log File On-Disk Format Changes .............................. 93
   New Features/Improvements ........................................................... 93
   Database Environment Changes ....................................................... 94
   Concurrent Data Store Changes ....................................................... 94
   Access Method Changes .................................................................. 94
   API Changes ................................................................................ 94
   SQL-specific API Changes ............................................................... 94
   Tcl-specific API Changes ............................................................... 94
   Java-specific API Changes .............................................................. 95
   C#-specific API Changes ............................................................... 95
   C++-specific API Changes .............................................................. 95
   Replication Changes ...................................................................... 95
   Locking Subsystem Changes ........................................................... 96
   Logging Subsystem Changes ............................................................ 96
   Memory Pool Subsystem Changes ................................................... 96
   Mutex Subsystem Changes ............................................................. 96
   Transaction Subsystem Changes ...................................................... 96
   Test Suite Changes ....................................................................... 96
   Utility Changes .............................................................................. 96
   Configuration, Documentation, Sample Apps, Portability and Build Changes .................................................................................. 97
   Example Changes ......................................................................... 97
   Miscellaneous Bug Fixes ............................................................... 97
   Platform Certification Changes ....................................................... 98
   Deprecated Features ..................................................................... 98
   Known Bugs ................................................................................ 98
11. Upgrading Berkeley DB 11.2.5.3 applications to Berkeley DB 12.1.6.0 99
   Introduction .................................................................................. 99
   Binary Large Object Support .......................................................... 99
      New Functions ............................................................................... 99
      New Flags Accepted by Blob Streaming Interfaces ........................ 100
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other New Flags</td>
<td>100</td>
</tr>
<tr>
<td>New Pragma</td>
<td>100</td>
</tr>
<tr>
<td>Replication Views</td>
<td>100</td>
</tr>
<tr>
<td>New Function</td>
<td>100</td>
</tr>
<tr>
<td>New Flag</td>
<td>100</td>
</tr>
<tr>
<td>Enhanced BDB SQL Replication Options</td>
<td>100</td>
</tr>
<tr>
<td>New Pragmas</td>
<td>100</td>
</tr>
<tr>
<td>Replication Manager Automatic Takeover</td>
<td>101</td>
</tr>
<tr>
<td>New Event</td>
<td>101</td>
</tr>
<tr>
<td>New Berkeley DB SQL API Pragma</td>
<td>101</td>
</tr>
<tr>
<td>New Pragma</td>
<td>101</td>
</tr>
<tr>
<td>UNIX/POSIX SQL Encryption Default</td>
<td>101</td>
</tr>
<tr>
<td>Sequence Datatype Changes</td>
<td>101</td>
</tr>
<tr>
<td>Comparison Function Signature Change</td>
<td>102</td>
</tr>
<tr>
<td>License Change</td>
<td>102</td>
</tr>
<tr>
<td>Updated SQLite Version</td>
<td>102</td>
</tr>
<tr>
<td>Sun Workshop Compiler Optimization Level</td>
<td>102</td>
</tr>
<tr>
<td>Dropped Support</td>
<td>102</td>
</tr>
<tr>
<td>Berkeley DB Library Version 12.1.6.0 Change Log</td>
<td>102</td>
</tr>
<tr>
<td>Database or Log File On-Disk Format Changes</td>
<td>102</td>
</tr>
<tr>
<td>New Features</td>
<td>103</td>
</tr>
<tr>
<td>Database Environment Changes</td>
<td>103</td>
</tr>
<tr>
<td>Concurrent Data Store Changes</td>
<td>104</td>
</tr>
<tr>
<td>Access Method Changes</td>
<td>104</td>
</tr>
<tr>
<td>API Changes</td>
<td>104</td>
</tr>
<tr>
<td>SQL-specific API Changes</td>
<td>104</td>
</tr>
<tr>
<td>Tcl-specific API Changes</td>
<td>105</td>
</tr>
<tr>
<td>Java-specific API Changes</td>
<td>105</td>
</tr>
<tr>
<td>C#-specific API Changes</td>
<td>105</td>
</tr>
<tr>
<td>Replication Changes</td>
<td>105</td>
</tr>
<tr>
<td>Locking Subsystem Changes</td>
<td>106</td>
</tr>
<tr>
<td>Logging Subsystem Changes</td>
<td>106</td>
</tr>
<tr>
<td>Memory Pool Subsystem Changes</td>
<td>106</td>
</tr>
<tr>
<td>Mutex Subsystem Changes</td>
<td>107</td>
</tr>
<tr>
<td>Transaction Subsystem Changes</td>
<td>107</td>
</tr>
<tr>
<td>Test Suite Changes</td>
<td>107</td>
</tr>
<tr>
<td>Utility Changes</td>
<td>107</td>
</tr>
<tr>
<td>Configuration, Documentation, Sample Apps, Portability and Build Changes</td>
<td>108</td>
</tr>
<tr>
<td>Example Changes</td>
<td>108</td>
</tr>
<tr>
<td>Miscellaneous Bug Fixes</td>
<td>108</td>
</tr>
<tr>
<td>Deprecated Features</td>
<td>109</td>
</tr>
<tr>
<td>Known Bugs</td>
<td>109</td>
</tr>
</tbody>
</table>

12. Upgrading Berkeley DB 11.2.5.2 applications to Berkeley DB 11.2.5.3

Introduction                                                           | 110  |
Changes to the build_windows Folder                                    | 110  |
Replication Connection Status in the Java API                          | 110  |
New Function                                                           | 110  |
New Class                                                              | 110  |
Deprecated Function                                                    | 110  |
Exclusive Database Handles ................................................................. 110
  New Functions .................................................................................. 111
Configure the Region Size of Heap Databases .................................................. 111
  New Functions .................................................................................. 111
New Hotbackup Interface ................................................................................. 111
  New Functions .................................................................................. 111
Flags Accepted by DB_ENV->backup() ......................................................... 111
Flags Accepted by DB_ENV->dbbackup() ....................................................... 112
Enumerations Accepted by DB_ENV->set_backup_config() ............................... 112
Updated JDBC Version ............................................................................... 112
Configure Directory to Store Metadata Files ................................................... 112
  New Functions .................................................................................. 112
Changes in the SQL API Build ...................................................................... 112
New Berkeley DB SQL API PRAGMAs .......................................................... 113
  New PRAGMAs ............................................................................. 113
Replication for Existing Databases in the SQL API ............................................. 113
  PRAGMAs With Permanent Effects ....................................................... 113
  PRAGMAs That Can Now Operate on Existing Databases ....................... 113
Berkeley DB X/Open Compliant XA Resource Manager and Transaction Snapshots 113
Berkeley DB Library Version 11.2.5.3 Change Log ............................................ 114
  Database or Log File On-Disk Format Changes ...................................... 114
  New Features .................................................................................. 114
  Database Environment Changes ......................................................... 115
  Access Method Changes .................................................................. 115
  SQL API Changes .......................................................................... 116
  Java-specific API changes ................................................................. 117
  Replication Changes ..................................................................... 117
  Locking Subsystem Changes .............................................................. 117
  Logging Subsystem Changes .............................................................. 117
  Memory Pool Subsystem Changes ........................................................ 118
  Mutex Subsystem Changes ............................................................... 118
  Transaction Subsystem Changes .......................................................... 118
  Utility Changes ............................................................................. 118
  Configuration, Documentation, Sample Apps, Portability and Build Changes 118
  Known Bugs .................................................................................. 118
13. Upgrading Berkeley DB 11.2.5.1 applications to Berkeley DB 11.2.5.2 ................... 119
  Introduction .................................................................................... 119
  SQLite Interface Upgrade .................................................................... 119
  32bit/64bit Compatibility on Windows .................................................... 119
  Read Only flag for DBT ...................................................................... 119
  New Flag ..................................................................................... 119
  Dynamic Environment Configuration ...................................................... 119
    New Functions ............................................................................. 119
    Deprecated Functions ................................................................... 120
  Exclusive Transactions in the SQL Layer .................................................. 120
  Group Membership in Repmgr ............................................................. 120
    Upgrading .................................................................................... 121
    New Functions ............................................................................. 121
    Modified Functions ....................................................................... 122
SQLite Interface Upgrade ........................................................................ 134
Mod_db4 Support Discontinued ................................................................. 134
Berkeley DB Library Version 11.2.5.1 Change Log ......................................... 134
  Database or Log File On-Disk Format Changes ........................................ 134
  New Features ........................................................................................................ 134
  Database Environment Changes ........................................................................ 136
  Concurrent Data Store Changes ....................................................................... 136
  Access Method Changes ..................................................................................... 136
  API Changes ........................................................................................................ 137
  SQL-Specific API Changes ............................................................................. 137
  Tcl-Specific API Changes ............................................................................... 137
  Java-Specific API Changes ........................................................................... 138
  C#-Specific API Changes ................................................................................ 138
  Direct Persistence Layer (DPL), Bindings and Collections API ....................... 139
  Replication Changes ....................................................................................... 139
  Locking Subsystem Changes ......................................................................... 140
  Logging Subsystem Changes ......................................................................... 140
  Memory Pool Subsystem Changes .................................................................. 140
  Mutex Subsystem Changes ............................................................................ 140
  Transaction Subsystem Changes ................................................................... 141
  Test Suite Changes ......................................................................................... 141
  UtilityChanges ............................................................................................. 141
  Configuration, Documentation, Sample Apps, Portability, and Build Changes ... 141
  Example Changes ........................................................................................... 142
  Miscellaneous Bug Fixes ............................................................................... 142
  Deprecated Features ...................................................................................... 142

15. Upgrading Berkeley DB 4.8 applications to Berkeley DB 11.2.5.0 ................. 143
  Introduction ....................................................................................................... 143
  db_sql Renamed to db_sql_codegen ............................................................... 143
  DB_REP_CONF_NOAUTOINIT Replaced .......................................................... 143
  Support for Multiple Client-to-Client Peers .................................................. 143
  Cryptography Support ..................................................................................... 143
  DB_NOSYNC Flag to Flush Files .................................................................... 144
  Dropped Support ............................................................................................ 144
  Changing Stack Size ........................................................................................ 144
  Berkeley DB 11g Release 2 Change Log .......................................................... 144
  Changes between 11.2.5.0.26 and 11.2.5.0.32 ................................................ 144
  Changes between 11.2.5.0.21 and 11.2.5.0.26 .............................................. 146
  Changes between 4.8 and 11.2.5.0.21 ........................................................... 147
  Database or Log File On-Disk Format Changes ............................................ 147
  New Features .................................................................................................... 147
  Database Environment Changes ...................................................................... 147
  Access Method Changes ................................................................................. 148
  Locking Subsystem Changes .......................................................................... 149
  Logging Subsystem Changes .......................................................................... 150
  Memory Pool Subsystem Changes .................................................................. 150
  Mutex Subsystem Changes ............................................................................. 150
  Tcl-specific API Changes ................................................................................ 151
C#-specific API Changes ............................................................ 151
API Changes ........................................................................... 151
Replication Changes ................................................................. 151
Transaction Subsystem Changes ................................................... 152
Utility Changes ....................................................................... 152
Example Changes .................................................................... 153
Deprecated Features ................................................................ 153
Configuration, Documentation, Sample Apps, Portability and Build Changes ................................................................. 153

Known Bugs .................................................................................. 154

16. Upgrading Berkeley DB 4.7 applications to Berkeley DB 4.8 .................................................................................. 155
Introduction ........................................................................................ 155
Registering DPL Secondary Keys .................................................... 155
Minor Change in Behavior of DB_MPOOLFILE->get ........................ 155
Dropped Support for fcntl System Calls ........................................... 155
Upgrade Requirements ..................................................................... 156
Berkeley DB 4.8.28 Change Log ........................................................ 156
Changes between 4.8.26 and 4.8.28: .................................................. 156
Known bugs in 4.8 ....................................................................... 156
Changes between 4.8.24 and 4.8.26: .................................................. 156
Changes between 4.8.21 and 4.8.24: .................................................. 157
Changes between 4.7 and 4.8.21: ...................................................... 157
Database or Log File On-Disk Format Changes: .............................. 157
New Features: ............................................................................. 157
Database Environment Changes: ..................................................... 158
Concurrent Data Store Changes: .................................................... 158
General Access Method Changes: .................................................. 158
Btree Access Method Changes: ...................................................... 159
Hash Access Method Changes: ...................................................... 159
Queue Access Method Changes: ..................................................... 160
Recno Access Method Changes: ..................................................... 160
C-specific API Changes: ............................................................... 160
C++specific API Changes: ............................................................ 160
Java-specific API Changes: ........................................................... 160
Direct Persistence Layer (DPL), Bindings and Collections API: .............. 161
Tcl-specific API Changes: ............................................................. 162
RPC-specific Client/Server Changes: ............................................. 162
Replication Changes: .................................................................. 162
XA Resource Manager Changes: .................................................. 164
Locking Subsystem Changes: ......................................................... 164
Logging Subsystem Changes: ........................................................ 165
Memory Pool Subsystem Changes: ............................................... 165
Mutex Subsystem Changes: ........................................................... 165
Test Suite Changes ....................................................................... 166
Transaction Subsystem Changes: ................................................... 166
Utility Changes: ........................................................................... 166
Configuration, Documentation, Sample Application, Portability and Build Changes: ......................................................... 167

17. Test Suite ........................................................................................ 169
Preface

Welcome to Berkeley DB (DB). This document describes how to build, install and upgrade Berkeley DB

This document reflects Berkeley DB 18 Release 18.1, which provides DB library version 18.1.40.

Conventions Used in this Book

The following typographical conventions are used within in this manual:

Variable or non-literal text is presented in italics. For example: "Go to your DB_INSTALL directory."

Program examples are displayed in a monospaced font on a shaded background. For example:

Note

Finally, notes of interest are represented using a note block such as this.

For More Information

Beyond this manual, you may also find the following sources of information useful when building a DB application:

• Getting Started with Transaction Processing for C
• Berkeley DB Getting Started with Replicated Applications for C
• Berkeley DB C API Reference Guide
• Berkeley DB C++ API Reference Guide
• Berkeley DB STL API Reference Guide
• Berkeley DB TCL API Reference Guide
• Berkeley DB Programmer’s Reference Guide
• Berkeley DB Upgrade Guide

To download the latest documentation along with white papers and other collateral, visit http://www.oracle.com/technetwork/indexes/documentation/index.html.


Contact Us

You can post your comments and questions at the Oracle Technology (OTN) forum for
For sales or support information, email to: berkeleydb-info_us@oracle.com You can subscribe to a low-volume email announcement list for the Berkeley DB product family by sending email to: bdb-join@oss.oracle.com
Chapter 1. Introduction

Welcome to the Berkeley DB. This manual describes how to configure, build and install Berkeley DB. Installation of DB for all of the platforms it officially supports is described in this manual. Upgrade instructions and release notes for newer versions of this product are described here. For information on upgrading from historical versions, see the Berkeley DB Upgrade Guide.

Note that some operating systems and distributions might provide DB, either by default or as part of an installation option. If so, those platforms will have installation instructions for DB specific to them. In this situation, you should see the documentation for your operating system or distribution provider for information on how to get DB on your platform.

Installation Overview

Berkeley DB is an open-source product, and as such it is usually offered in source-code format. This means that placing DB on your platform requires you to configure the build scripts, compile it, and then install the product onto your host system. The exception to this are Microsoft Windows platforms for which a binary installer is available. The Windows binary installer provides a fairly standard configuration of Berkeley DB; if you need special features you may enable them by compiling the product from source.

For *nix systems, including the BSD and Linux systems, the usual configure, make and make install installation process is used to place DB on your platform.

For information on building and installing Berkeley DB on:

- Microsoft Windows, see Building Berkeley DB for Windows (page 18).
- Unix/POSIX — including Linux, BSD, Apple iOS (known as iPhone OS previously), and Mac OS X — see Building Berkeley DB for UNIX/POSIX (page 30).
- VxWorks, see Building Berkeley DB for VxWorks (page 56).

Supported Platforms

You can install Berkeley DB on the following platforms:

- Most versions of Linux (x86-64 and x86) including:
  - Oracle Linux
  - Red Hat
  - Ubuntu
  - Wind River
  - MontaVista Embedded Linux version 6.0
- Oracle Solaris versions 9, 10, and 11 on x86_64, x86, and SPARC.
• FreeBSD
• Microsoft Windows (x86-64 and x86).
  • Vista
  • Windows 7
  • Windows 8
  • Server 2008
  • Server 2012
• Apple Mac OS X.
• IBM AIX version 6 and 7.
• VxWorks 6.x
• Apple iOS (previously known as iPhone OS)

Apart from those mentioned in the list above, you can install Berkeley DB on most other systems which are POSIX-compliant. When there is a need to run Berkeley DB on a platform that is currently not supported, DB is distributed in source code form that you can use as base source to port Berkeley DB to that platform. For more information on porting to other platforms, see the Berkeley DB Porting Guide.
Chapter 2. System Installation Notes

File utility /etc/magic information

The `file` utility is a UNIX utility that examines and classifies files, based on information found in its database of file types, the `/etc/magic` file. The following information may be added to your system's `/etc/magic` file to enable `file` to correctly identify Berkeley DB database files.

The `file` utility magic(5) information for the standard System V UNIX implementation of the `file` utility is included in the Berkeley DB distribution for both big-endian (for example, Sparc) and little-endian (for example, x86) architectures. See Big-endian magic information (page 4) and Little-endian magic information (page 7) respectively for this information.

The `file` utility magic(5) information for Release 3.X of Ian Darwin's implementation of the file utility (as distributed by FreeBSD and most Linux distributions) is included in the Berkeley DB distribution. This `magic.txt` information is correct for both big-endian and little-endian architectures. See the next section for this information.

Magic information

```
# Berkeley DB
#
# Ian Darwin's file /etc/magic files: big/little-endian version.
# Hash 1.85/1.86 databases store metadata in network byte order.
# Btree 1.85/1.86 databases store the metadata in host byte order.
# Hash and Btree 2.X and later databases store the metadata in
# host byte order.

0 long 0x00061561 Berkeley DB
>8 belong 4321
>>4 belong >2  1.86
>>4 belong <3  1.85
>>4 belong >0  (Hash, version %d, native byte-order)
>8 belong 1234
>>4 belong >2  1.86
>>4 belong <3  1.85
>>4 belong >0  (Hash, version %d, little-endian)

0 belong 0x00061561 Berkeley DB
>8 belong 4321
>>4 belong >2  1.86
>>4 belong <3  1.85
>>4 belong >0  (Hash, version %d, big-endian)
>8 belong 1234
>>4 belong >2  1.86
>>4 belong <3  1.85
>>4 belong >0  (Hash, version %d, native byte-order)
```
Big-endian magic information

# Berkeley DB
#
# System V /etc/magic files: big-endian version.
#
# Hash 1.85/1.86 databases store metadata in network byte order.
# Btree 1.85/1.86 databases store the metadata in host byte order.
# Hash and Btree 2.X and later databases store the metadata in
# host byte order.

0 long 0x00053162 Berkeley DB 1.85/1.86
>4 long >0 (Btree, version %d, native byte-order)
0 belong 0x00053162 Berkeley DB 1.85/1.86
>4 belong >0 (Btree, version %d, big-endian)
0 lelong 0x00053162 Berkeley DB 1.85/1.86
>4 lelong >0 (Btree, version %d, little-endian)

12 long 0x00061561 Berkeley DB
>16 long >0 (Hash, version %d, native byte-order)
12 belong 0x00061561 Berkeley DB
>16 belong >0 (Hash, version %d, big-endian)
12 lelong 0x00061561 Berkeley DB
>16 lelong >0 (Hash, version %d, little-endian)

12 long 0x00053162 Berkeley DB
>16 long >0 (Btree, version %d, native byte-order)
12 belong 0x00053162 Berkeley DB
>16 belong >0 (Btree, version %d, big-endian)
12 lelong 0x00053162 Berkeley DB
>16 lelong >0 (Btree, version %d, little-endian)

12 long 0x00042253 Berkeley DB
>16 long >0 (Queue, version %d, native byte-order)
12 belong 0x00042253 Berkeley DB
>16 belong >0 (Queue, version %d, big-endian)
12 lelong 0x00042253 Berkeley DB
>16 lelong >0 (Queue, version %d, little-endian)

12 long 0x00040988 Berkeley DB
>16 long >0 (Log, version %d, native byte-order)
12 belong 0x00040988 Berkeley DB
>16 belong >0 (Log, version %d, big-endian)
12 lelong 0x00040988 Berkeley DB
>16 lelong >0 (Log, version %d, little-endian)
>0 long 0x00053162 native byte-order)

0 long 0x62310500 Berkeley DB 1.85/1.86 (Btree,
>4 long 0x02000000 version 2,
>4 long 0x03000000 version 3,
>0 long 0x62310500 little-endian)

12 long 0x00053162 Berkeley DB (Btree,
>16 long 0x00000004 version 4,
>16 long 0x00000005 version 5,
>16 long 0x00000006 version 6,
>16 long 0x00000007 version 7,
>16 long 0x00000008 version 8,
>16 long 0x00000009 version 9,
>12 long 0x00053162 native byte-order)

12 long 0x62310500 Berkeley DB (Btree,
>16 long 0x04000000 version 4,
>16 long 0x05000000 version 5,
>16 long 0x06000000 version 6,
>16 long 0x07000000 version 7,
>16 long 0x08000000 version 8,
>16 long 0x09000000 version 9,
>12 long 0x62310500 little-endian)

0 long 0x00061561 Berkeley DB
>4 long >2 1.86
>4 long <3 1.85
>0 long 0x00061561 (Hash,
>4 long 2 version 2,
>4 long 3 version 3,
>8 long 0x000004D2 little-endian)
>8 long 0x000010E1 native byte-order)

12 long 0x00061561 Berkeley DB (Hash,
>16 long 0x00000004 version 4,
>16 long 0x00000005 version 5,
>16 long 0x00000006 version 6,
>16 long 0x00000007 version 7,
>16 long 0x00000008 version 8,
>16 long 0x00000009 version 9,
>12 long 0x00061561 native byte-order)

12 long 0x61150600 Berkeley DB (Hash,
>16 long 0x04000000 version 4,
>16 long 0x05000000 version 5,
>16 long 0x06000000 version 6,
>16 long 0x07000000 version 7,
>16 long 0x08000000 version 8,
>16 long 0x09000000 version 9,
>12 long 0x61150600 little-endian)

12 long 0x00042253 Berkeley DB (Queue,
>16 long 0x00000001 version 1,
>16 long 0x00000002 version 2,
>16 long 0x00000003 version 3,
>16 long 0x00000004 version 4,
>16 long 0x00000005 version 5,
>16 long 0x00000006 version 6,
>16 long 0x00000007 version 7,
>16 long 0x00000008 version 8,
>16 long 0x00000009 version 9,
>12 long 0x00042253 native byte-order)

12 long 0x53220400 Berkeley DB (Queue,
>16 long 0x01000000 version 1,
>16 long 0x02000000 version 2,
>16 long 0x03000000 version 3,
>16 long 0x04000000 version 4,
>16 long 0x05000000 version 5,
>16 long 0x06000000 version 6,
>16 long 0x07000000 version 7,
>16 long 0x08000000 version 8,
>16 long 0x09000000 version 9,
>12 long 0x53220400 little-endian)

12 long 0x00040988 Berkeley DB (Log,
>16 long 0x00000001 version 1,
>16 long 0x00000002 version 2,
>16 long 0x00000003 version 3,
>16 long 0x00000004 version 4,
>16 long 0x00000005 version 5,
>16 long 0x00000006 version 6,
>16 long 0x00000007 version 7,
>16 long 0x00000008 version 8,
>16 long 0x00000009 version 9,
>16 long 0x0000000a version 10,
>16 long 0x0000000b version 11,
>16 long 0x0000000c version 12,
>16 long 0x0000000d version 13,
>16 long 0x0000000e version 14,
>16 long 0x0000000f version 15,
>12 long 0x00040988 native byte-order)

12 long 0x88090400 Berkeley DB (Log,
>16 long 0x01000000 version 1,
>16 long 0x02000000 version 2,
>16 long 0x03000000 version 3,
Little-endian magic information

# Berkeley DB
# System V /etc/magic files: little-endian version.
# Hash 1.85/1.86 databases store metadata in network byte order.
# Btree 1.85/1.86 databases store the metadata in host byte order.
# Hash and Btree 2.X and later databases store the metadata in
# host byte order.

0 long 0x00053162 Berkeley DB 1.85/1.86 (Btree,
  4 long 0x00000002 version 2,
  4 long 0x00000003 version 3,
  0 long 0x00053162 native byte-order)

0 long 0x62310500 Berkeley DB 1.85/1.86 (Btree,
  4 long 0x02000000 version 2,
  4 long 0x03000000 version 3,
  0 long 0x62310500 big-endian)

12 long 0x00053162 Berkeley DB (Btree,
  16 long 0x00000004 version 4,
  16 long 0x00000005 version 5,
  16 long 0x00000006 version 6,
  16 long 0x00000007 version 7,
  16 long 0x00000008 version 8,
  16 long 0x00000009 version 9,
  12 long 0x00053162 native byte-order)

12 long 0x62310500 Berkeley DB (Btree,
  16 long 0x04000000 version 4,
  16 long 0x05000000 version 5,
  16 long 0x06000000 version 6,
  16 long 0x07000000 version 7,
  16 long 0x08000000 version 8,
>16 long 0x09000000 version 9,
>12 long 0x62310500 big-endian)

0 long 0x61150600 Berkeley DB
>4 long >0x02000000 1.86
>4 long <0x03000000 1.85
>0 long 0x00061561 (Hash,
>4 long 0x02000000 version 2,
>4 long 0x03000000 version 3,
>8 long 0xD2040000 native byte-order)
>8 long 0x11000000 big-endian)

12 long 0x00061561 Berkeley DB (Hash,
>16 long 0x00000004 version 4,
>16 long 0x00000005 version 5,
>16 long 0x00000006 version 6,
>16 long 0x00000007 version 7,
>16 long 0x00000008 version 8,
>16 long 0x00000009 version 9,
>12 long 0x00061561 native byte-order)

12 long 0x61150600 Berkeley DB (Hash,
>16 long 0x04000000 version 4,
>16 long 0x05000000 version 5,
>16 long 0x06000000 version 6,
>16 long 0x07000000 version 7,
>16 long 0x08000000 version 8,
>16 long 0x09000000 version 9,
>12 long 0x61150600 big-endian)

12 long 0x00042253 Berkeley DB (Queue,
>16 long 0x00000001 version 1,
>16 long 0x00000002 version 2,
>16 long 0x00000003 version 3,
>16 long 0x00000004 version 4,
>16 long 0x00000005 version 5,
>16 long 0x00000006 version 6,
>16 long 0x00000007 version 7,
>16 long 0x00000008 version 8,
>16 long 0x00000009 version 9,
>12 long 0x00042253 native byte-order)

12 long 0x53220400 Berkeley DB (Queue,
>16 long 0x01000000 version 1,
>16 long 0x02000000 version 2,
>16 long 0x03000000 version 3,
>16 long 0x04000000 version 4,
>16 long 0x05000000 version 5,
>16 long 0x06000000 version 6,
Building with multiple versions of Berkeley DB

In some cases it may be necessary to build applications which include multiple versions of Berkeley DB. Examples include applications which include software from other vendors, or applications running on a system where the system C library itself uses Berkeley DB. In such cases, the two versions of Berkeley DB may be incompatible, that is, they may have different external and internal interfaces, and may even have different underlying database formats.
To create a Berkeley DB library whose symbols won’t collide with other Berkeley DB libraries (or other application or library modules, for that matter), configure Berkeley DB using the `--with-uniquename=NAME` configuration option, and then build Berkeley DB as usual. (Note that `--with-uniquename=NAME` only affects the Berkeley DB C language library build; loading multiple versions of the C++ or Java APIs will require additional work.) The modified symbol names are hidden from the application in the Berkeley DB header files, that is, there is no need for the application to be aware that it is using a special library build as long as it includes the appropriate Berkeley DB header file.

If “NAME” is not specified when configuring with `--with-uniquename=NAME`, a default value built from the major and minor numbers of the Berkeley DB release will be used. It is rarely necessary to specify NAME; using the major and minor release numbers will ensure that only one copy of the library will be loaded into the application unless two distinct versions really are necessary.

When distributing any library software that uses Berkeley DB, or any software which will be recompiled by users for their systems, we recommend two things: First, include the Berkeley DB release as part of your release. This will insulate your software from potential Berkeley DB API changes as well as simplifying your coding because you will only have to code to a single version of the Berkeley DB API instead of adapting at compile time to whatever version of Berkeley DB happens to be installed on the target system. Second, use `--with-uniquename=NAME` when configuring Berkeley DB, because that will insure that you do not unexpectedly collide with other application code or a library already installed on the target system.
Chapter 3. Debugging Applications

Introduction to debugging

Because Berkeley DB is an embedded library, debugging applications that use Berkeley DB is both harder and easier than debugging a separate server. Debugging can be harder because when a problem arises, it is not always readily apparent whether the problem is in the application, is in the database library, or is a result of an unexpected interaction between the two. Debugging can be easier because it is easier to track down a problem when you can review a stack trace rather than deciphering interprocess communication messages. This chapter is intended to assist you with debugging applications and reporting bugs to us so that we can provide you with the correct answer or fix as quickly as possible.

When you encounter a problem, there are a few general actions you can take:

Review the Berkeley DB error output:
If an error output mechanism has been configured in the Berkeley DB environment, additional run-time error messages are made available to the applications. If you are not using an environment, it is well worth modifying your application to create one so that you can get more detailed error messages. See Run-time error information (page 12) for more information on configuring Berkeley DB to output these error messages.

Review the options available for the DB_ENV->set_verbose() method:
Look to see if it offers any additional informational and/or debugging messages that might help you understand the problem.

Add run-time diagnostics:
You can configure and build Berkeley DB to perform run-time diagnostics. (By default, these checks are not done because they can seriously impact performance.) See Compile-time configuration (page 11) for more information.

Apply all available patches:
Before reporting a problem in Berkeley DB, please upgrade to the latest Berkeley DB release, if possible, or at least make sure you have applied any updates available for your release from the Berkeley DB web site.

Run the test suite:
If you see repeated failures or failures of simple test cases, run the Berkeley DB test suite to determine whether the distribution of Berkeley DB you are using was built and configured correctly.

Compile-time configuration

There are three compile-time configuration options that assist in debugging Berkeley DB and Berkeley DB applications:

--enable-debug
If you want to build Berkeley DB with -g as the C and C++ compiler flag, enter --enable-debug as an argument to configure. This will create Berkeley DB with debugging symbols, as well as load various Berkeley DB routines that can be called directly from a debugger to display database page content, cursor queues, and so forth. (Note that the -O optimization flag will still be specified. To compile with only the -g, explicitly set the CFLAGS environment variable before configuring.)
--enable-diagnostic

If you want to build Berkeley DB with debugging run-time sanity checks and with DIAGNOSTIC #defined during compilation, enter --enable-diagnostic as an argument to configure. This will cause a number of special checks to be performed when Berkeley DB is running. This flag should not be defined when configuring to build production binaries because it degrades performance.

--enable-umrw

When compiling Berkeley DB for use in run-time memory consistency checkers (in particular, programs that look for reads and writes of uninitialized memory), use --enable-umrw as an argument to configure. This guarantees, among other things, that Berkeley DB will completely initialize allocated pages rather than initializing only the minimum necessary amount.

Run-time error information

Normally, when an error occurs in the Berkeley DB library, an integer value (either a Berkeley DB specific value or a system errno value) is returned by Berkeley DB. In some cases, however, this value may be insufficient to completely describe the cause of the error, especially during initial application debugging.

Most Berkeley DB errors will result in additional information being written to a standard file descriptor or output stream. Additionally, Berkeley DB can be configured to pass these verbose error messages to an application function. There are four methods intended to provide applications with additional error information: DB_ENV->set_errcall(), DB_ENV->set_errfile(), DB_ENV->set_errpfx() and DB_ENV->set_verbose().

The Berkeley DB error-reporting facilities do not slow performance or significantly increase application size, and may be run during normal operation as well as during debugging. Where possible, we recommend these options always be configured and the output saved in the filesystem. We have found that this often saves time when debugging installation or other system-integration problems.

In addition, there are three methods to assist applications in displaying their own error messages: db_strerror(), DB_ENV->err(), and DB_ENV->errx(). The first is a superset of the ANSI C strerror function, and returns a descriptive string for any error return from the Berkeley DB library. The DB_ENV->err() and DB_ENV->errx() methods use the error message configuration options described previously to format and display error messages to appropriate output devices.

Reviewing Berkeley DB log files

If you are running with transactions and logging, the db_printlog utility can be a useful debugging aid. The db_printlog utility will display the contents of your log files in a human readable (and machine-readable) format.

The db_printlog utility will attempt to display any and all log files present in a designated db_home directory. For each log record, the db_printlog utility will display a line of the form:

[22][28]db_big: rec: 43 txnid 80000963 prevlsn [21][10483281]
The opening numbers in square brackets are the log sequence number (LSN) of the log record being displayed. The first number indicates the log file in which the record appears, and the second number indicates the offset in that file of the record.

The first character string identifies the particular log operation being reported. The log records corresponding to particular operations are described following. The rest of the line consists of name/value pairs.

The rec field indicates the record type (this is used to dispatch records in the log to appropriate recovery functions).

The txnid field identifies the transaction for which this record was written. A txnid of 0 means that the record was written outside the context of any transaction. You will see these most frequently for checkpoints.

Finally, the prevlsn contains the LSN of the last record for this transaction. By following prevlsn fields, you can accumulate all the updates for a particular transaction. During normal abort processing, this field is used to quickly access all the records for a particular transaction.

After the initial line identifying the record type, each field of the log record is displayed, one item per line. There are several fields that appear in many different records and a few fields that appear only in some records.

The following table presents each currently written log record type with a brief description of the operation it describes. Any of these record types may have the string "_debug" appended if they were written because DB_TXN_NOT_DURABLE was specified and the system was configured with --enable-diagnostic.

<table>
<thead>
<tr>
<th>Log Record Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bam_adj</td>
<td>Used when we insert/remove an index into/from the page header of a Btree page.</td>
</tr>
<tr>
<td>bam_cadjjust</td>
<td>Keeps track of record counts in a Btree or Recno database.</td>
</tr>
<tr>
<td>bam_cdel</td>
<td>Used to mark a record on a page as deleted.</td>
</tr>
<tr>
<td>bam_curadj</td>
<td>Used to adjust a cursor location when a nearby record changes in a Btree database.</td>
</tr>
<tr>
<td>bam_merge</td>
<td>Used to merge two Btree database pages during compaction.</td>
</tr>
<tr>
<td>bam_pgno</td>
<td>Used to replace a page number in a Btree record.</td>
</tr>
<tr>
<td>bam_rcuradj</td>
<td>Used to adjust a cursor location when a nearby record changes in a Recno database.</td>
</tr>
<tr>
<td>bam_relink</td>
<td>Fix leaf page prev/next chain when a page is removed.</td>
</tr>
<tr>
<td>bam_repl</td>
<td>Describes a replace operation on a record.</td>
</tr>
<tr>
<td>bam_root</td>
<td>Describes an assignment of a root page.</td>
</tr>
<tr>
<td>Log Record Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bam_rsplit</td>
<td>Describes a reverse page split.</td>
</tr>
<tr>
<td>bam_split</td>
<td>Describes a page split.</td>
</tr>
<tr>
<td>crdel_inmem_create</td>
<td>Record the creation of an in-memory named database.</td>
</tr>
<tr>
<td>crdel_inmem_remove</td>
<td>Record the removal of an in-memory named database.</td>
</tr>
<tr>
<td>crdel_inmem_rename</td>
<td>Record the rename of an in-memory named database.</td>
</tr>
<tr>
<td>crdel_metasub</td>
<td>Describes the creation of a metadata page for a subdatabase.</td>
</tr>
<tr>
<td>db_addrem</td>
<td>Add or remove an item from a page of duplicates.</td>
</tr>
<tr>
<td>db_big</td>
<td>Add an item to an overflow page (overflow pages contain items too large to place on the main page)</td>
</tr>
<tr>
<td>db_cksum</td>
<td>Unable to checksum a page.</td>
</tr>
<tr>
<td>db_debug</td>
<td>Log debugging message.</td>
</tr>
<tr>
<td>db_noop</td>
<td>This marks an operation that did nothing but update the LSN on a page.</td>
</tr>
<tr>
<td>db_ovref</td>
<td>Increment or decrement the reference count for a big item.</td>
</tr>
<tr>
<td>db_pg_alloc</td>
<td>Indicates we allocated a page to a database.</td>
</tr>
<tr>
<td>db_pg_free</td>
<td>Indicates we freed a page (freed pages are added to a freelist and reused).</td>
</tr>
<tr>
<td>db_pg_freedata</td>
<td>Indicates we freed a page that still contained data entries (freed pages are added to a freelist and reused.)</td>
</tr>
<tr>
<td>db_pg_init</td>
<td>Indicates we reinitialized a page during a truncate.</td>
</tr>
<tr>
<td>db_pg_sort</td>
<td>Sort the free page list and free pages at the end of the file.</td>
</tr>
<tr>
<td>dbreg_register</td>
<td>Records an open of a file (mapping the filename to a log-id that is used in subsequent log operations).</td>
</tr>
<tr>
<td>fop_create</td>
<td>Create a file in the file system.</td>
</tr>
<tr>
<td>fop_file_remove</td>
<td>Remove a name in the file system.</td>
</tr>
<tr>
<td>fop_remove</td>
<td>Remove a file in the file system.</td>
</tr>
<tr>
<td>fop_rename</td>
<td>Rename a file in the file system.</td>
</tr>
<tr>
<td>fop_write</td>
<td>Write bytes to an object in the file system.</td>
</tr>
<tr>
<td>Log Record Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ham_chgpg</td>
<td>Used to adjust a cursor location when a Hash page is removed, and its elements are moved to a different Hash page.</td>
</tr>
<tr>
<td>ham_copypage</td>
<td>Used when we empty a bucket page, but there are overflow pages for the bucket; one needs to be copied back into the actual bucket.</td>
</tr>
<tr>
<td>ham_curadj</td>
<td>Used to adjust a cursor location when a nearby record changes in a Hash database.</td>
</tr>
<tr>
<td>ham_groupalloc</td>
<td>Allocate some number of contiguous pages to the Hash database.</td>
</tr>
<tr>
<td>ham_insdel</td>
<td>Insert/delete an item on a Hash page.</td>
</tr>
<tr>
<td>ham_metagroup</td>
<td>Update the metadata page to reflect the allocation of a sequence of contiguous pages.</td>
</tr>
<tr>
<td>ham_newpage</td>
<td>Adds or removes overflow pages from a Hash bucket.</td>
</tr>
<tr>
<td>ham_replace</td>
<td>Handle updates to records that are on the main page.</td>
</tr>
<tr>
<td>ham_splitdata</td>
<td>Record the page data for a split.</td>
</tr>
<tr>
<td>heap_addrem</td>
<td>Add or remove an entry from a Heap database.</td>
</tr>
<tr>
<td>heap_pg_alloc</td>
<td>Indicates we allocated a page to a Heap database.</td>
</tr>
<tr>
<td>heap_trunc_meta</td>
<td>Records the truncation of the meta page in a Heap database.</td>
</tr>
<tr>
<td>heap_trunc_page</td>
<td>Records the truncation of a data page in a Heap database.</td>
</tr>
<tr>
<td>qam_add</td>
<td>Describes the actual addition of a new record to a Queue.</td>
</tr>
<tr>
<td>qam_del</td>
<td>Delete a record in a Queue.</td>
</tr>
<tr>
<td>qam_delext</td>
<td>Delete a record in a Queue with extents.</td>
</tr>
<tr>
<td>qam_incfirst</td>
<td>Increments the record number that refers to the first record in the database.</td>
</tr>
<tr>
<td>qam_mvptr</td>
<td>Indicates we changed the reference to either or both of the first and current records in the file.</td>
</tr>
<tr>
<td>txn_child</td>
<td>Commit a child transaction.</td>
</tr>
<tr>
<td>txn_ckp</td>
<td>Transaction checkpoint.</td>
</tr>
<tr>
<td>txn_recycle</td>
<td>Transaction IDs wrapped.</td>
</tr>
</tbody>
</table>
Log Record Type | Description
--- | ---
txn_regop | Logs a regular (non-child) transaction commit.
txn_xa_regop | Logs a prepare message.

### Augmenting the Log for Debugging

When debugging applications, it is sometimes useful to log not only the actual operations that modify pages, but also the underlying Berkeley DB functions being executed. This form of logging can add significant bulk to your log, but can permit debugging application errors that are almost impossible to find any other way. To turn on these log messages, specify the `--enable-debug_rop` and `--enable-debug_wop` configuration options when configuring Berkeley DB. See Configuring Berkeley DB (page 30) for more information.

### Extracting Committed Transactions and Transaction Status

Sometimes, it is helpful to use the human-readable log output to determine which transactions committed and aborted. The awk script, `commit.awk`, (found in the `db_printlog` directory of the Berkeley DB distribution) allows you to do just that. The following command, where `log_output` is the output of `db_printlog`, will display a list of the transaction IDs of all committed transactions found in the log:

```
awk -f commit.awk log_output
```

If you need a complete list of both committed and aborted transactions, then the script `status.awk` will produce it. The syntax is as follows:

```
awk -f status.awk log_output
```

### Extracting Transaction Histories

Another useful debugging aid is to print out the complete history of a transaction. The awk script `txn.awk` allows you to do that. The following command line, where `log_output` is the output of `db_printlog` utility and `txnlist` is a comma-separated list of transaction IDs, will display all log records associated with the designated transaction ids:

```
awk -f txn.awk TXN=txnlist log_output
```

### Extracting File Histories

The awk script `fileid.awk` allows you to extract all log records that refer to a designated file. The syntax for the `fileid.awk` script is the following, where `log_output` is the output of `db_printlog` and `fids` is a comma-separated list of fileids:

```
awk -f fileid.awk PGNO=fids log_output
```

### Extracting Page Histories

The awk script `pgno.awk` allows you to extract all log records that refer to designated page numbers. However, because this script will extract records with the designated page numbers for all files, it is most useful in conjunction with the `fileid` script. The syntax for the `pgno.awk` script is the following, where `log_output` is the output of `db_printlog` and `pgnolist` is a comma-separated list of page numbers:
Other log processing tools

The awk script count.awk prints out the number of log records encountered that belonged to some transaction (that is, the number of log records excluding those for checkpoints and non-transaction-protected operations).

The script range.awk will extract a subset of a log. This is useful when the output of db_printlog utility is too large to be reasonably manipulated with an editor or other tool. The syntax for range.awk is the following, where \( sf \) and \( so \) represent the LSN of the beginning of the sublog you want to extract, and \( ef \) and \( eo \) represent the LSN of the end of the sublog you want to extract:

```bash
awk -f range.awk START_FILE=sf START_OFFSET=so END_FILE=ef \ END_OFFSET=eo log_output
```
Chapter 4. Building Berkeley DB for Windows

This chapter contains general instructions on building Berkeley DB for specific windows platforms using specific compilers. The Windows FAQ (page 28) also contains helpful information.

The build_windows directory in the Berkeley DB distribution contains project files for Microsoft Visual Studio:

<table>
<thead>
<tr>
<th>Project File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berkeley_DB_vs2010.sln</td>
<td>Visual Studio 2010 workspace</td>
</tr>
<tr>
<td>VS10*.vcxproj</td>
<td>Visual Studio 2010 projects</td>
</tr>
<tr>
<td>Berkeley_DB_vs2012.sln</td>
<td>Visual Studio 2012 workspace</td>
</tr>
<tr>
<td>VS12*.vcxproj</td>
<td>Visual Studio 2012 projects</td>
</tr>
<tr>
<td>Berkeley_DB_vs2015.sln</td>
<td>Visual Studio 2015 workspace</td>
</tr>
<tr>
<td>VS15*.vcxproj</td>
<td>Visual Studio 2015 projects</td>
</tr>
</tbody>
</table>

These project files can be used to build Berkeley DB for Windows Server 2008, Windows Vista, Windows 7, Windows Server 2008 R2, and Windows 8.

Building Berkeley DB for Windows

Visual C++ .NET 2015

1. Choose File -> Open -> Project/Solution.... In the build_windows directory, select Berkeley_DB_vs2015.sln and click Open.

2. Choose the desired project configuration from the drop-down menu on the tool bar (either Debug or Release).

3. Choose the desired platform configuration from the drop-down menu on the tool bar.

4. To build, right-click on the Berkeley_DB_vs2015 solution and select Build Solution.

Visual C++ .NET 2013


2. Choose File -> Open -> Project/Solution.... In the build_windows directory, select Berkeley_DB_vs2012.sln and click Open. The Upgrade VC++ Compiler and Libraries dialog box will appear. Click OK to upgrade.

3. Choose the desired project configuration from the drop-down menu on the tool bar (either Debug or Release).

4. Choose the desired platform configuration from the drop-down menu on the tool bar.

5. To build, right-click on the Berkeley_DB_vs2012 solution and select Build Solution.
**Visual C++ .NET 2012**

1. Choose *File -> Open -> Project/Solution....* In the build_windows directory, select Berkeley_DB_vs2012.sln and click Open.

2. Choose the desired project configuration from the drop-down menu on the tool bar (either Debug or Release).

3. Choose the desired platform configuration from the drop-down menu on the tool bar (usually Win32 or x64).

4. To build, right-click on the Berkeley_DB_vs2012 solution and select Build Solution.

**Visual C++ .NET 2010**

1. Choose *File -> Open -> Project/Solution....* In the build_windows directory, select Berkeley_DB_vs2010.sln and click Open.

2. Choose the desired project configuration from the drop-down menu on the tool bar (either Debug or Release).

3. Choose the desired platform configuration from the drop-down menu on the tool bar (usually Win32 or x64).

4. To build, right-click on the Berkeley_DB_vs2010 solution and select Build Solution.

**Build results**

The results of your build will be placed in one of the following Berkeley DB subdirectories, depending on the configuration that you chose:

build_windows\x64\Debug
build_windows\x64\Release
build_windows\x64\Debug_static
build_windows\x64\Release_static
build_windows\Win32\Debug
build_windows\Win32\Release
build_windows\Win32\Debug_static
build_windows\Win32\Release_static

When building your application during development, you should normally use compile options "Debug Multithreaded DLL" and link against build_windows\Debug\libdb181d.lib.

**Note**

Building your application with the debug configuration will create a Berkeley DB library and utilities with debugging symbols, as well as load various routines that can be called from a debugger to display pages, cursor queues, and so forth. If installed, the utilities will not be stripped. This configuration should not be specified when building production binaries.
You can also build using a release version of the Berkeley DB libraries and tools, which will be placed in \build_windows\Win32\Release\libdb181.lib. When linking against the release build, you should compile your code with the "Release Multithreaded DLL" compile option. You will also need to add the build_windows directory to the list of include directories of your application's project, or copy the Berkeley DB include files to another location.

**Building Berkeley DB with Cygwin**

To build Berkeley DB with Cygwin, follow the instructions in Building for UNIX/POSIX (page 30).

**Building the C++ API**

C++ support is built automatically on Windows.

**Building the C++ STL API**

In the project list of the appropriate Berkeley_DB_vx20xx.sln solution, build the "db_stl" project and "db_stl_static" project to build STL API as a dynamic or static library respectively. And in your application, you should link this library file as well as the Berkeley DB library file to your application. The STL API library file is by default always put at the same location as the Berkeley DB library file.

And you need to include the STL API header files in your application code. If you are using the Berkeley DB source tree, the header files are in <Berkeley DB Source Root>/stl directory; If you are using the pre-built installed version, these header files are in < Berkeley DB Installed Directory>/include, as well as the db.h and db_cxx.h header files.

**Building the Java API**

Java support is not built automatically. The following instructions assume that you have installed the Sun Java Development Kit in d:\java. Of course, if you installed elsewhere or have different Java software, you will need to adjust the pathnames accordingly.

1. Set your include directories.
   - In Visual Studio 2010/2012/2013/2015 - Right-click db_java project, choose Properties->Configuration Properties-> VC++ Directories->Include Directories. Add the full pathnames for the d:\java\include and d:\java\include\win32 directories. Then click OK.
   
   These are the directories needed when including jni.h.

2. Set the executable files directories.
   - In Visual Studio 2010/2012/2013/2015 - Right-click db_java project, choose Properties->Configuration Properties-> VC++ Directories->Executable Directories. Add the full pathnames for the d:\java\bin directories. Then click OK.
   
   This is the directory needed to find javac.

3. Set the build type to Release or Debug in the drop-down on the tool bar.
4. To build, right-click on db_java and select Build. This builds the Java support library for Berkeley DB and compiles all the java files, placing the resulting db.jar and dbexamples.jar files in one of the following Berkeley DB subdirectories, depending on the configuration that you chose:

build_windows\Win32\Debug
build_windows\Win32\Release
build_windows\x64\Debug
build_windows\x64\Release

**Building the C# API**

The C# support is built by a separate Visual Studio solution and requires version 2.0 (or higher) of the .NET platform. If the Berkeley DB utilities are required, build Berkeley DB first following the instructions under Building Berkeley DB for Windows (page 18).

To build the C# API in Visual Studio 2010 the solution is build_windows \BDB_dotnet_vs2010.sln; in Visual Studio 2012 and 2013 the solution is build_windows \BDB_dotnet_vs2012.sln; and in Visual Studio 2015 the solution is build_windows \BDB_dotnet_vs2015.sln.

By default, the solution will build the native libraries, the managed assembly and all example programs. The NUnit tests need to be built explicitly because of their dependence upon the NUnit assembly. The native libraries will be placed in one of the following subdirectories, depending upon the chosen configuration:

build_windows\x64\Debug
build_windows\x64\Release
build_windows\Win32\Debug
build_windows\Win32\Release

The managed assembly and all C# example programs will be placed in one of the following subdirectories, depending upon the chosen configuration:

build_windows\AnyCPU\Debug
build_windows\AnyCPU\Release

The native libraries need to be locatable by the .NET platform, meaning they must be copied into an application's directory, the Windows or System directory, or their location must be added to the PATH environment variable. The example programs demonstrate how to programmatically edit the PATH variable.

**Building the Tcl API**

Tcl support is not built automatically. See Loading Berkeley DB with Tcl for information on sites from which you can download Tcl and which Tcl versions are compatible with Berkeley DB. These notes assume that Tcl is installed as d:\tcl, but you can change that if you want.

The Tcl library must be built as the same build type as the Berkeley DB library (both Release or both Debug). We found that the binary release of Tcl can be used with the Release configuration of Berkeley DB, but you will need to build Tcl from sources for the Debug
configuration. Before building Tcl, you will need to modify its makefile to make sure that you are building a debug version, including thread support. This is because the set of DLLs linked into the Tcl executable must match the corresponding set of DLLs used by Berkeley DB.

1. Set the include directories.
   - In Visual Studio 2010/2012/2013/2015 - Right-click the db_tcl project, choose Properties->Configuration Properties-> VC++ Directories->Include Directories. Add the full pathnames for d:\tcl\include, then click OK.

   This is the directory that contains tcl.h.

2. Set the library files directory.
   - In Visual Studio 2010/2012/2013/2015 - Right-click the db_tcl project, choose Properties->Configuration Properties-> VC++ Directories->Library Directories. Add the full pathname for the d:\tcl\lib directory, then click OK.

   This is the directory needed to find tcl86.lib (or whatever the library is named in your distribution).

3. Set the build type to Release or Debug in the drop-down on the tool bar.

4. To build, right-click on db_tcl and select Build. This builds the Tcl support library for Berkeley DB, placing the result into one of the following Berkeley DB subdirectories, depending upon the configuration that you chose:

   build_windows\x64\Debug\libdb_tcl181d.dll  
   build_windows\x64\Release\libdb_tcl181.dll  
   build_windows\Win32\Debug\libdb_tcl181d.dll  
   build_windows\Win32\Release\libdb_tcl181.dll  

   If you use a version different from Tcl 8.6.x you will need to change the name of the Tcl library used in the build (for example, tcl86.lib) to the appropriate name. To do this, right click on db_tcl, go to Properties -> Linker -> Input -> Additional dependencies and change the filename to match the Tcl version you are using.

**Building Secure Sockets Layer (SSL) Support for the Replication Manager**

Berkeley DB provides SSL support for the Replication Manager via OpenSSL-library to enable secure communication among Replication Manager group members. For more information on OpenSSL, see [https://www.openssl.org/](https://www.openssl.org/).

SSL support for replication manager is not built by default. If you want to build it then you have to define HAVE_REPMGR_SSL_SUPPORT in db_config.h (in build_windows folder) and provide path to SSL libraries and headers for each project as specified below.

**Note**

You must use OpenSSL-library versions greater than or equal to 1.0.1 that includes thread support (see [https://www.openssl.org/blog/blog/2017/02/21/threads/](https://www.openssl.org/blog/blog/2017/02/21/threads/)). Unless this condition is met, the build will fail.
To build Berkeley DB with OpenSSL support, follow the steps below:

Note that the following steps assume that you have installed OpenSSL in D:\OpenSSL-Win64. You can, however, change the location if you want. Also note that you would use Visual Studio (2010/2012/2013/2015) to perform the steps mentioned below.

1. Set Additional Include Directories.
   a. Right-click db project and choose Properties->Configuration Properties-> c/c++->General-> Additional Include Directories.
   b. Add the full pathnames for D:\OpenSSL-Win64\include.
   c. Click OK.

   This directory contains the following header files: openssl\ssl.h, openssl\err.h and openssl\opensslconf.h.

2. Specify the library files and their corresponding directories.
   b. Add the full pathnames for D:\OpenSSL-Win64\lib.
   c. Click OK.

   This directory is needed to find the libeay32.lib and ssleay32.lib library files.

3. Specify which libraries are to be linked during the build.
   b. Add the ssleay32.lib and libeay32.lib library names to the existing list of dependencies.

4. Set the build type to Release or Debug in the drop-down on the tool bar.

5. Build Berkeley DB by right-clicking db and select Build. This builds the Berkeley DB project with SSL support for the Replication Manager.

Note that you must follow the same instructions to build other projects like db_tcl or db_test.

Note

- To build Berkeley DB without OpenSSL support, remove the definition for the HAVE_REPMGR_SSL_SUPPORT in db_config.h configuration files under the build_windows directory.
• Once you have built Berkeley DB with SSL support for the Replication Manager, you can disable the use of SSL for Replication Manager communication by setting the flag `DB_REPMGR_CONF_DISABLE_SSL` via `DB_ENV->REP_SET_CONFIG` API.

Distributing DLLs

When distributing applications linked against the DLL (not static) version of the library, the DLL files you need will be found in one of the following Berkeley DB subdirectories, depending upon the configuration that you chose:

```
build_windows\Win32\Debug
build_windows\Win32\Release
build_windows\Win32\Debug_static
build_windows\Win32\Release_static
build_windows\x64\Debug
build_windows\x64\Release
build_windows\x64\Debug_static
build_windows\x64\Release_static
```

You may also need to redistribute DLL files needed for the compiler’s runtime. Generally, these runtime DLL files can be installed in the same directory that will contain your installed Berkeley DB DLLs. This directory may need to be added to your System PATH environment variable. Check your compiler’s license and documentation for specifics on redistributing runtime DLLs.

Additional build options

There are several build options that you can configure when building Berkeley DB on Windows. To specify these, select `Project Properties` -> `C/C++` -> `Command Line` and add the property.

These are some of the additional properties that you can specify when you are building Berkeley DB on Windows:

• Enable failchk broadcasting. This feature allows `DB_ENV->failchk()` to inform all processes in the environment that a failure has occurred. Those processes will return `DB_RUNRECOVERY` rather than hanging for an indefinite amount of time.

• `/D HAVE_LOCALIZATION`

Enable localized error message text, if available. This option should not be used when enabling stripped messages.

• `/D HAVE_MIXED_SIZE_ADDRESSING`

Allows for the sharing of the BDB database environment between 32-bit and 64-bit applications. Note that if you use this macro to rebuild your Berkeley DB library, then you need to also rebuild both your 32-bit and 64-bit applications using `/D HAVE_MIXED_SIZE_ADDRESSING`.

Note that use of this macro means that private environments are disabled for the library.
• /D HAVE_STRIPPED_MESSAGES

Causes all error messages to be stripped of their textual information. This option should not be used when enabling localization support. Use of this property can reduce your library footprint by up to 42KB (for DLLs) or 98KB (for a .lib).

Note that this option is automatically enabled if you build using the db_small project name. For more information on building a small library, see Building a small memory footprint library (page 25).

If you use this build option, you can get an idea of what text should be issued for a given error message by using the Berkeley DB Message Reference Guide.

• /D DIAGNOSTIC

Included automatically in the Debug build. Allows a number of additional checks to be performed when BDB is running, and also causes some failures to trigger process abort rather than returning errors to the application. Applications built using this option should not share database environments with applications built without this option. The DIAGNOSTIC option should not be specified when configuring to build production binaries because it degrades performance.

• /D UMRW

Rational Software's Purify product and other software analysis run-time tools complain about uninitialized reads/writes of structure fields whose only purpose is padding, as well as when heap memory that was never initialized is written to disk. Specify this option during configuration to mask these errors. The UMRW option should not be specified when configuring to build production binaries.

This flag can also be used to prevent data in the heap from leaking onto unused parts of database pages when using the environment flag DB_PRIVATE.

Building a small memory footprint library

For applications that don't require all of the functionality of the full Berkeley DB library, an option is provided to build a static library with certain functionality disabled. In particular, cryptography, hash, heap and queue access methods, replication and verification are all turned off. In addition, all message text is stripped from the library. This can reduce the memory footprint of Berkeley DB significantly.

Note

If your library has stripped messages, you can get an idea of what text should be issued for a given error message by using the Berkeley DB Message Reference Guide.

In general on Windows systems, you will want to evaluate the size of the final application, not the library build. The Microsoft LIB file format (like UNIX archives) includes copies of all of the object files and additional information. The linker rearranges symbols and strips out the overhead, and the resulting application is much smaller than the library. There is also a Visual C++ optimization to "Minimize size" that will reduce the library size by a few percent.
A Visual C++ project file called db_small is provided for this small memory configuration. During a build, static libraries are created in Release or Debug, respectively. The library name is libdb_small181s1d.lib for the debug build, or libdb_small181s.lib for the release build.

For assistance in further reducing the size of the Berkeley DB library, or in building small memory footprint libraries on other systems, please contact Berkeley DB support.

Running the test suite under Windows

To build the test suite on Windows platforms, you will need to configure Tcl support. You will also need sufficient main memory (at least 64MB), and disk (around 250MB of disk will be sufficient).

Building the software needed by the tests

The test suite must be run against a Debug version of Berkeley DB, so you will need a Debug version of the Tcl libraries. This involves building Tcl from its source. See the Tcl sources for more information. Then build the Tcl API - see Building the Tcl API (page 21) for details.

Visual Studio 2010 or newer

To build for testing, perform the following steps:

1. Open the Berkeley DB solution.
2. Ensure that the target configuration is Debug
3. Right click the db_tcl project in the Solution Explorer, and select Build.
4. Right click the db_test project in the Solution Explorer, and select Build.

Running the test suite under Windows

Before running the tests for the first time, you must edit the file include.tcl in your build directory and change the line that reads:

```
set tclsh_path SET_YOUR_TCLSH_PATH
```

You will want to use the location of the tclsh program (be sure to include the name of the executable). For example, if Tcl is installed in d:\tcl, this line should be the following:

```
set tclsh_path d:\tcl\bin\tclsh86.exe
```

If your path includes spaces be sure to enclose it in quotes:

```
set tclsh_path "c:\Program Files\tcl\bin\tclsh86.exe"
```

Make sure that the path to Berkeley DB’s tcl library is in your current path. Edit your PATH using the Control Panel -> System -> Advanced system settings -> Environment Variables dialog. On earlier versions of Windows, you may find it convenient to add a line to c:\AUTOEXEC.BAT:
SET PATH=%PATH%;c:\db\build_windows

Then, in a shell of your choice enter the following commands:

1. cd build_windows

2. run d:\tcl\bin\tclsh86.exe, or the equivalent name of the Tcl shell for your system.
   You should get a "%" prompt.

3. % source ../test/tcl/test.tcl
   If no errors occur, you should get a "%" prompt.

You are now ready to run tests in the test suite; see Running the test suite for more information.

Windows notes

If a system memory environment is closed by all processes, subsequent attempts to open it will return an error. To successfully open a transactional environment in this state, recovery must be run by the next process to open the environment. For non-transactional environments, applications should remove the existing environment and then create a new database environment.

1. Berkeley DB does not support the Windows XP, Windows 2003 or earlier Windows platforms.

2. On Windows, system paging file memory is freed on last close. For this reason, multiple processes sharing a database environment created using the DB_SYSTEM_MEM flag must arrange for at least one process to always have the environment open, or alternatively that any process joining the environment be prepared to re-create it.

3. When using the DB_SYSTEM_MEM flag, Berkeley DB shared regions are created without ACLs, which means that the regions are only accessible to a single user. If wider sharing is appropriate (for example, both user applications and Windows/NT service applications need to access the Berkeley DB regions), the Berkeley DB code will need to be modified to create the shared regions with the correct ACLs. Alternatively, by not specifying the DB_SYSTEM_MEM flag, filesystem-backed regions will be created instead, and the permissions on those files may be directly specified through the DB_ENV->open() method.

4. Applications that operate on wide character strings can use the Windows function WideCharToMultiByte with the code page CP_UTF8 to convert paths to the form expected by Berkeley DB. Internally, Berkeley DB calls MultiByteToWideChar on paths before calling Windows functions.

5. Various Berkeley DB methods take a mode argument, which is intended to specify the underlying file permissions for created files. Berkeley DB currently ignores this argument on Windows systems.

It would be possible to construct a set of security attributes to pass to CreateFile that accurately represents the mode. In the worst case, this would involve looking up user and all group names, and creating an entry for each. Alternatively, we could call the _chmod
(partial emulation) function after file creation, although this leaves us with an obvious race.

Practically speaking, however, these efforts would be largely meaningless on a FAT file system, which only has a “readable” and “writable” flag, applying to all users.

**Windows FAQ**

1. **My Win* C/C++ application crashes in the Berkeley DB library when Berkeley DB calls fprintf (or some other standard C library function).**

   You should be using the “Debug Multithreaded DLL” compiler option in your application when you link with the build_windows\Debug\libdb60d.lib library (this .lib file is actually a stub for libdb60d.DLL). To check this setting in Visual C++, choose the Project/Settings menu item and select Code Generation under the tab marked C/C++; and see the box marked Use runtime library. This should be set to Debug Multithreaded DLL. If your application is linked against the static library, build_windows\Debug\libdb60sd.lib; then, you will want to set Use runtime library to Debug Multithreaded.

   Setting this option incorrectly can cause multiple versions of the standard libraries to be linked into your application (one on behalf of your application, and one on behalf of the Berkeley DB library). That violates assumptions made by these libraries, and traps can result.

   Also, using different Visual Studio compilers in the application and libraries can lead to a crash. So rebuild the application with the same Visual C++ version as that of the library.

2. **Why are the build options for DB_DLL marked as “Use MFC in a Shared DLL”? Does Berkeley DB use MFC?**

   Berkeley DB does not use MFC at all. It does however, call malloc and free and other facilities provided by the Microsoft C runtime library. We found in our work that many applications and libraries are built assuming MFC, and specifying this for Berkeley DB solves various interoperability issues, and guarantees that the right runtime libraries are selected. Note that because we do not use MFC facilities, the MFC library DLL is not marked as a dependency for libdb.dll, but the appropriate Microsoft C runtime is.

3. **How can I build Berkeley DB for MinGW?**

   Follow the instructions in Building for UNIX/POSIX (page 30), and specify the --enable-mingw option to the configuration script. This configuration option currently only builds static versions of the library, it does not yet build a DLL version of the library, and file sizes are limited to 2GB (2^32 bytes.)

4. **How can I build a Berkeley DB for Windows 98/ME?**

   Windows 98/ME is no longer supported by Berkeley DB. The following is therefore only of interest to historical users of Berkeley DB.

   By default on Windows, Berkeley DB supports internationalized filenames by treating all directory paths and filenames passed to Berkeley DB methods as UTF-8 encoded strings.
All paths are internally converted to wide character strings and passed to the wide character variants of Windows system calls.

This allows applications to create and open databases with names that cannot be represented with ASCII names while maintaining compatibility with applications that work purely with ASCII paths.

Windows 98 and ME do not support Unicode paths directly. To build for those versions of Windows, either:

- Follow the instructions at Microsoft’s web site.

- Open the workspace or solution file with Visual Studio. Then open the Project properties/settings section for the project you need to build (at least db_dll). In the C/C++\Preprocessor->Preprocessor Definitions section, remove _UNICODE and UNICODE entries. Add in an entry of _MBCS. Build the project as normal.

The ASCII builds will also work on newer versions of Windows, but will not translate paths to wide character strings.
Chapter 5. Building Berkeley DB for UNIX/POSIX

Building for UNIX/POSIX

The Berkeley DB distribution builds up to six separate libraries: the base C API Berkeley DB library and the optional C++, Java, and Tcl. For portability reasons, each library is standalone and contains the full Berkeley DB support necessary to build applications; that is, the C++ API Berkeley DB library does not require any other Berkeley DB libraries to build and run C++ applications.

Building for Linux, Apple iOS (known as iPhone OS previously), or Mac OS X is the same as building for a conventional UNIX platform.

The Berkeley DB distribution uses the Free Software Foundation’s autoconf and libtool tools to build on UNIX platforms. In general, the standard configuration and installation options for these tools apply to the Berkeley DB distribution.

To perform a standard UNIX build of Berkeley DB, change to the build_unix directory and then enter the following two commands:

```
../dist/configure
make
```

This will build the Berkeley DB library.

To install the Berkeley DB library, enter the following command:

```
make install
```

To rebuild Berkeley DB, enter:

```
make clean
make
```

If you change your mind about how Berkeley DB is to be configured, you must start from scratch by entering the following command:

```
make realclean
../dist/configure
make
```

To uninstall Berkeley DB, enter:

```
make uninstall
```

To build multiple UNIX versions of Berkeley DB in the same source tree, create a new directory at the same level as the build_unix directory, and then configure and build in that directory as described previously.

Configuring Berkeley DB

There are several arguments you can specify when configuring Berkeley DB. Although only the Berkeley DB-specific ones are described here, most of the standard GNU autoconf arguments
are available and supported. To see a complete list of possible arguments, specify the `--help` flag to the configure program.

The Berkeley DB specific arguments are as follows:

- **--disable-largefile**

  Some systems, notably versions of Solaris, require special compile-time options in order to create files larger than $2^{32}$ bytes. These options are automatically enabled when Berkeley DB is compiled. For this reason, binaries built on current versions of these systems may not run on earlier versions of the system because the library and system calls necessary for large files are not available. To disable building with these compile-time options, enter `--disable-largefile` as an argument to configure.

- **--disable-shared, --disable-static**

  On systems supporting shared libraries, Berkeley DB builds both static and shared libraries by default. (Shared libraries are built using the GNU Project’s Libtool distribution, which supports shared library builds on many (although not all) systems.) To not build shared libraries, configure using the `--disable-shared` argument. To not build static libraries, configure using the `--disable-static` argument.

- **--disable-heap**

  Disables the Heap access method so that it cannot be used by Berkeley DB applications.

- **--enable-compat185**

  To compile or load Berkeley DB 1.85 applications against this release of the Berkeley DB library, enter `--enable-compat185` as an argument to configure. This will include Berkeley DB 1.85 API compatibility code in the library.

- **--enable-cxx**

  To build the Berkeley DB C++ API, enter `--enable-cxx` as an argument to configure.

- **--enable-debug**

  To build Berkeley DB with `-g` as a compiler flag and with `DEBUG` #defined during compilation, enter `--enable-debug` as an argument to configure. This will create a Berkeley DB library and utilities with debugging symbols, as well as load various routines that can be called from a debugger to display pages, cursor queues, and so forth. If installed, the utilities will not be stripped. This argument should not be specified when configuring to build production binaries.

- **--enable-debug_rop**

  To build Berkeley DB to output log records for read operations, enter `--enable-debug_rop` as an argument to configure. This argument should not be specified when configuring to build production binaries.

- **--enable-debug_wop**
To build Berkeley DB to output log records for write operations, enter `--enable-debug_wop` as an argument to configure. This argument should not be specified when configuring to build production binaries.

- **--enable-diagnostic**

To build Berkeley DB with run-time debugging checks, enter `--enable-diagnostic` as an argument to configure. This causes a number of additional checks to be performed when Berkeley DB is running, and also causes some failures to trigger process abort rather than returning errors to the application. Applications built using this argument should not share database environments with applications built without this argument. This argument should not be specified when configuring to build production binaries.

- **--enable-dump185**

To convert Berkeley DB 1.85 (or earlier) databases to this release of Berkeley DB, enter `--enable-dump185` as an argument to configure. This will build the `db_dump185` utility, which can dump Berkeley DB 1.85 and 1.86 databases in a format readable by the Berkeley DB `db_load` utility.

The system libraries with which you are loading the `db_dump185` utility must already contain the Berkeley DB 1.85 library routines for this to work because the Berkeley DB distribution does not include them. If you are using a non-standard library for the Berkeley DB 1.85 library routines, you will have to change the Makefile that the configuration step creates to load the `db_dump185` utility with that library.

- **--enable-error_history**

Causes more detailed error messages to be displayed for most error messages. The additional information includes the process and thread identifiers, timestamp, and certain previous unusual conditions detected during the API call. Stack traces are also displayed, where they are supported.

This option is available only for Unix-like platforms which support the `pthread_getspecific()` thread local storage functions.

- **--enable-failchk_broadcast**

Enable the mode of the `DB_ENV->failchk()` feature which causes processes in the environment to be notified of an application failure as soon as possible. When this feature is enabled threads waiting on a mutex owned by a crashed process will return a `DB_RUNRECOVERY` error. Without this feature, those processes may hang until intervention by some other entity, such as a “watcher process” as mentioned in Architecting Transactional Data Store applications or a user interrupting the program.

- **--enable-java**

To build the Berkeley DB Java API, enter `--enable-java` as an argument to configure. To build Java, you must also build with shared libraries. Before configuring, you must set your PATH environment variable to include javac. Note that it is not sufficient to include a symbolic
link to javac in your PATH because the configuration process uses the location of javac to determine the location of the Java include files (for example, jni.h). On some systems, additional include directories may be needed to process jni.h; see Changing compile or load options (page 39) for more information.

Note
The --enable-java option may not be specified at the same time as --enable-smallbuild or --disable-statistics.

• --enable-posixmutexes

To force Berkeley DB to use the POSIX pthread mutex interfaces for underlying mutex support, enter --enable-posixmutexes as an argument to configure. This is rarely necessary: POSIX mutexes will be selected automatically on systems where they are the preferred implementation.

The --enable-posixmutexes configuration argument is normally used in two ways:
First, when there are multiple mutex implementations available and the POSIX mutex implementation is not the preferred one (for example, on Solaris where the LWP mutexes are used by default). Second, by default the Berkeley DB library will only select the POSIX mutex implementation if it supports mutexes shared between multiple processes, as described for the pthread_condattr_setpshared and pthread_mutexattr_setpshared interfaces. The --enable-posixmutexes configuration argument can be used to force the selection of POSIX mutexes in this case, which can improve application performance significantly when the alternative mutex implementation is a non-blocking one (for example test-and-set assembly instructions). However, configuring to use POSIX mutexes when the implementation does not have inter-process support will only allow the creation of private database environments, that is, environments where the DB_PRIVATE flag is specified to the DB_ENV->open() method.

Specifying the --enable-posixmutexes configuration argument may require that applications and Berkeley DB be linked with the -lpthread library.

• --enable-pthread_api

To configure Berkeley DB for a POSIX pthreads application (with the exception that POSIX pthread mutexes may not be selected as the underlying mutex implementation for the build), enter --enable-pthread_api as an argument to configure. The build will include the Berkeley DB replication manager interfaces and will use the POSIX standard pthread_self and pthread_yield functions to identify threads of control and yield the processor. The --enable-pthread_api argument requires POSIX pthread support already be installed on your system.

Specifying the --enable-pthread_api configuration argument may require that applications and Berkeley DB be linked with the -lpthread library.

• --enable-slices

Enable environment and database slice support.

• --enable-smallbuild
To build a small memory footprint version of the Berkeley DB library, enter --enable-smallbuild as an argument to configure. The --enable-smallbuild argument is equivalent to individually specifying --disable-compression, --with-cryptography=no, --disable-hash, --disable-heap, --disable-queue, --disable-partition, --disable-replication, --disable-statistics and --disable-verify, turning off compression and cryptography support, the Hash, Heap and Queue access methods, support for partitioning, database environment replication support and database and log verification support. See Building a small memory footprint library (page 37) for more information.

--enable-stl

To build the Berkeley DB C++ STL API, enter --enable-stl as an argument to configure. Setting this argument implies that --enable-cxx is set, and the Berkeley DB C++ API will be built too.

There will be a libdb_stl-X.X.a and libdb_stl-X.X.so built, which are the static and shared library you should link your application with in order to make use of Berkeley DB via its STL API.

If your compiler is not ISO C++ compliant, the configure may fail with this argument specified because the STL API requires standard C++ template features. In this case, you will need a standard C++ compiler. So far gcc is the best choice, we have tested and found that gcc-3.4.4 and all its newer versions can build the Berkeley DB C++ STL API successfully.

For information on db_stl supported compilers, see the Portability section in the Programmer’s Reference Guide.

And you need to include the STL API header files in your application code. If you are using the Berkeley DB source tree, the header files are in <Berkeley DB Source Root>/stl directory; If you are using the installed version, these header files are in < Berkeley DB Installed Directory>/include, as well as the db.h and db_cxx.h header files.

--enable-tcl

To build the Berkeley DB Tcl API, enter --enable-tcl as an argument to configure. This configuration argument expects to find Tcl's tclConfig.sh file in the /usr/local/lib directory. See the --with-tcl argument for instructions on specifying a non-standard location for the Tcl installation. See Loading Berkeley DB with Tcl for information on sites from which you can download Tcl and which Tcl versions are compatible with Berkeley DB. To build Tcl, you must also build with shared libraries.

--enable-test

To build the Berkeley DB test suite, enter --enable-test as an argument to configure. To run the Berkeley DB test suite, you must also build the Tcl API. This argument should not be specified when configuring to build production binaries.

--enable-uimutexes
To force Berkeley DB to use the UNIX International (UI) mutex interfaces for underlying mutex support, enter --enable-uimutexes as an argument to configure. This is rarely necessary: UI mutexes will be selected automatically on systems where they are the preferred implementation.

The --enable-uimutexes configuration argument is normally used when there are multiple mutex implementations available and the UI mutex implementation is not the preferred one (for example, on Solaris where the LWP mutexes are used by default).

Specifying the --enable-uimutexes configuration argument may require that applications and Berkeley DB be linked with the -lthread library.

• --enable-umrw

Rational Software's Purify product and other run-time tools complain about uninitialized reads/writes of structure fields whose only purpose is padding, as well as when heap memory that was never initialized is written to disk. Specify the --enable-umrw argument during configuration to mask these errors. This argument should not be specified when configuring to build production binaries.

This flag can also be used to prevent data in the heap from leaking onto unused parts of database pages when using the environment flag DB_PRIVATE.

• --enable-dtrace [--enable-perfmon-statistics]

To build Berkeley DB with performance event monitoring probes add --enable-dtrace to the configuration options. Both native DTrace (on Solaris and Mac OS X) and the Statically Defined Tracing compatibility layer in Linux SystemTap version 1.1 or better are supported. That compatibility package may be called systemtap-sdt-devel; it includes sys/sdt.h.

If --enable-perfmon-statistics is combined with --enable-dtrace then additional probes are defined for the tracking variables from which DB's statistics are obtained. They allow DTrace and SystemTap access to these values when they are updated, are the basis of the statistics as displayed db_stat and the API functions that return statistics.

The --enable-dtrace option may not be specified at the same time as --disable-statistics.

For information on Berkeley DB Performance Event Monitoring, see the Performance Event Monitoring section in the Programmer's Reference Guide.

• --enable-localization

Enable localized error message text, if available. This option should not be used when --enable-stripped_messages is in use.

• --enable-stripped_messages

Causes all error messages to be stripped of their textual information. Instead, only error return codes are used. This option should not be used when --enable-localization is in use. Use of this build option can reduce your library footprint by up to 44KB (.so) or 50KB (.a).
If you use this configuration option, you can get an idea of what text should be issued for a given error message by using the Berkeley DB Message Reference Guide.

- **--with-cryptography**

Enter `--with-cryptography=yes|no` to build Berkeley DB libraries with or without cryptography support. By default, cryptography is enabled for Berkeley DB base libraries.

To build Berkeley DB with support for cryptography using Intel's Performance Primitive (IPP) library, enter `--with-cryptography=ipp` as an argument to configure. Additionally, set the following arguments:

- `-L/path/to/ipp/sharedlib` to `LDFLAGS`
- `-I/path/to/ipp/include` to `CPPFLAGS`
- `-lippcpem64t -lpthread` to `LIBS`

An example configuration command for IPP encryption is as follows:

```
../dist/configure --with-cryptography=ipp
CPPFLAGS="-I/opt/intel/ipp/6.1.3.055/em64t/include"
LDFLAGS="-L/opt/intel/ipp/6.1.3.055/em64t/sharedlib"
LIBS="-lippcpem64t -lpthread"
```

See the Intel Documentation for specific instructions on configuring environment variables.

Note: The `--with-cryptography=ipp` argument works only on Linux.

- **--with-mutex=MUTEX**

To force Berkeley DB to use a specific mutex implementation, configure with `--with-mutex=MUTEX`, where MUTEX is the mutex implementation you want. For example, `--with-mutex=x86/gcc-assembly` will configure Berkeley DB to use the x86 GNU gcc compiler based test-and-set assembly mutexes. This is rarely necessary and should be done only when the default configuration selects the wrong mutex implementation. A list of available mutex implementations can be found in the distribution file `dist/aclocal/mutex.m4`.

- **--with-repmgr-ssl**

Enter `--with-repmgr-ssl=yes|no|default` to build Berkeley DB libraries with or without Secure Sockets Layer (SSL) support. If `--with-repmgr-ssl` is not specified, then SSL support for replication manager is built by default depending on the availability of suitable SSL libraries and headers. If SSL libraries and headers are not present then appropriate warning will be generated. Note that to ensure that SSL support is built or blocking issues are reported as errors during configuration, you must specify `--with-repmgr-ssl=yes` to configure script as an argument. Entering `--with-repmgr-ssl=no` disables build of SSL support for the Replication Manager.
To build Berkeley DB Replication Manager with support for SSL library installed in a custom location, enter --with-repmgr-ssl=yes as an argument to configure. Additionally, set the following arguments when you run the configuration command:

- -L/path/to/ssl/sharedlib to LDFLAGS
- -I/path/to/ssl/include to CPPFLAGS
- -lssl -lcrypto -lpthread to LIBS

A sample configuration command for custom SSL location is as follows:

```bash
cd ..
dist/configure --with-repmgr-ssl=yes
CPPFLAGS="-I/home/ssluser/Downloads/openssl-1.1.0g/include"
LDFLAGS="-L/home/ssluser/Downloads/openssl-1.1.0g/"
LIBS="-lssl -lcrypto -lpthread"
```

- --with-tcl=DIR

To build the Berkeley DB Tcl API, enter --with-tcl=DIR, replacing DIR with the directory in which the Tcl tclConfig.sh file may be found. See Loading Berkeley DB with Tcl for information on sites from which you can download Tcl and which Tcl versions are compatible with Berkeley DB. To build Tcl, you must also build with shared libraries.

- --with-uniquename=NAME

To build Berkeley DB with unique symbol names (in order to avoid conflicts with other application modules or libraries), enter --with-uniquename=NAME, replacing NAME with a string that to be appended to every Berkeley DB symbol. If "NAME" is not specified, a default value of "_MAJORMINOR" is used, where MAJORMINOR is the major and minor release numbers of the Berkeley DB release. See Building with multiple versions of Berkeley DB (page 9) for more information.

## Building a small memory footprint library

There are a set of configuration options to assist you in building a small memory footprint library. These configuration options turn off specific functionality in the Berkeley DB library, reducing the code size. These configuration options include:

- --enable-smallbuild
  Equivalent to individually specifying all of the following configuration options. In addition, when compiling building with the GNU gcc compiler, this option uses the -Os compiler build flag instead of the default -O3.

- --disable-compression
  Builds Berkeley DB without support for the automatic compression of key/data pairs.

- --with-cryptography=no
  Builds Berkeley DB without support for cryptography.

- --disable-hash
  Builds Berkeley DB without support for the Hash access method.
--disable-heap
Builds Berkeley DB without support for the Heap access method.

--disable-queue
Builds Berkeley DB without support for the Queue access method.

--disable-partition
Builds Berkeley DB without support for the partitioning of databases.

--disable-replication
Builds Berkeley DB without support for the database environment replication.

--disable-statistics
Builds Berkeley DB without support for the statistics interfaces.

--disable-verify
Builds Berkeley DB without support for database verification.

--enable-stripped_messages
Strips message text from the error messages issued by Berkeley DB. This can reduce the size of the library by roughly another 22KB.

If your library has stripped messages, you can get an idea of what text should be issued for a given error message by using the Berkeley DB Message Reference Guide.

Note
--disable-cryptography and --enable-cryptography are deprecated in the Berkeley DB 11gR2 release. Use --with-cryptography=no and --with-cryptography=yes instead.

The following configuration options will increase the size of the Berkeley DB library dramatically and are only useful when debugging applications:

--enable-debug
Build Berkeley DB with symbols for debugging.

--enable-debug_rop
Build Berkeley DB with read-operation logging.

--enable-debug_wop
Build Berkeley DB with write-operation logging.

--enable-diagnostic
Build Berkeley DB with run-time debugging checks.

In addition, static libraries are usually smaller than shared libraries. By default Berkeley DB will build both shared and static libraries. To build only a static library, configure Berkeley DB with the Configuring Berkeley DB (page 30) option.

The size of the Berkeley DB library varies depending on the compiler, machine architecture, and configuration options. As an estimate, production Berkeley DB libraries built with GNU gcc version 4.X compilers have footprints in the range of 600KB to 1.4MB on 32-bit x86 architectures, and in the range of 700KB to 1.6MB on 64-bit x86 architectures.
For assistance in further reducing the size of the Berkeley DB library, or in building small memory footprint libraries on other systems, please contact Berkeley DB support.

### Changing compile or load options

You can specify compiler and/or compile and load time flags by using environment variables during Berkeley DB configuration. For example, if you want to use a specific compiler, specify the CC environment variable before running configure:

```
prompt: env CC=gcc ../dist/configure
```

Using anything other than the native compiler will almost certainly mean that you'll want to check the flags specified to the compiler and loader, too.

To specify debugging and optimization options for the C compiler, use the CFLAGS environment variable:

```
prompt: env CFLAGS=-O2 ../dist/configure
```

To specify header file search directories and other miscellaneous options for the C preprocessor and compiler, use the CPPFLAGS environment variable:

```
prompt: env CPPFLAGS=-I/usr/contrib/include ../dist/configure
```

To specify debugging and optimization options for the C++ compiler, use the CXXFLAGS environment variable:

```
prompt: env CXXFLAGS=-Woverloaded-virtual ../dist/configure
```

To specify support for the C11/C++11 standards, use the CFLAGS and CXXFLAGS environment variables:

```
prompt: env CFLAGS="-std=c11" ../dist/configure
```

or

```
prompt: env CFLAGS="-std=c11" CXXFLAGS="-std=c++11" ../dist/configure
```

To specify miscellaneous options or additional library directories for the linker, use the LDFLAGS environment variable:

```
prompt: env LDFLAGS="-N32 -L/usr/local/lib" ../dist/configure
```

If you want to specify additional libraries, set the LIBS environment variable before running configure. For example, the following would specify two additional libraries to load, "posix" and "socket":

```
prompt: env LIBS="-lposix -lsocket" ../dist/configure
```

Make sure that you prepend -L to any library directory names and that you prepend -l to any include file directory names! Also, if the arguments you specify contain blank or tab characters, be sure to quote them as shown previously; that is with single or double quotes around the values you are specifying for LIBS.

The env command, which is available on most systems, simply sets one or more environment variables before running a command. If the env command is not available to you, you can set
the environment variables in your shell before running configure. For example, in sh or ksh, you could do the following:

```
prompt: LIBS="-lposix -lsocket" ../dist/configure
```

In csh or tcsh, you could do the following:

```
prompt: setenv LIBS "-lposix -lsocket"
prompt: ../dist/configure
```

See your command shell's manual page for further information.

## Cross-Compiling on Unix

The purpose of cross-compiling is to build a Berkeley DB library on one platform that is to be used on a different platform. This section demonstrates how to build a library compatible with the ARM platform from a 32-bit x86 Linux platform. You will need to adjust the build tools and options to match your particular platforms.

1. Download, install, and test a toolchain that can build the desired target binaries. In this example, we use the `arm-linux-gnueabi-gcc` package.

2. Decide on your configuration options, and set up your environment to use the appropriate compiler and compiler tools. It is often easiest to set this up as a small shell script. For example:

```
#!/bin/sh -f
env \n CC=/usr/bin/arm-linux-gnueabi-gcc \n STRIP=/usr/bin/arm-linux-gnueabi-strip \n ../dist/configure \n --build=i686-pc-linux-gnu \n --host=arm-linux-gnueabi \n --enable-java \n --enable-sql \n --enable-jdbc \n --enable-smallbuild \n --enable-shared \n --enable-stripped_messages \n --prefix=$HOME/ARM-linux/install \n $*
```

The `--build` flag indicates the system on which you are compiling and the `--host` flag indicates the target platform. Adjust or omit the `--enable` and `--prefix` flag settings as dictated by your own needs.

3. Unpack your Berkeley DB distribution and go to the `build_unix` directory. Alternatively, you may create a more appropriately-named directory at the same level and build in it. For example, `build_arm`.

4. In your build directory, configure by executing your script, then make and make install. Review any compiler warnings and fix if necessary, repeating the configure and
make steps until you are satisfied. The resulting libraries and executables can then be transferred to your target system.

## Installing Berkeley DB

Berkeley DB installs the following files into the following locations, with the following default values:

<table>
<thead>
<tr>
<th>Configuration Variables</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--prefix</td>
<td>/usr/local/BerkeleyDB.Major.Minor</td>
</tr>
<tr>
<td>--exec_prefix</td>
<td>$(prefix)</td>
</tr>
<tr>
<td>--bindir</td>
<td>$(exec_prefix)/bin</td>
</tr>
<tr>
<td>--includedir</td>
<td>$(prefix)/include</td>
</tr>
<tr>
<td>--libdir</td>
<td>$(exec_prefix)/lib</td>
</tr>
<tr>
<td>docdir</td>
<td>$(prefix)/docs</td>
</tr>
</tbody>
</table>

### Files

<table>
<thead>
<tr>
<th>Files</th>
<th>Default location</th>
</tr>
</thead>
<tbody>
<tr>
<td>include files</td>
<td>$(includedir)</td>
</tr>
<tr>
<td>libraries</td>
<td>$(libdir)</td>
</tr>
<tr>
<td>utilities</td>
<td>$(bindir)</td>
</tr>
<tr>
<td>documentation</td>
<td>$(docdir)</td>
</tr>
</tbody>
</table>

With one exception, this follows the GNU Autoconf and GNU Coding Standards installation guidelines; please see that documentation for more information and rationale.

The single exception is the Berkeley DB documentation. The Berkeley DB documentation is provided in HTML format, not in UNIX-style man or GNU info format. For this reason, Berkeley DB configuration does not support --infodir or --mandir. To change the default installation location for the Berkeley DB documentation, modify the Makefile variable, docdir.

When installing Berkeley DB on filesystems shared by machines of different architectures, please note that although Berkeley DB include files are installed based on the value of $(prefix), rather than $(exec_prefix), the Berkeley DB include files are not always architecture independent.

To move the entire installation tree to somewhere besides /usr/local, change the value of prefix.

To move the binaries and libraries to a different location, change the value of exec_prefix. The values of includedir and libdir may be similarly changed.

Any of these values except for docdir may be set as part of the configuration:

```
prompt: ../dist/configure --bindir=/usr/local/bin
```

Any of these values, including docdir, may be changed when doing the install itself:

```
prompt: make prefix=/usr/contrib/bdb install
```
The Berkeley DB installation process will attempt to create any directories that do not already exist on the system.

**Dynamic shared libraries**

**Warning**: the following information is intended to be generic and is likely to be correct for most UNIX systems. Unfortunately, dynamic shared libraries are not standard between UNIX systems, so there may be information here that is not correct for your system. If you have problems, consult your compiler and linker manual pages, or your system administrator.

The Berkeley DB dynamic shared libraries are created with the name libdb-major.minor.so, where **major** is the major version number and **minor** is the minor version number. Other shared libraries are created if Java and Tcl support are enabled: specifically, libdb_java-major.minor.so and libdb_tcl-major.minor.so.

On most UNIX systems, when any shared library is created, the linker stamps it with a "SONAME". In the case of Berkeley DB, the SONAME is libdb-major.minor.so. It is important to realize that applications linked against a shared library remember the SONAMEs of the libraries they use and not the underlying names in the filesystem.

When the Berkeley DB shared library is installed, links are created in the install lib directory so that libdb-major.minor.so, libdb-major.so, and libdb.so all refer to the same library. This library will have an SONAME of libdb-major.minor.so.

Any previous versions of the Berkeley DB libraries that are present in the install directory (such as libdb-2.7.so or libdb-2.so) are left unchanged. (Removing or moving old shared libraries is one drastic way to identify applications that have been linked against those vintage releases.)

Once you have installed the Berkeley DB libraries, unless they are installed in a directory where the linker normally looks for shared libraries, you will need to specify the installation directory as part of compiling and linking against Berkeley DB. Consult your system manuals or system administrator for ways to specify a shared library directory when compiling and linking applications with the Berkeley DB libraries. Many systems support environment variables (for example, LD_LIBRARY_PATH or LD_RUN_PATH), or system configuration files (for example, /etc/ld.so.conf) for this purpose.

**Warning**: some UNIX installations may have an already existing /usr/lib/libdb.so, and this library may be an incompatible version of Berkeley DB.

We recommend that applications link against libdb.so (for example, using -ldb). Even though the linker uses the file named libdb.so, the executable file for the application remembers the library's SONAME (libdb-major.minor.so). This has the effect of marking the applications with the versions they need at link time. Because applications locate their needed SONAMEs when they are executed, all previously linked applications will continue to run using the library they were linked with, even when a new version of Berkeley DB is installed and the file libdb.so is replaced with a new version.

Applications that know they are using features specific to a particular Berkeley DB release can be linked to that release. For example, an application wanting to link to Berkeley DB major
release "3" can link using -ldb-3, and applications that know about a particular minor release number can specify both major and minor release numbers; for example, -ldb-3.5.

If you want to link with Berkeley DB before performing library installation, the "make" command will have created a shared library object in the .libs subdirectory of the build directory, such as build_unix/.libs/libdb-major.minor.so. If you want to link a file against this library, with, for example, a major number of "3" and a minor number of "5", you should be able to do something like the following:

```
cc -L BUILD_DIRECTORY/.libs -o testprog testprog.o -ldb-3.5
env LD_LIBRARY_PATH="BUILD_DIRECTORY/.libs:$LD_LIBRARY_PATH" ./testprog
```

where BUILD_DIRECTORY is the full directory path to the directory where you built Berkeley DB.

The libtool program (which is configured in the build directory) can be used to set the shared library path and run a program. For example, the following runs the gdb debugger on the db_dump utility after setting the appropriate paths:

```
libtool gdb db_dump
```

Libtool may not know what to do with arbitrary commands (it is hardwired to recognize "gdb" and some other commands). If it complains the mode argument will usually resolve the problem:

```
libtool --mode=execute my_debugger db_dump
```

On most systems, using libtool in this way is exactly equivalent to setting the LD_LIBRARY_PATH environment variable and then executing the program. On other systems, using libtool has the virtue of knowing about any other details on systems that don't behave in this typical way.

**Running the test suite under UNIX**

The Berkeley DB test suite is built if you specify --enable-test as an argument when configuring Berkeley DB. The test suite also requires that you configure and build the Tcl interface to the library.

Before running the tests for the first time, you may need to edit the include.tcl file in your build directory. The Berkeley DB configuration assumes that you intend to use the version of the tclsh utility included in the Tcl installation with which Berkeley DB was configured to run the test suite, and further assumes that the test suite will be run with the libraries prebuilt in the Berkeley DB build directory. If either of these assumptions are incorrect, you will need to edit the include.tcl file and change the following line to correctly specify the full path to the version of tclsh with which you are going to run the test suite:

```
set tclsh_path ...
```

You may also need to change the following line to correctly specify the path from the directory where you are running the test suite to the location of the Berkeley DB Tcl library you built:

```
set test_path ...
```
It may not be necessary that this be a full path if you have configured your system's shared library mechanisms to search the directory where you built or installed the Tcl library.

All Berkeley DB tests are run from within tclsh. After starting tclsh, you must source the file test.tcl in the test directory. For example, if you built in the build_unix directory of the distribution, this would be done using the following command:

```
% source ../test/tcl/test.tcl
```

If no errors occur, you should get a “%” prompt.

You are now ready to run tests in the test suite; see Running the test suite for more information.

## Architecture independent FAQ

1. I have gcc installed, but configure fails to find it.

   Berkeley DB defaults to using the native C compiler if none is specified. That is usually "cc", but some platforms require a different compiler to build multithreaded code. To configure Berkeley DB to build with gcc, run configure as follows:

   ```
   env CC=gcc ../dist/configure ...
   ```

2. When compiling with gcc, I get unreferenced symbols; for example the following:

   ```
   symbol __muldi3: referenced symbol not found
   symbol __cmpdi2: referenced symbol not found
   ```

   Berkeley DB often uses 64-bit integral types on systems supporting large files, and gcc performs operations on those types by calling library functions. These unreferenced symbol errors are usually caused by linking an application by calling "ld" rather than by calling "gcc"; gcc will link in libgcc.a and will resolve the symbols. If that does not help, another possible workaround is to reconfigure Berkeley DB using the `--disable-largefile` configuration option and then rebuild.

3. My C++ program traps during a failure in a DB call on my gcc-based system.

   We believe there are some severe bugs in the implementation of exceptions for some gcc compilers. Exceptions require some interaction between compiler, assembler, and runtime libraries. We're not sure exactly what is at fault, but one failing combination is gcc 2.7.2.3 running on SuSE Linux 6.0. The problem on this system can be seen with a rather simple test case of an exception thrown from a shared library and caught in the main program.

   A variation of this problem seems to occur on AIX, although we believe it does not necessarily involve shared libraries on that platform.

   If you see a trap that occurs when an exception might be thrown by the Berkeley DB runtime, we suggest that you use static libraries instead of shared libraries. See the documentation for configuration. If this doesn't work and you have a choice of compilers, try using a more recent gcc- or a non-gcc based compiler to build Berkeley DB.
Finally, you can disable the use of exceptions in the C++ runtime for Berkeley DB by using the `DB_CXX_NO_EXCEPTIONS` flag with the DbEnv or Db constructors. When this flag is on, all C++ methods fail by returning an error code rather than throwing an exception.

4. I get unexpected results and database corruption when running threaded programs.

I get error messages that mutex (for example, pthread_mutex_XXX or mutex_XXX) functions are undefined when linking applications with Berkeley DB.

On some architectures, the Berkeley DB library uses the ISO POSIX standard pthreads and UNIX International (UI) threads interfaces for underlying mutex support; for example, Solaris and HP-UX. You can specify compilers or compiler flags, or link with the appropriate thread library when loading your application to resolve the undefined references:

```
cc ... -lpthread ...
c... -lthread ...
xlc_r ...
c... -mt ...
```

See the appropriate architecture-specific Reference Guide pages for more information.

On systems where more than one type of mutex is available, it may be necessary for applications to use the same threads package from which Berkeley DB draws its mutexes. For example, if Berkeley DB was built to use the POSIX pthreads mutex calls for mutex support, the application may need to be written to use the POSIX pthreads interfaces for its threading model. This is only conjecture at this time, and although we know of no systems that actually have this requirement, it’s not unlikely that some exist.

In a few cases, Berkeley DB can be configured to use specific underlying mutex interfaces. You can use the `--enable-posixmutexes` and `--enable-uimutexes` configuration options to specify the POSIX and Unix International (UI) threads packages. This should not, however, be necessary in most cases.

In some cases, it is vitally important to make sure that you load the correct library. For example, on Solaris systems, there are POSIX pthread interfaces in the C library, so applications can link Berkeley DB using only C library and not see any undefined symbols. However, the C library POSIX pthread mutex support is insufficient for Berkeley DB, and Berkeley DB cannot detect that fact. Similar errors can arise when applications (for example, tclsh) use dlopen to dynamically load Berkeley DB as a library.

If you are seeing problems in this area after you confirm that you’re linking with the correct libraries, there are two other things you can try. First, if your platform supports interlibrary dependencies, we recommend that you change the Berkeley DB Makefile to specify the appropriate threads library when creating the Berkeley DB shared library, as an interlibrary dependency. Second, if your application is using dlopen to dynamically load Berkeley DB, specify the appropriate thread library on the link line when you load the application itself.

5. I get core dumps when running programs that fork children.
Berkeley DB handles should not be shared across process forks, each forked child should acquire its own Berkeley DB handles.

6. I get reports of uninitialized memory reads and writes when running software analysis tools (for example, Rational Software Corp.’s Purify tool).

For performance reasons, Berkeley DB does not write the unused portions of database pages or fill in unused structure fields. To turn off these errors when running software analysis tools, build with the --enable-umrw configuration option.

7. Berkeley DB programs or the test suite fail unexpectedly.

The Berkeley DB architecture does not support placing the shared memory regions on remote filesystems -- for example, the Network File System (NFS) or the Andrew File System (AFS). For this reason, the shared memory regions (normally located in the database home directory) must reside on a local filesystem. See Shared memory region for more information.

With respect to running the test suite, always check to make sure that TESTDIR is not on a remote mounted filesystem.

8. The db_dump utility fails to build.

The db_dump185 utility is the utility that supports the conversion of Berkeley DB 1.85 and earlier databases to current database formats. If the build errors look something like the following, it means the db.h include file being loaded is not a Berkeley DB 1.85 version include file:

```
db_dump185.c: In function `main':
db_dump185.c:210: warning: assignment makes pointer from integer without a cast
db_dump185.c:212: warning: assignment makes pointer from integer without a cast
```

If the build errors look something like the following, it means that the Berkeley DB 1.85 code was not found in the standard libraries:

```
cc -o db_dump185 db_dump185.o
ld:
Unresolved:
  dbopen
```

To build the db_dump185 utility, the Berkeley DB version 1.85 code must already been built and available on the system. If the Berkeley DB 1.85 header file is not found in a standard place, or if the library is not part of the standard libraries used for loading, you will need to edit your Makefile, and change the following lines:

```
DB185INC=
DB185LIB=
```
So that the system Berkeley DB 1.85 header file and library are found; for example:

```plaintext
DB185INC=/usr/local/include
DB185LIB=-ldb185
```

### AIX

1. **I can't compile and run multithreaded applications.**

   Special compile-time flags are required when compiling threaded applications on AIX. If you are compiling a threaded application, you must compile with the `_THREAD_SAFE` flag and load with specific libraries; for example, `"-lc_r"`. Specifying the compiler name with a trailing "r" usually performs the right actions for the system.

   ```plaintext
   xlc_r ...
   cc -D_THREAD_SAFE -lc_r ...
   ```

   The Berkeley DB library will automatically build with the correct options.

2. **I can't run using the DB_SYSTEM_MEM option to DB_ENV->open().**

   AIX 4.1 allows applications to map only 10 system shared memory segments. In AIX 4.3, this has been raised to 256K segments, but only if you set the environment variable "export EXTSHM=ON".

3. **On AIX 4.3.2 (or before) I see duplicate symbol warnings when building the C++ shared library and when linking applications.**

   We are aware of some duplicate symbol warnings with this platform, but they do not appear to affect the correct operation of applications.

4. **On AIX 4.3.3 I see undefined symbols for DbEnv::set_error_stream, Db::set_error_stream or DbEnv::verify when linking C++ applications. (These undefined symbols also appear when building the Berkeley DB C++ example applications).**

   By default, Berkeley DB is built with _LARGE_FILES set to 1 to support the creation of "large" database files. However, this also affects how standard classes, like iostream, are named internally. When building your application, use a "-D_LARGE_FILES=1" compilation option, or insert 
   "#define _LARGE_FILES 1" before any #include statements.

5. **I can't create database files larger than 1GB on AIX.**

   If you're running on AIX 4.1 or earlier, try changing the source code for `os/os_open.c` to always specify the `O_LARGEFILE` flag to the `open` system call, and recompile Berkeley DB from scratch.

   Also, the documentation for the IBM Visual Age compiler states that it does not support the 64-bit filesystem APIs necessary for creating large files; the ibmcxx product must be used instead. We have not heard whether the GNU gcc compiler supports the 64-bit APIs or not.
Finally, to create large files under AIX, the filesystem has to be configured to support large files and the system wide user hard-limit for file sizes has to be greater than 1GB.

6. I see errors about "open64" when building Berkeley DB applications.

System include files (most commonly fcntl.h) in some releases of AIX, HP-UX and Solaris redefine "open" when large-file support is enabled for applications. This causes problems when compiling applications because "open" is a method in the Berkeley DB APIs. To work around this problem:

a. Avoid including the problematical system include files in source code files which also include Berkeley DB include files and call into the Berkeley DB API.

b. Before building Berkeley DB, modify the generated include file db.h to itself include the problematical system include files.

c. Turn off Berkeley DB large-file support by specifying the --disable-largefile configuration option and rebuilding.

7. I see the error "Redeclaration of lseek64" when building Berkeley DB with the --enable-sql and --enable-test options.

In some releases of AIX, the system include files (most commonly unistd.h) redefine lseek to lseek64 when large-file support is enabled even though lseek may have already been defined when the _LARGE_FILE_API macro is on. To work around this problem, do either one of the following:

a. Disable large-file support in Berkeley DB by specifying the --disable-largefile configuration option and rebuilding.

b. Edit db.h manually after running the configure command, and remove the line that includes unistd.h.

FreeBSD

1. I can’t compile and run multithreaded applications.

Special compile-time flags are required when compiling threaded applications on FreeBSD. If you are compiling a threaded application, you must compile with the _THREAD_SAFE and -pthread flags:

```
cc -D_THREAD_SAFE -pthread ...
```

The Berkeley DB library will automatically build with the correct options.

2. I see fsync and close system call failures when accessing databases or log files on NFS-mounted filesystems.

Some FreeBSD releases are known to return ENOLCK from fsync and close calls on NFS-mounted filesystems, even though the call has succeeded. The Berkeley DB code should
be modified to ignore ENOLCK errors, or no Berkeley DB files should be placed on NFS-mounted filesystems on these systems.

**HP-UX**

1. **I can’t specify the DB_SYSTEM_MEM flag to DB_ENV->open().**

   The `shmget(2)` interfaces are not always used on HP-UX, even though they exist, because anonymous memory allocated using `shmget(2)` cannot be used to store the standard HP-UX msemaphore semaphores. For this reason, it may not be possible to specify the `DB_SYSTEM_MEM` flag on some versions of HP-UX. (We have seen this problem only on HP-UX 10.XX, so the simplest workaround may be to upgrade your HP-UX release.)

2. **I can’t specify both the DB_PRIVATE and DB_THREAD flags to DB_ENV->open().**

   It is not possible to store the standard HP-UX msemaphore semaphores in memory returned by `malloc(3)` in some versions of HP-UX. For this reason, it may not be possible to specify both the `DB_PRIVATE` and `DB_THREAD` flags on some versions of HP-UX. (We have seen this problem only on some older HP-UX platforms, so the simplest workaround may be to upgrade your HP-UX release.)

3. **I can’t compile and run multithreaded applications.**

   Special compile-time flags are required when compiling threaded applications on HP-UX. If you are compiling a threaded application, you must compile with the `_REENTRANT` flag:

   ```
   cc -D_REENTRANT ...
   ```

   The Berkeley DB library will automatically build with the correct options.

4. **An ENOMEM error is returned from DB_ENV->open() or DB_ENV->remove().**

   Due to the constraints of the PA-RISC memory architecture, HP-UX does not allow a process to map a file into its address space multiple times. For this reason, each Berkeley DB environment may be opened only once by a process on HP-UX; that is, calls to `DB_ENV->open()` will fail if the specified Berkeley DB environment has been opened and not subsequently closed.

5. **When compiling with gcc, I see the following error:**

   ```
   #error "Large Files (ILP32) not supported in strict ANSI mode."
   ```

   We believe this is an error in the HP-UX include files, but we don't really understand it. The only workaround we have found is to add `-D__STDC_EXT__` to the C preprocessor defines as part of compilation.

6. **When using the Tcl or Perl APIs (including running the test suite), I see the error** "Can't shl_load() a library containing Thread Local Storage".

   This problem happens when HP-UX has been configured to use pthread mutex locking, and an attempt is made to call Berkeley DB using the Tcl or Perl APIs. We have never found any way to fix this problem as part of the Berkeley DB build process. To work around the
problem, rebuild tclsh or Perl, and modify its build process to explicitly link it against the HP-UX pthread library (currently /usr/lib/libthread.a).

7. When running an executable that has been dynamically linked against the Berkeley DB library, I see the error "Can't find path for shared library" even though I correctly set the SHLIB_PATH environment variable.

By default, some versions of HP-UX ignore the dynamic library search path specified by the SHLIB_PATH environment variable. To work around this, specify the "+s" flag to ld when linking, or run the following command on the executable that is not working:

```
chattr +s enable -l /full/path/to/libdb-3.2.sl ...
```

8. When building for an IA64 processor, I see either bus errors or compiler warnings about converting between unaligned types (#4232). How can I resolve them?

Berkeley DB requires that data types containing members with different sizes be aligned in a consistent way. The HP-UX compiler does not provide this alignment property by default.

The compiler can be made to generate adequately aligned data by passing the +u1 option to the compiler. See the HP documentation about the +u1 flag for more information.

9. I see errors about "open64" when building Berkeley DB applications.

System include files (most commonly fcntl.h) in some releases of AIX, HP-UX and Solaris redefine "open" when large-file support is enabled for applications. This causes problems when compiling applications because "open" is a method in the Berkeley DB APIs. To work around this problem:

   a. Avoid including the problematical system include files in source code files which also include Berkeley DB include files and call into the Berkeley DB API.

   b. Before building Berkeley DB, modify the generated include file db.h to itself include the problematical system include files.

   c. Turn off Berkeley DB large-file support by specifying the --disable-largefile configuration option and rebuilding.

**Apple iOS (iPhone OS)**

Building Berkeley DB in Apple iOS (known as iPhone OS previously) is the same as building for a conventional UNIX platform. This section lists the commands for building Berkeley DB in both the iPhone simulator (a software simulator included in the iPhone SDK that you can use to test your application without using the iPhone/iPod Touch) and the iPhone device.

Prior to building BDB in an iPhone simulator/iPhone device, set the required environment variables for iOS (iPhone OS). For a simulator/iPhone version 4.2 or older, set the LDFLAGS variable as follows:

```
export LDFLAGS="-L$SDKROOT/usr/lib/"
```

Otherwise, set LDFLAGS as follows:
The steps to build BDB in an iPhone simulator are as follows:

```bash
export CFLAGS="-arch i386 -pipe -no-cpp-precomp --sysroot=$SDKROOT"
export CXXFLAGS="-arch i386 -pipe -no-cpp-precomp --sysroot=$SDKROOT"

cd $BDB_HOME/build_unix
../dist/configure --host=i386-apple-darwin
  --prefix=$SDKROOT ... 
made
```

The steps to build BDB in an iPhone device are as follows:

```bash
export CFLAGS="-arch armv6 -pipe -Os -gdwarf-2\ 
  -no-cpp-precomp -mthumb -isysroot $SDKROOT "
export CXXFLAGS="-arch armv6 -pipe -Os -gdwarf-2\ 
  -no-cpp-precomp -mthumb -isysroot $SDKROOT "

cd $BDB_HOME/build_unix
../dist/configure --host=arm-apple-darwin9\ 
  --prefix=$SDKROOT ... 
made
```

Both sets of commands create the BDB dynamic library - libdb-18.1.dylib. To build the static library, libdb-18.1.a, add the --enable-shared=no option while configuring.

### IRIX

1. **I can't compile and run multithreaded applications.**

   Special compile-time flags are required when compiling threaded applications on IRIX. If you are compiling a threaded application, you must compile with the _SGI_MP_SOURCE flag:

   ```bash
   cc -D_SGI_MP_SOURCE ... 
   ```

   The Berkeley DB library will automatically build with the correct options.

### Linux

1. **I can't compile and run multithreaded applications.**

   Special compile-time flags are required when compiling threaded applications on Linux. If you are compiling a threaded application, you must compile with the _REENTRANT flag:

   ```bash
   cc -D_REENTRANT ... 
   ```

   The Berkeley DB library will automatically build with the correct options.

2. **I see database corruption when accessing databases.**

   Some Linux filesystems do not support POSIX filesystem semantics. Specifically, ext2 and early releases of ReiserFS, and ext3 in some configurations, do not support "ordered
data mode” and may insert random data into database or log files when systems crash. Berkeley DB files should not be placed on a filesystem that does not support, or is not configured to support, POSIX semantics.

3. **What scheduler should I use?**

In some Linux kernels you can select schedulers, and the default is the "anticipatory" scheduler. We recommend not using the "anticipatory" scheduler for transaction processing workloads.

### Mac OS X

1. **When trying to link multiple Berkeley DB language interfaces (for example, Tcl, C++, Java, Python) into a single process, I get "multiple definitions" errors from dyld.**

   To fix this problem, set the environment variable MACOSX_DEPLOYMENT_TARGET to 10.3 (or your current version of OS X), and reconfigure and rebuild Berkeley DB from scratch. See the OS X ld(1) and dyld(1) man pages for information about how OS X handles symbol namespaces, as well as undefined and multiply-defined symbols.

2. **When trying to use system-backed shared memory on OS X I see failures about "too many open files".**

   The default number of shared memory segments on OS X is too low. To fix this problem, create a file named `/etc/sysctl.conf`, containing the variable assignments:

   ```
   kern.sysv.shmmax=134217728
   kern.sysv.shmmin=1
   kern.sysv.shmmni=32
   kern.sysv.shmseg=32
   kern.sysv.shmall=1024
   ```

   and then reboot the system.

   Check the version of OS X carefully because some versions require all five parameters to be set.

   The **SHMMAX** variable should be set to a value that is an exact multiple of 4096. Some versions of OS X ignore the value if it is not an exact multiple.

   Please check and verify with the OS X documentation on the proper setting of these parameters. In some cases a reboot is necessary and in others it is possible to change them dynamically using `sysctl`

   The `/etc/sysctl.conf` file exists in OS X 10.3.9 and later. If you are running a version of OS X previous to 10.3.x, edit the `/etc/rc` file and change the values in the following commands:

   ```
   sysctl -w kern.sysv.shmmax=134217728
   sysctl -w kern.sysv.shmseg=32
   ```

   and then reboot the system.
OS X system updates will overwrite `/etc/rc` so any changes to the file must be redone after system updates.

**SCO**

1. If I build with gcc, programs such as `db_dump` and `db_stat` core dump immediately when invoked.

   We suspect gcc or the runtime loader may have a bug, but we haven't tracked it down. If you want to use gcc, we suggest building static libraries.

**Solaris**

1. I can't compile and run multithreaded applications.

   Special compile-time flags and additional libraries are required when compiling threaded applications on Solaris. If you are compiling a threaded application, you must compile with the `D_REENTRANT` flag and link with the `libpthread.a` or `libthread.a` libraries:

   ```
   cc -mt ...
   cc -D_REENTRANT ... -lthread
   cc -D_REENTRANT ... -lpthread
   ```

   The Berkeley DB library will automatically build with the correct options.

2. I've installed gcc on my Solaris system, but configuration fails because the compiler doesn't work.

   On some versions of Solaris, there is a `cc` executable in the user's path, but all it does is display an error message and fail:

   ```
   % which cc
   /usr/ucb/cc
   % cc
   /usr/ucb/cc: language optional software package not installed
   ```

   Because Berkeley DB always uses the native compiler in preference to gcc, this is a fatal error. If the error message you are seeing is the following, then this may be the problem:

   ```
   checking whether the C compiler (cc -O) works... no
   configure: error: installation or configuration problem: C compiler cannot create executables.
   ```

   The simplest workaround is to set your `CC` environment variable to the system compiler and reconfigure; for example:

   ```
   env CC=gcc ../dist/configure
   ```

   If you are using the `--configure-cxx` option, you may also want to specify a C++ compiler, for example the following:

   ```
   env CC=gcc CCC=g++ ../dist/configure
   ```
3. I see the error "libc internal error: _rmutex_unlock: rmutex not held", followed by a core dump when running threaded or JAVA programs.

This is a known bug in Solaris 2.5 and it is fixed by Sun patch 103187-25.

4. I see error reports of nonexistent files, corrupted metadata pages and core dumps.

Solaris 7 contains a bug in the threading libraries (-lpthread, -lthread), which causes the wrong version of the pwrite routine to be linked into the application if the thread library is linked in after the C library. The result will be that the pwrite function is called rather than the pwrite64. To work around the problem, use an explicit link order when creating your application.

Sun Microsystems is tracking this problem with Bug Id's 4291109 and 4267207, and patch 106980-09 to Solaris 7 fixes the problem:

<table>
<thead>
<tr>
<th>Bug Id: 4291109</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplicate of: 4267207</td>
</tr>
<tr>
<td>Category: library</td>
</tr>
<tr>
<td>Subcategory: libthread</td>
</tr>
<tr>
<td>State: closed</td>
</tr>
<tr>
<td>Synopsis: pwrite64 mapped to pwrite</td>
</tr>
<tr>
<td>Description:</td>
</tr>
<tr>
<td>When libthread is linked after libc, there is a table of functions in libthread that gets &quot;wired into&quot; libc via _libc_threads_interface(). The table in libthread is wrong in both Solaris 7 and on28_35 for the TI_PWRITE64 row (see near the end).</td>
</tr>
</tbody>
</table>

5. I see corrupted databases when doing hot backups or creating a hot failover archive.

The Solaris cp utility is implemented using the mmap system call, and so writes are not blocked when it reads database pages. See Berkeley DB recoverability for more information.

6. Performance is slow and the application is doing a lot of I/O to the disk on which the database environment's files are stored.

By default, Solaris periodically flushes dirty blocks from memory-mapped files to the backing filesystem. This includes the Berkeley DB database environment's shared memory regions and can affect Berkeley DB performance. Workarounds include creating the shared regions in system shared memory (DB_SYSTEM_MEM) or application private memory (DB_PRIVATE), or configuring Solaris to not flush memory-mapped pages. For more information, see the “Solaris Tunable Parameters Reference Manual: fsflush and Related Tunables”.

7. I see errors about "open64" when building Berkeley DB applications.

System include files (most commonly fcntl.h) in some releases of AIX, HP-UX and Solaris redefine "open" when large-file support is enabled for applications. This causes problems when compiling applications because "open" is a method in the Berkeley DB APIs. To work around this problem:
a. Avoid including the problematical system include files in source code files which also include Berkeley DB include files and call into the Berkeley DB API.

b. Before building Berkeley DB, modify the generated include file db.h to itself include the problematical system include files.

c. Turn off Berkeley DB large-file support by specifying the `--disable-largefile` configuration option and rebuilding.

8. I see that Berkeley DB automatically uses optimization level `-xO2` for the Sun Workshop compiler. Can I change this to a higher level?

We have noticed some test failures when compiling with level `-xO3` and higher due to overaggressive compiler optimizations. We do not recommend changing this setting.

SunOS

1. I can’t specify the DB_SYSTEM_MEM flag to DB_ENV->open().

The `shmget(2)` interfaces are not used on SunOS releases prior to 5.0, even though they apparently exist, because the distributed include files did not allow them to be compiled. For this reason, it will not be possible to specify the DB_SYSTEM_MEM flag to those versions of SunOS.
Chapter 6. Building Berkeley DB for VxWorks

Building for VxWorks 6.x and VxWorks 7

Building With Wind River Workbench using the Makefile

In VxWorks 6.x and VxWorks 7, developers use the Wind River Workbench for software development. Makefiles are provided in the db/build_vxworks directory. Use them to build Berkeley DB or Berkeley DB small build, using the build chain provided with Wind River Workbench.

We assume that you have installed all necessary VxWorks software including the Wind River Workbench, and that you know how to use it.

Use the following steps to build Berkeley DB:

1. Setting variables in the Makefile. Open the Makefile and modify the BDB_ROOT variable to the path of your Berkeley DB source tree's root directory. You may need to set other variables when you build on different platforms, such as BUILD_SPEC, DEBUG_MODE, PROJECT_TYPE, CC_ARCH_SPEC, VXVER, build tool flags, and BUILD_SPEC specific settings. Please refer to the documentation of the Workbench for a complete list of available values of each variable. You can also find out the list of values by creating a project using the Workbench. Each variable's available values will be listed in the GUI window which assigns values to that variable.

2. Make sure "make" can be found. Basically you need to set the make utility's path to environment variables.

3. Start up the wrenv utility of the Wind River Workbench.

4. In the command console, move to the $(BDB_ROOT)/build_vxworks/ directory, rename the target makefile (Makefile.6x, Makefile.6x.small, Makefile.7, or Makefile.7.small) to "Makefile", and run "make". The make process will start and create the directory "bdbvxw". It will contain all intermediate object files as well as the final built image "bdbvxw.out".

5. After the "bdbvxw.out" image is built, you can use command tools or the Workbench IDE to download and run it.

6. Test and Verification. There is a dbdemo and test_micro, which you can run to verify whether everything works fine.

VxWorks notes

Berkeley DB currently disallows the DB_TRUNCATE flag to the DB->open() method on VxWorks because the operations this flag represents are not fully supported under VxWorks.

The DB->sync() method is implemented using an ioctl call into the file system driver with the FIOSYNC command. Most, but not all file system drivers support this call. Berkeley DB requires the use of a file system that supports FIOSYNC.
Building and Running the Demo Program

The demo program should be built in a manner very similar to building Berkeley DB. If you want different or additional BSP build specifications you should add them by following the directions indicated in Building for VxWorks 6.x and VxWorks 7 (page 56).

The demo program can be downloaded and run by calling the entry function dbdemo with the pathname of a database to use. The demo program will ask for some input keys. It creates a database and adds those keys into the database, using the reverse of the key as the data value. When complete you can either enter EOF (control-D) or quit and the demo program will display all of the key/data items in the database.

Building and Running the Utility Programs

The Berkeley DB utilities can be downloaded and run by calling the function equivalent to the utility's name. The utility functions take a string containing all the supported arguments. The program will then decompose that string into a traditional argc argv used internally. For example, to execute db_stat utility on a database within an environment you would execute the following from the windsh prompt. Obviously you would change the pathname and database name to reflect your system.

```bash
> db_stat "-h /tmp/myenvhome -d mydatabase.db"
```

Support for Replication Manager

The Berkeley DB Replication Manager component is available on Vxworks 6.x and 7 because they provide support for TCP/IP sockets and POSIX 1003.1 style networking and threads. You must build Berkley DB for Vxworks using the command line. Prior to building Berkeley DB, ensure you set appropriate values for the variables specified in Step 1 of Building for VxWorks 6.x and VxWorks 7 (page 56). To use Berkeley DB Replication Manager, netLib and ioLib must be present in the Vxworks image.

VxWorks FAQ

- **I get the error "Workspace open failed: This project workspace is an older format.", when trying to open the supplied workspace on Tornado 2.0 under Windows.**

  This error will occur if the files were extracted in a manner that adds a CR/LF to lines in the file. Make sure that you download the Berkeley DB "zip" version of the Berkeley DB distribution, and, when extracting the Berkeley DB sources, that you use an unzzipper program that will not do any conversion.

- **I sometimes see spurious output errors about temporary directories.**

  These messages are coming from the stat(2) function call in VxWorks. Unlike other systems, there may not be a well known temporary directory on the target. Therefore, we highly recommend that all applications use DB_ENV->set_tmp_dir() to specify a temporary directory for the application.

- **How can I build Berkeley DB without using Tornado?**
The simplest way to build Berkeley DB without using Tornado is to configure Berkeley DB on a UNIX system, and then use the Makefile and include files generated by that configuration as the starting point for your build. The Makefile and include files are created during configuration, in the current directory, based on your configuration decisions (for example, debugging vs. non-debugging builds), so you'll need to configure the system for the way you want Berkeley DB to be built.

Additionally, you'll need to account for the slight difference between the set of source files used in a UNIX build and the set used in a VxWorks build. You can use the following command to create a list of the Berkeley DB VxWorks files. The commands assume you are in the build_vxworks directory of the Berkeley DB distribution:

```
% cat > /tmp/files.sed
s/<BEGIN> FILE_//
s/_objects//
^D
% grep FILE_ BerkeleyDB.wpj | grep _objects | sed -f /tmp/files.sed \\
> /tmp/db.files
```

You will then have a template Makefile and include files, and a list of VxWorks-specific source files. You will need to convert this Makefile and list of files into a form that is acceptable to your specific build environment.

- **Does Berkeley DB use floating point registers?**

  Yes, there are a few places in Berkeley DB where floating point computations are performed. As a result, all applications that call `taskSpawn` should specify the `VX_FP_TASK` option.

- **Can I run the test suite under VxWorks?**

  The test suite requires the Berkeley DB Tcl library. In turn, this library requires Tcl 8.5 or greater. In order to run the test suite, you would need to port Tcl 8.5 or greater to VxWorks. The Tcl shell included in `windsh` is not adequate for two reasons. First, it is based on Tcl 8.0. Second, it does not include the necessary Tcl components for adding a Tcl extension.

- **Are all Berkeley DB features available for VxWorks?**

  All Berkeley DB features are available for VxWorks with the exception of the `DB_TRUNCATE` flag for `DB->open()`. The underlying mechanism needed for that flag is not available consistently across different file systems for VxWorks.

- **Are there any constraints using particular filesystem drivers?**

  There are constraints using the dosFs filesystems with Berkeley DB. Namely, you must configure your dosFs filesystem to support long filenames if you are using Berkeley DB logging in your application. The VxWorks’ dosFs 1.0 filesystem, by default, uses the old MS-DOS 8.3 file-naming constraints, restricting to 8 character filenames with a 3 character extension. If you have configured with VxWorks’ dosFs 2.0 you should be compatible with Windows FAT32 filesystems which supports long filenames.
• Are there any dependencies on particular filesystem drivers?

There is one dependency on specifics of filesystem drivers in the port of Berkeley DB to VxWorks. Berkeley DB synchronizes data using the FIOSYNC function to ioctl() (another option would have been to use the FIOFLUSH function instead). The FIOSYNC function was chosen because the NFS client driver, nfsDrv, only supports it and doesn't support FIOFLUSH. All local file systems, as of VxWorks 5.4, support FIOSYNC -- with the exception of rt11fsLib, which only supports FIOFLUSH. To use rt11fsLib, you will need to modify the os/os_fsync.c file to use the FIOFLUSH function; note that rt11fsLib cannot work with NFS clients.

• Are there any known filesystem problems?

During the course of our internal testing, we came across three problems with the dosFs 2.0 filesystem that warranted patches from Wind River Systems. We strongly recommend you upgrade to dosFs 2.2, SPR 79795 (x86) and SPR 79569 (PPC) which fixes all of these problems and many more. You should ask Wind River Systems for the patches to these problems if you encounter them and are unable to upgrade to dosFs 2.2.

The first problem is that files will seem to disappear. You should look at SPR 31480 in the Wind River Systems' Support pages for a more detailed description of this problem.

The second problem is a semaphore deadlock within the dosFs filesystem code. Looking at a stack trace via CrossWind, you will see two or more of your application's tasks waiting in semaphore code within dosFs. The patch for this problem is under SPR 33221 at Wind River Systems. There are several SPR numbers at Wind River Systems that refer to this particular problem.

The third problem is that all tasks will hang on a dosFs semaphore. You should look at SPR 72063 in the Wind River Systems' Support pages for a more detailed description of this problem.

• Are there any filesystems I cannot use?

Currently both the Target Server File System (TSFS) and NFS are not able to be used.

The Target Server File System (TSFS) uses the netDrv driver. This driver does not support any ioctl that allows flushing to the disk, nor does it allow renaming of files via FIORENAME. The NFS file system uses nfsDrv and that driver does not support FIORENAME and cannot be used with Berkeley DB.

• What VxWorks primitives are used for mutual exclusion in Berkeley DB?

Mutexes inside of Berkeley DB use the basic binary semaphores in VxWorks. The mutexes are created using the FIFO queue type.

• What are the implications of VxWorks’ mutex implementation using microkernel resources?

On VxWorks, the semaphore primitives implementing mutexes consume system resources. Therefore, if an application unexpectedly fails, those resources could leak. Berkeley DB solves this problem by always allocating mutexes in the persistent shared memory.
regions. Then, if an application fails, running recovery or explicitly removing the database environment by calling the DB_ENV->remove() method will allow Berkeley DB to release those previously held mutex resources. If an application specifies the DB_PRIVATE flag (choosing not to use persistent shared memory), and then fails, mutexes allocated in that private memory may leak their underlying system resources. Therefore, the DB_ENV->open() flag should be used with caution on VxWorks.
Chapter 7. Building Third Party APIs

This chapter provides information on building third party APIs. See the sections below:

- Building the Go API (page 61)
- Building Node.js (page 63)

Building the Go API

To install Go bindings for Oracle Berkeley DB, get a go-bdb package from https://orahub.oraclecorp.com/nosql-bdb-dev/go-bdb. You will need an Oracle SSO account. Download your preferred package format; then install it following the steps mentioned in the sections below.

Install Oracle Berkeley DB


- Windows
  1. Build Berkeley DB from the source code using Minimalist GNU for Windows (MinGW). The easiest way to get access to MinGW is to download and install MinGW-64 (https://sourceforge.net/projects/mingw-w64/) and MSYS (https://sourceforge.net/projects/mingw-w64/files/External%20binary%20packages%20%28Win64%20hosted%29/MSYS%20%2832-bit%29/)
  
     For more information on MSYS and MinGW, see http://www.mingw.org/wiki/MSYS

    2. Start a MSYS terminal and change into the unpacked db-6.2.32 directory.

    3. Execute the following command in the db-6.2.32/build_unix directory.

```
../dist/configure --prefix=/c/bdb/install
--enable-cxx --enable-mingw --host=x86_64-w64-mingw32
make
make install
```

  4. Set the following environment variables:

```
set CPATH=C:\bdb\install\include
set LIBRARY_PATH=C:\bdb\install\lib
set PATH=C:\bdb\install\lib;%PATH%
```

- Linux/Posix
  1. Unpack the Berkeley DB source library and change to the directory db-6.2.32/build_unix.

    2. Execute the following command:
..dist/configure --prefix=/usr/local/bdb/install --enable-cxx
make
make install

3. Set the following environment variables:
   export CPATH=/usr/local/bdb/install/include
   export LIBRARY_PATH=/usr/local/bdb/install/lib
   export LD_LIBRARY_PATH=/usr/local/bdb/install/lib:$LD_LIBRARY_PATH

Build Go Berkeley DB (go-bdb)

1. Under your Go installation, change to the directory Go/src and add the directory orahub.oraclecorp.com/nosql-bdb-dev.
2. Change to orahub.oraclecorp.com/nosql-bdb-dev.
3. Choose either of the following options:
   a. Checkout go-bdb using the command:
      
```bash
      git clone git@orahub.oraclecorp.com:nosql-bdb-dev/go-bdb
      ```
4. Build go-bdb using the command:
   
```bash
   go build orahub.oraclecorp.com/nosql-bdb-dev/go-bdb
   ```

Test Go Berkeley DB (go-bdb)

1. Change to the examples directory in Go/src/orahub.oraclecorp.com/nosql-bdb-dev/go-bdb/_examples.
2. Change to the directory HelloWorld.
3. Enter the command:
   
```bash
   go run HelloWorld.go
   ```
   If Hello World! is printed out, then the installation was successful.

Additional Information

- Go/src/orahub.oraclecorp.com/nosql-bdb-dev/go-bdb/_examples - Examples of how to use Berkeley DB are located in this directory.
- HelloWorld - Shows the basics of Berkeley DB, how to create a database and put and get data.
- Replication - Shows how to set up a replication group and replicate data from the master environment to a replica environment.
Building Node.js

To install Node.js bindings for the embedded key/value database Oracle Berkeley DB, you must follow the steps below:

1. Install Oracle Berkeley DB 6.3 or higher.
2. Set the appropriate environment variables to point to the installation location (after installing BDB).
3. Install `bdb-oraclejs`

   Once Berkeley DB is installed, install `bdb-oraclejs` by running the following command:

   ```sh
   npm install bdb-oraclejs --bdb=/usr/local/bdb/install
   ```

Additional Information

- `node_modules/bdb-oraclejs/examples` - Examples of how to use Berkeley DB are located in this directory.
- `basic/helloWorld.js` - Shows the basics of Berkeley DB, how to create a database and put and get data.
- `basic/replication.js` - Shows how to set up a replication group and replicate data from the master environment to a replica environment.
- `basic/index.js` - Shows how to create an index on a database.
- `basic/externalFiles.js` - Shows how to create an external file record, and how to stream data to and from it.
- `basic/sequence.js` - Shows how to create and use a sequence, a persistent object that returns an increasing or decreasing sequence of integers.
- `node_modules/bdb-oraclejs/docs/index.html` - Provides js documents for bdb-oraclejs.
Chapter 8. Upgrading Berkeley DB 12.1.6.2 applications to Berkeley DB 18.1

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 12.1.6.2 library (which belongs to Berkeley DB 12c Release 1) to the Berkeley DB 18.1 library (which belongs to Berkeley DB 18 Release 1). This information does not describe how to upgrade Berkeley DB 1.85 release applications.

For information on the general process of upgrading Berkeley DB installations and upgrade instructions related to historical releases, see the Berkeley DB Upgrade Guide.

License Change

Starting with the 18.1 releases, Berkeley DB products are licensed in two ways depending on the source of the package. Packages downloaded from the Oracle Technology Network (OTN) are licensed under the GNU AFFERO GENERAL PUBLIC LICENSE (AGPL), version 3. This license is published by the Free Software Foundation (FSF) and approved by the Open Source Initiative. Packages downloaded from the Oracle Software Data Cloud (OSDC) are licensed under the Oracle Master Agreement (OMA). Additionally, starting with the 18.1 releases, example code included in the packages is in all cases licensed under the Universal Permissive License (UPL).

Please review the terms of the appropriate license(s) to ensure compliance before upgrading to the 18.1 release.

Secure Sockets Layer (SSL) Support for the Replication Manager

Berkeley DB’s 18.1 release provides SSL support for the replication manager to enable secure communication among replication manager group members. This mainly prevents:

- Snooping of data being exchanged between replication nodes by encrypting it
- Spoofing in a replication group by means of certificate-based authentication

Berkeley DB uses openssl-library to provide this support. SSL support is enabled by default in the replication manager. It can be disabled by setting the replication manager flag (DB_REPMGR_CONF_DISABLE_SSL) before starting the replication manager.

Note that Berkeley DB supports openssl-library versions greater than or equal to 1.0.1 (as TLSv1_2_method() was introduced between version 1.0.0h and 1.0.1).

Note

Note that SSL support exists on POSIX/UNIX and windows platforms.

The methods below are used by the replication manager to provide secure communication.
New Function

**DB_ENV->repmgr_set_ssl_cert()**

This method configures the SSL support for Berkeley DB Replication Manager Messaging. The SSL support for Replication Manager may also be configured using the environment's DB_CONFIG file. This method is used to set the value of one of the following SSL configuration options:

- **DB_REPMGR_SSL_CA_CERT**
  Location of CA certificate or CA chain certificate for verification.

- **DB_REPMGR_SSL_CA_DIR**
  Location of directory containing all CA /Intermediate CA certificates for verification.

- **DB_REPMGR_SSL_REPNODE_CERT**
  Location of certificate presented by this node to peers for authentication.

- **DB_REPMGR_SSL_REPNODE_PRIVATE_KEY**
  Location of Private Key corresponding to the RepNode certificate.

- **DB_REPMGR_SSL_REPNODE_KEY_PASSWD**
  Password protecting the aforementioned Private Key.

- **DB_REPMGR_SSL_VERIFY_DEPTH**
  Number of levels of verification allowed for peer certificate verification.

New Flags

- **DB_REPMGR_CONF_DISABLE_SSL**
  This flag can be enabled via DB_ENV->rep_set_config() method. When enabled, this flag disables SSL for the Replication Manager.

- **DB_VERB_REPMGR_SSL_ALL**
  A new which parameter value for the DB_ENV->set_verbose() method to display detailed diagnostic information about all Replication Manager SSL processing.

- **DB_VERB_REPMGR_SSL_CONN**
  A new which parameter value for the DB_ENV->set_verbose() method to display detailed diagnostic information about Replication Manager SSL connection setup (connect and accept) and shutdown processing.

- **DB_VERB_REPMGR_SSL_IO**
A new which parameter value for the DB_ENV->set_verbose() method to display detailed diagnostic information about Replication Manager SSL connection IO (reads and writes).

**Multiversion Concurrency Control (MVCC) for Replication Clients**

Multiversion Concurrency Control (MVCC) is a Berkeley DB feature that offers repeatable reads, but with a different performance tradeoff than default Berkeley DB Transactional Data Store (TDS) behavior. MVCC makes extra copies of database pages when performing write operations. This is known as the copy-on-write protocol. These additional copies of database pages enable read operations to avoid locking while still remaining repeatable.

MVCC was originally implemented for replication masters and has been extended to replication clients. MVCC is desirable on replication clients because some applications are read-intensive on client sites and it is useful to optimize read operation performance on these sites.

**New Replication Environment Region Objects**

New object types are now added to DB_ENV->set_memory_init, allowing a user to account for replication's environment region memory needs.

**New Parameters**

- **DB_MEM_DATABASE**
  
  Initialize the number of databases in the environment.

- **DB_MEM_DATABASE_LENGTH**
  
  Initialize the maximum combined length of a database's directory and name strings.

- **DB_MEM_EXTFILE_DATABASE**
  
  Initialize the number of databases and subdatabases using external files.

- **DB_MEM_REP_SITE**
  
  Initialize the maximum number of sites in the replication group.

**New Replication Manager Flags**

Prior to the 18.1 release, Berkeley DB's replication manager managed connections using the select() call which has a limitation of 1024 open file descriptors. Application working with a very large number of databases, files or network connections and thus consuming more than 1024 file descriptors, were likely to run into this limitation.

Berkeley DB's 18.1 release offers the use of poll() and/or epoll() in preference to select(). In the new implementation, poll() is preferred over select() because it does
not limit the number of open file descriptors. If epoll() is available and the number of open file descriptors is greater than 1024, epoll() is preferred over poll() because it scales better.

No changes are necessary to use the new defaults, but new flags have been added for the user to disable poll() and epoll() and fall back to select().

**New Parameter Values**

- **DB_REPMGR_CONF_DISABLE_POLL**
  
  A new flags parameter value for the DB_ENV->rep_set_config() method to indicate that poll() should not be used.

- **DB_REPMGR_CONF_DISABLE_EPOLL**
  
  A new flags parameter value for the DB_ENV->rep_set_config() method to indicate that epoll() should be used (if supported by the target OS). Note that this flag needs to be explicitly specified for epoll() to be used by the replication subsystem.

**New Information Available to Replicated Applications**

Several small enhancements have been made to the base replication and replication manager APIs.

**New Error**

DB_ENV->rep_set_priority() returns the new error DB_REP_INELECT if attempting to change the database environment's priority while an election is in progress.

**New Replication Manager Event**

DB_EVENT_REP_AUTOTAKEOVER

The current process, which was previously a subordinate process, has taken over as the replication process.

**New Replication Manager Statistic**

ST_GROUP_STABLE_LOG_FILE

The stable log file indicates the earliest log file which is still potentially needed by at least one site in the replication group and should not yet be archived.

**New Replication Manager Site Field**

MAX_ACK_LSN

The log sequence number (LSN) of the most recent transaction acknowledged as durable by this remote client site.
New Replication Manager Site Flag

DB_REPMGR_ISELECTABLE

This is a new flags value in the DB_REPMGR_SITE structures returned by DB_ENV->repmgr_site_list(). The DB_REPMGR_ISELECTABLE value means that the site can potentially be elected master.

New Utility db_convert

Berkeley DB has added a new utility db_convert for converting databases between big-endian (BE) and little-endian (LE) formats. The utility allows you to specify the desired byte order without regard to the endianness of the architecture of the host machine.

New Direct Persistence Layer (DPL) API


New Utility Command Line Option

The utilities db_stat, db_dump, and db_upgrade now support the command line option -S, which verifies a database before performing the specified action.

SQL API Changes

SQL User Authentication

The SQL user authentication extension now requires encryption to ensure security.

New PRAGMAs

- PRAGMA bdbsql_upgrade, which upgrades the connected database.
- PRAGMA statistics, which prints out statistics about a database and its environment.
- PRAGMA statistics_files, which redirects statistics about a database and its environment to a file.

Internal Berkeley DB Messages

Internal Berkeley DB messages are now redirected to the same file where internal Berkeley DB error messages are redirected.

Updated SQL Version

The Berkeley DB SQL API has been updated to be compatible with SQLite 3.18.2. For information on SQLite release history, see: https://www.sqlite.org/changes.html
Changed Visual Studio Support

- Berkeley DB has added Visual Studio 2015 projects.
- Berkeley DB has added the ARM platform to Visual Studio 2015 projects and solutions.
- Berkeley DB has removed support for Visual Studio 2008.

Added VxWorks Support

Berkeley DB has added support for VxWorks 7.0.

Added PHP 7 Support

Berkeley DB now supports PHP bindings for PHP 7. However, Berkeley DB has removed support for PHP 5 or below. Berkeley DB also does not support the following PHP function:

```
Db4->pget
```

It will throw an error if called.

Error Message Numbers

Berkeley DB has made error message numbers consistent. Identical messages now always share the same message number.

Removed Utility db_sql_codegen

Berkeley DB no longer includes the db_sql_codegen utility.

Removed QNX Support

Berkeley DB no longer tests or qualifies releases on QNX.

Berkeley DB Library Version 18.1.40 Change Log

This is the changelog for Berkeley DB Release 18.1 (library version 18.1.40).

Changes between version 18.1.32 and version 18.1.40

1. Removed ASM library source files, ASM license from LICENSE file, and DPL test targets with ASM optimizations.
2. The Berkeley DB client-server architecture is no longer supported. If you require the Berkeley DB client-server architecture you must use Berkeley DB 18.1.32 or earlier.
3. The SQL API is no longer supported. If you require SQL support you must use Berkeley DB 18.1.32 or earlier.
4. Android is no longer supported. If you require Android support you must use Berkeley DB 18.1.32 or earlier.
5. Fixed several possible crashes when running db_verify on a corrupted database. [#27864]
6. Fixed several possible hangs when running db_verify on a corrupted database. [#27864]
7. Added a warning message when attempting to verify a queue database which has many extent files. Verification will take a long time if there are many extent files. [#27864]

Changes between version 18.1.25 and version 18.1.32

1. Fixed a few null pointer crashes in the Tcl API when a function returned a null value. [#26331]
2. Automatically copy any databases in sub-directories when using the utility db_hotbackup or the ENV->backup() function by including the -C command line argument or the flag DB_BACKUP_DEEP_COPY, respectively [#26736].
3. Fixed a "malformed disk error" caused by not properly refreshing the thread info. [#26945]
4. If a database rename or remove operation fails due to replication lockout, DB_REP_LOCKOUT will now be returned instead of EINVAL. [#27010]
5. Fixed an "invalid page format" error when performing a key_range search on an empty database. [#27011]
6. Fixed a bug in the JNI API involving the function Environment.lockVector(). [#27068]
7. Added the option -S to the db_convert and db_tuner utilities, which causes db_verify to verify the database before running the utility. [#27080]
8. Improved stability with several changes that prevent crashes when running db_verify against corrupt files. [#27088]
9. Improved the reliability of DB_ENV->open(DB_REGISTER) when the Berkeley DB application is the principal process of a Docker container which is stopped and restarted. [#27100]
10. Fixed build failure in repmgr ssl support for openssl version 1.1.0. [#27152]
11. Added support for Java 10. [#27154]
12. Added the db_sqlthreadtest project to Windows. [#27170]
13. Fixed a possible crash when running db_verify on a corrupted DB_QUEUE database. [#27179]
14. Improved the performance of the purging process performed when the number of free mutexes is low. [#27219]
15. Fixed an error in the repmgr035.tcl test. [#27392]
16. The Berkeley DB client-server architecture has been updated to use Thrift 0.11.0. [#27407]
Database or Log File On-Disk Format Changes

1. The log file format did not change in library version 18.1
2. The database file formats did not change in library version 18.1.

New Features/Improvements

1. Multiversion concurrency control and snapshot isolation are now supported on replication clients. [#21968]
2. Several new types of objects have been added to the DB_ENV->set_memory_init() method to enable an application to account for any additional environment region memory needed for replication. [#24153]
3. There are several new pieces of information available to replicated applications. The DB_EVENT_REP_AUTOTAKEOVER event indicates which subordinate process has taken over as the new replication process. The st_group_stable_log_file Replication Manager statistic indicates the earliest log file that is still needed by any site in the replication group. The DB_REPMGR_SITE structure now contains a max_ack_lsn field that indicates to a master site the most recently acknowledged LSN on each replica. The DB_REPMGR_SITE structure now also sets a new flag value DB_REPMGR_ISELECTABLE. [#25073]
4. Added a utility program "db_convert" to convert between big-endian (e.g. SPARC) and little-endian (e.g., X86) machines. This supports multi-platform applications which standardize on one endianness, even when both big-endian and little-endian machines are used. Since db_upgrade requires the database to be in the local machine's endianness, such applications can use db_convert to translate the byte order of the database into the local format. Then, the application can run db_upgrade. [#25086]
5. Increased the number of instances in which failchk can successfully clean the environment without requiring full recovery after a thread crashes while using Berkeley DB. [#25087]
6. Added DPL APIs to Berkeley DB Java client driver. [#25089]
7. Added support for the C11/C++11 standards. [#25416]
8. Added a graphical user interface "db_gui" to Berkeley DB that allows easy browsing of database data and execution of some administrative tasks such as running recovery. [#25496]
9. The replication manager has been changed to use poll() and epoll() in preference to select() to manage connections. This scales better for applications with a very large number of databases, network connections, or files. [#25795]
10. Starting with this release, Berkeley DB products are licensed in two ways depending on the source of the package. Packages downloaded from the Oracle Technology Network (OTN) are licensed under the GNU AFFERO GENERAL PUBLIC LICENSE (AGPL), version 3. Packages downloaded from the Oracle Software Data Cloud (OSDC) are licensed under the Oracle Master Agreement. Example code included in the packages is in all cases licensed under the Universal Permissive License. [#25916]
11. Added the -S command line option to `db_stat`, `db_dump`, and `db_upgrade`, which verifies the correctness of the database before executing the utilities. [#26207]

12. The version numbering scheme for Berkeley DB changed from a five-part number to a three-part number of the form `<year>.<release>.<build>`. This release is numbered 18.1.x, indicating that it is the first release of the year 2018. 18.1.x is the successor to the 6.2.x (also known as 12.2.6.2.x) releases. The third and fourth parameters of `db_full_version()` are no longer meaningful and are now deprecated; if present, they are are set to zero. [#26743]

**Database Environment Changes**

1. Improved the compatibility of `DB_FAILCHK` with `DB_REGISTER`. [#25204]

2. Improve environment ‘panic’ detection so that it is much less likely for processes to incorrectly use outdated environment handles after the environment has been recovered. [#25204]

3. Add the error code `DB_SYSTEM_MEM_MISSING` which is used when a System-V style shared memory segment is not found during `DB_ENV->open()`. [#25204]

4. Fixed a bug where setting the absolute path of a file on Windows could result in the file being created on the wrong disk. [#25284]

5. Syncing a database with external files will now also sync the meta database associated with the external files. [#25284]

6. Include a new field in the per-thread statistics output: the number of locked mutexes. [#25690]

**Concurrent Data Store Changes**

1. None.

**Access Method Changes**

1. Improved parallelism of `DB->open()` calls to open a `DB_BTREE` when the database name is NULL; that is, when opening a file that does not contain sub-databases. [#25624]

2. Fixed a bug on btree databases using unusually large keys that can occur when compaction is performed. [#25847]

**API Changes**

1. Added `DB->convert()` which converts database file byte order. [#25086]

**SQL-specific API Changes**

1. Fixed a hang which could occur when Berkeley DB errors are returned to the SQLite layer. [#23081]
2. SQLite user authentication extension now always requires encryption. To build with the authentication extension encryption must be enabled. Otherwise, it results in a compiler error. To convert a database to require authentication, the database must be encrypted. sqlite3_user_authenticate() now always returns SQLITE_AUTH if called on a database that does not require authentication. [#24598]

3. Added PRAGMA bdsql_upgrade, which upgrades the connected database. If the database is encrypted, the encryption key must be set before calling this pragma. If the database uses keystore authentication and the encryption key is not unknown, a user must log in before calling this pragma. [#24598]

4. Added PRAGMA statistics, which prints out statistics about the database and its environment. [#25550]

5. Added PRAGMA statistics_files, which redirects output from the statistics PRAGMA to a file. [#25550]

6. Internal Berkeley DB messages are now redirected to the same file where internal Berkeley DB error messages are redirected. [#25568]

7. Updated SQL API build to be compatible with SQLite 3.18.2. [#25986]

8. Fixed a race condition if multiple threads simultaneously attempt to create a previously non-existing on-disk environment. [#26708]

**Tcl-specific API Changes**

1. None.

**Java-specific API Changes**

1. Added support for Java 9. [#26847]

2. Fixed a compile warning in BytecodeEnhancer.java. [#26824]

3. Fixed a possible crash when using the Slices feature through the Java API. [#24988]

4. Added Database.convert() which converts database file byte order. [#25086]

5. Fixed a bug that could have caused duplicate records to be returned via the iterator() method of the DPL and Collections API. The iterator reads records in batches, and if a record at the end of the last batch was deleted by another thread, fetching the next batch could have read (and later returned via the iterator) duplicate records, depending on thread timing. [#25976]

**C#-specific API Changes**

1. Added Database.convert() which converts database file byte order. [#25086]

**Replication Changes**

1. Fixed a bug preventing some changes to external files performed through the DB_STREAM API from replicating to the clients. [#25432]
2. Fixed a bug that could cause a client undergoing internal initialization to fail to request all the necessary logs for a correct recovery. The major symptom of this bug was that some client database files were left with empty pages that would later cause log sequence errors or other failures. [#25624]

3. Fixed a bug where clients in the replication group might panic during synchronization with the master. [#25800]

4. Fixed a bug where client specific log records resulted in incorrect calculation of subsequent log offset. This would later result in a DB panic on master. [#26090]

### Locking Subsystem Changes

1. Eliminated a spurious crash in CDB environments that use both --enable-diagnostic and DB_ENV->set_thread_count(). [#26149]

#### Logging Subsystem Changes

1. None.

#### Memory Pool Subsystem Changes

1. When the number of free mutexes is low, removed databases and closed unnamed in-memory databases are purged from the memory pool to free mutexes. [#24974]

#### Mutex Subsystem Changes

1. Fixed a bug where an application might hang because it could not obtain a TAS lock when running Linux on Sparc architecture. [#24805]

2. Fixed a bug where mutexes could not be acquired on arm64-v8a Android systems. [#25752]

3. Fixed the panic "No space available in latch table" which could occur on systems where conventional (non-hybrid) Pthreads-based mutexes are used. [#26903].

#### Transaction Subsystem Changes

1. Added a new error message when opening an environment which is still being recovered: Recovery is still running on the newly created (<start>) environment The <start> field contains the ctime(3) date when recovery started. [#25204]

2. Improve the robustness of applications that ignore certain error return codes from DB_ENV->txn_begin() and incorrectly use the transaction handle. Before this fix, DB_ENV->txn_begin() did not always set the returned transaction pointer to NULL when returning a DB_REP_LOCKOUT, EINVAL, or DB_RUNRECOVERY error. Now this pointer is set to NULL in all cases, making it more apparent if the application later ignores the DB_ENV->txn_begin() failure. [#25607]

#### Test Suite Changes

1. Added test memp009 and mut004. [#24974]
Utility Changes

1. Removed the db_sql_codegen utility. [#25030]
2. Fixed a possible segfault in db_verify when used with external files. [#25920]
3. Fixed several out-of-bounds memory reads and divide-by-zero bugs in the utility programs. [#26270]
4. The -c option of db_hotbackup can now work with replication manager applications. The -c option cannot be used with base replication applications. [#25702]

Configuration, Documentation, Sample Apps, Portability and Build Changes

1. Removed support for Windows Mobile. [#24675]
2. Fixed a bug where built-in atomic functions are used only when --enable-cxx is specified. [#24974]
3. Added cross-compile support for detecting built-in atomic functions. [#25055]
4. Re-organized message numbers so that identical messages now have the same message number. [#25196]
5. Fixed a configure bug which prevented shared library support on FreeBSD 10. [#25251]
6. Added support for -with-mutex=BSD/OSSpinLockTry, to avoid using the undocumented spin_lock_try function, which is not permitted on iOS. [#25342]
7. Fixed a bug where ADO .Net package had debug pre-processors for release configurations. [#25549]
8. Fixed a bug where the configure script may fail to find the JNI include directories on macOS. [#25713]
9. Fixed a bug where the JAVACFLAGS environment variable was not passed to the JDBC driver. [#25752]
10. Added the ARM platform to Visual Studio 2015 projects and solutions. [#25860]
11. Added Visual Studio 2015 projects and solutions. [#25860]
13. Environments are now automatically recreated the first time a new version of Berkeley DB opens the environment, if the flock() system call is available. This is typical for versions of Linux. [#25884]
14. Added support for VxWorks 7. [#25930]
15. When opening a database handle which was created outside of an environment, if the current working directory contains a DB_CONFIG file it will no longer be processed. [#26319]
16. Removed support for QNX. [#26339]

17. Fixed a documentation mistake that stated that the flag DB_VERB_BACKUP should be passed to DB_ENV->backup() instead of DB_ENV->setVerbose(). [#26501]

18. Added support for PHP 7. [#26566]

19. Removed support for PHP 5 and below and the php function DB4->pget. [#26566]

20. Third party JDBC driver code was removed from the distribution. Instructions have been added about how to download and integrate with Berkeley DB. [#26746]

21. Third party ADO.NET driver code was removed from the distribution. Instructions have been added about how to download and integrate with Berkeley DB. [#26746]

**Example Changes**

1. Fixed some unchecked and deprecated warnings in the Java examples. [#26123]

**Miscellaneous Bug Fixes**

1. Fixed a bug where SQL authentication APIs might raise assertion failures. [#24598]

2. Fixed undefined symbol errors for configurations including --enable-dtrace and -enable-perfmon_statistics when replication is also enabled. [#25061]

3. Improved db_verify’s handling of corrupted database files. [#25239]

4. Fixed bugs where printed statistics might contain "unknown" flags. [#25461]

5. Fixed a bug where hot backup of queue extent files did not follow the original directory structure. [#25545]

6. The function flag DB_BACKUP_NO_LOGS now prevents logs from being backed up when passed to the backup function. [#25651]

7. Fixed a bug on Linux systems where temporary files would not be properly deleted. [#25997]

8. Fixed a bug where setting initial number of lockers might cause a segfault. [#26085]

**Deprecated Features**

1. The BDB SQL test suite is no longer part of the release package. [#26749]

2. 32-bit Windows binaries will no longer be released. [#26823]

**Known Bugs**

1. The STL interface does not work correctly on a Mac OS X platform.

2. The Solaris Studio 12.5 C++ compiler can fail to build Berkeley DB because of our support for built-in atomic functions. Falling back to Solaris Studio 12.4 should be a successful workaround.
3. Some Java tests ported from Berkeley DB JE, used to test the Direct Persistence Layer, fail when run in enhanced mode.

4. Asynchronous loggers cannot be used.
Chapter 9. Upgrading Berkeley DB 12.1.6.1 applications to Berkeley DB 12.1.6.2

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 12.1.6.1 library (which belongs to Berkeley DB 12c Release 1) to the Berkeley DB 12.1.6.2 library (which also belongs to Berkeley DB 12c Release 1). This information does not describe how to upgrade Berkeley DB 1.85 release applications.

For information on the general process of upgrading Berkeley DB installations and upgrade instructions related to historical releases, see the Berkeley DB Upgrade Guide.

Database Slices

Berkeley DB supports a new SMP-optimized data model that is especially valuable for highly multithreaded applications. A sliced Berkeley DB instance contains a collection of related Berkeley DB environments known as slices, each containing a disjoint subset of records. Transactional consistency is maintained within an individual slice but not across multiple slices.

Database slices are enabled with a configuration option when building Berkeley DB and a simple addition to the DB_CONFIG file. Database slices are supported on UNIX/POSIX platforms.

New Functions

- DB_ENV->get_slice_count()
- DB_ENV->get_slices()
- DB->set_slice_callback()
- DB->get_slices()
- DB->slice_lookup()

New Error

- DB_SLICE_CORRUPT

New Parameter Values

- DB_SLICED — A new flags parameter value for the DB->open() method to indicate that the database is sliced.

- DB_VERB_SLICED — A new which parameter value for the DB_ENV->set_verbose() method to request verbose output for database slices.
Renaming the BLOB file feature to External Files

The BLOB file feature has been renamed to *external files* to avoid confusion with the SQL datatype BLOB. The functions and flags associated with BLOBs have been deprecated, and new functions have been added to replace them. Existing applications that use the old functions will still work, but users are encouraged to use the new functions.

Below is a list of the deprecated functions, flags, and statistics, along with the replacements.

**Functions**

- `DB->set_ext_file_dir()` replaces `DB->get_blob_dir`
- `DB->get_ext_file_threshold()` replaces `DB->get_blob_threshold`
- `DB->set_ext_file_dir()` replaces `DB->set_blob_dir`
- `DB->set_ext_file_threshold()` replaces `DB->set_blob_threshold`
- `DB_ENV->get_ext_file_dir()` replaces `DB_ENV->get_blob_dir`
- `DB_ENV->get_ext_file_threshold()` replaces `DB_ENV->get_blob_threshold`
- `DB_ENV->set_ext_file_dir()` replaces `DB_ENV->set_blob_dir`
- `DB_ENV->set_ext_file_threshold()` replaces `DB->set_blob_threshold`

**New Parameter Values**

- `DB_LOG_EXT_FILE` replaces `DB_LOG_BLOB` as a flags parameter value for the `DB_ENV->log_set_config()` method.
- `DB_DBT_EXT_FILE` replaces `DB_DBT_BLOB` as a flags parameter value for the `DBT` method.

**New Statistics**

`DB->stat()` now includes the following fields to replace the old fields in the returned statistical structure.

- `hash_ext_files` replaces `hash_nblobs`
- `heap_ext_files` replaces `heap_nblobs`
- `bt_ext_files` replaces `bt_nblobs`

**Configurable Region Directory**

Berkeley DB applications can now configure the directory in which the environment region files are stored.
New Functions

• DB_ENV->set_region_dir()
• DB_ENV->get_region_dir()

Replication Write Forwarding

Replication can now forward simple DB->put() and DB->del() write operations from a client site to the master site. These write operations must specify a NULL transaction handle in order to be forwarded. Each forwarded write operation is performed in its own transaction on the master site.

New Parameter Values

• DB_REPMGR_CONF_FORWARD_WRITES — A new which parameter value for the DB_ENV->rep_set_config() method to indicate that write forwarding is enabled.
• DB_REP_WRITE_FORWARD_TIMEOUT — A new which parameter value for the DB_ENV->rep_set_timeout() method to indicate the amount of time a replication client waits for a response from a forwarded write operation.

Replication Manager IPv6 Support

Replication Manager now supports the IPv6 internet protocol and creates IPv6 connections whenever possible. If IPv6 is not available, Replication Manager creates IPv4 connections instead. Applications can optionally supply a socket approval callback for additional control over the connections created by Replication Manager.

New Function

• DB_ENV->repmgr_set_socket()

Global Message Prefix

Berkeley DB applications can now configure a prefix to be used in informational, non-error messages and output informational messages that use it. This prefix is also used by the informational messages that are output by Berkeley DB library functions.

In addition, the signature of the message call function supplied to DB_ENV->set_msgcall() and DB->set_msgcall() has been changed to add a parameter containing the message prefix.

New Functions

• DB_ENV->set_msgpfx()
• DB_ENV->get_rrpfx()
• DB_ENV->msg()
• DB->set_msgpfx()}
• DB->get_msgpfx()
• DB->msg()

**Message Call Function Signature Change**

• DB_ENV->set_msgcall()
• DB->set_msgcall()

**Reindex SQL Databases**

Existing SQL databases must be reindexed after upgrading to this release. No actions are required for non-SQL databases.

**Changes to SQL Behavior**

SQL sequence names are now case-insensitive except when surrounded by quotes. The default cache size for SQL databases has been increased from 2000 pages to 5000 pages.

**Updated SQLite Version**

Berkeley DB’s SQL interface includes code from SQLite. The version of SQLite used has been upgraded, so that Berkeley DB SQL is now compatible with SQLite version 3.8.10.2. Please see the release notes at [http://sqlite.org/changes.html](http://sqlite.org/changes.html) for further information.

**Support for Java 8**

Berkeley DB now supports the Java 8 development platform.

**Support for arm64**

Berkeley DB now supports the arm64 architecture.

**JDBC Build Changes**

The JDBC jar file (sqlite.jar) is now installed in the “lib” directory instead of the “jar” directory on UNIX/POSIX platforms. Also, cross compiling is enabled for the JDBC driver.

**vargs.h Not Supported**

Use of the varargs.h include file is no longer supported. The stdarg.h include file should be used instead.

**Removed Java Methods**

The EnvironmentConfig.setTxnNotDurable() and Environment.getTxnNotDurable() methods are no longer supported.
Dropped Support


Berkeley DB Library Version 12.1.6.2 Change Log

This is the changelog for Berkeley DB 12c Release 1 (library version 12.1.6.2).

Database or Log File On-Disk Format Changes

1. The log file format changed in release 12.1.6.2.
2. Existing SQL databases will have to be reindexed after upgrading to this release. No actions are required for non-SQL databases. Check the upgrade documentation for more details. [#23469]

New Features/Improvements

1. Added support for global message prefixes. [#20514]
2. Added Replication Manager write forwarding, which enables a client to perform simple put and delete operations by forwarding them to the master. [#21049]
3. Improved the way we mark a password as consumed. [#23876]
4. Added support for IPv6. [#23878]
5. Added support for Java 8. [#24220]
6. Added support for ARMv7 and above, and for ARM64. ARMv6 and below are no longer supported. [#24312]
7. Add a new SMP-optimized data model that is especially valuable for highly multithreaded applications. A sliced Berkeley DB instance contains a collection of related Berkeley DB environments, known as slices, each containing a disjoint subset of records. Transactional consistency is individually maintained within each slice, rather than across multiple slices. [#24343]
8. DB now supports the client-server architecture by providing a stand-alone server program and Java client driver APIs. The server program offers remote access to DB features. The client driver APIs provide building blocks for applications that communicate with a database server. Multiple client applications can communicate with a single server simultaneously. [#24360]
9. Enabled support of the SQLite user authentication extension. [#24365]
10. Added Visual Studio 2012 projects. [#24674]
11. Encryption and checksum can now be enabled with external files (formerly known as BLOBs). Note that only the data stored in the database will be encrypted and checksum protected. The data in the external files will not be encrypted or checksum protected. [#24849]
12. The BLOB files feature has been renamed external files to reduce confusion with the SQL BLOB datatype. The BLOB related functions have been deprecated, and replaced with new function names; for example, ext_file_threshold has replaced blob_threshold, and blob_threshold is deprecated. See the Upgrading section of the documentation for more details. [#24850]

Database Environment Changes

1. The location of the region directory can now be configured using DB_ENV->get/set_region_dir. [#22374]

2. Improved the statistic displays of thread states. [#23577]

3. Fixed a bug that causes a crash when opening a corrupted environment with encryption and DB_REGISTER. [#23643]

4. Fixed a bug that causes a crash when opening a corrupted environment with DB_FAILCHK using a BDB compiled with DIAGNOSTIC defined. [#23643]

5. Fixed a bug where closing a panicked environment raised an access violation and crashed the program. [#23679]

6. Fixed a bug where closing a panicked environment could lead to an infinite loop. [#23814]

7. Fixed a bug that could cause a crash when opening multiple environments in a multi-threaded program. [#23913]

8. Fixed a bug that could cause the database to appear in the wrong directory when performing catastrophic recovery with logs created in 5.3 or earlier. [#24169]

9. Improved the sizing of the environment region to better match the needs of thread tracking as specified by DB_ENV->set_thread_count(). [#24314]

10. Fixed a bug where DB_ENV->open would fail when DB_ENV->set_memory_max was called on a 32-bit system. [#24783]

11. Fixed a bug that allowed illegal use of transactions when adding a sequence to a new database. [#24786]

Concurrent Data Store Changes

1. None.

Access Method Changes

1. DB->open() now detects invalid pagesizes sooner, avoiding a potential floating point exception when a purported database file has a pagesize of 0. [#24217]

2. DB->open() now detects invalid bt_minkey better, avoiding a potential floating point exception. [#24398]

3. Removed a potential race condition when three or more threads attempt to split adjacent pages in a btree. [#24733]
API Changes

1. Added new parameter message prefix to message callback function. [#20514]
2. Added a new public error code, DB_META_CHKSUM_FAIL, that gets thrown when there is a checksum failure on any metadata page. [#23304]
3. Added new statistics to DB_REP_STAT that collect information on external files. [#23944]
4. Removed EnvironmentConfig.setTxnNotDurable() and Environment.getTxnNotDurable() from the Java API. This flag cannot be set on environments. [#24416]

SQL-specific API Changes

1. Fixed a known bug in SQL where ORDER BY could return results out of order if an index is used to answer the query. [#23469]
2. Fixed a bug where BDB crashed when enabling encryption in the SQL API using PRAGMA key. [#23655]
3. Sequence names are now case insensitive except when surrounded by quotation marks. [#23964]
4. Fixed a bug where BDB was not handling tables with multiple SQL BLOB columns when using the SQL API. [#24092]
5. Fixed a bug that an error message was not getting displayed correctly when sequence creation failed. [#24103]
6. Creating a table will no longer fail after opening the database connection without the SQLITE_OPEN_CREATE flag. [#24115]
7. Improved the speed of many SQL queries, especially ones based on index lookups. [#24183]
8. The default cache size for SQL databases has been increased from 2000 pages to 5000 pages. [#24183]
9. Fixed a bug that was preventing the PRAGMA cache_size from changing the size of the cache. [#24206]
10. Fixed a crash when writing to a SQL BLOB while txn_bulk is enabled. [#24351]
11. Updated SQL API build to be compatible with SQLite 3.8.10.2. [#24397]
12. A single transaction is now used to manage SQL BLOB handles for tables that contain SQL BLOBs. [#24549]
13. Enabled keystore based user authentication to make the authentication extension more secure and easier to work with. [#24598]
14. Fixed a bug that may lead to a crash when working with sequences. [#24826]
15. Fixed a crash when closing connection to a database with sequences. [#24827]
16. Fixed the error “BDB0232 Error opening external file” when using SQL API with PRAGMA large_record_opt enabled. [#24895]

**Tcl-specific API Changes**

1. None.

**Java-specific API Changes**

1. Fixed a bug that may cause Database.removeSequence() to throw an IllegalArgumentException. [#24360]

**C#-specific API Changes**

1. Fixed a bug so that setting DatabaseEnvironmentConfig.Register now works. [#24094]

**Replication Changes**

1. Fixed a bug that could leave a 2SITE_STRICT replication group in a state where the master cannot reelect itself master and the other site cannot join if both sites crash at an inopportune time during the other site's initial attempt to join the replication group. [#20604]

2. Fixed a bug that causes a site joining a replication group to hang or to only partially complete the internal steps necessary to join the replication group. [#23750]

3. Client to client support is now enabled for replication groups that support external files. [#23944]

4. Fixed a bug that could cause a master crash in the following cases: an unencrypted client attempts to join an encrypted replication group, or an encrypted client attempts to join an unencrypted replication group. [#24288]

5. Fixed a bug when using BDB SQL where adding a replication site immediately after opening the master environment could cause the add site operation to hang. Also fixed a bug that caused BDB SQL to lose its encryption settings when turning replication on or off on an existing environment. [#24327]

6. Enhanced the vote comparison routine used to select a new master. [#24433]

7. Fixed a bug that caused mixed-version replication live upgrades from releases prior to 6.0 to fail with an EEXIST error when in-memory databases are present. [#24597]

8. Fixed a bug that caused a double deallocation when there was a failure to set the SO_KEEPALIVE socket option on a Replication Manager connection. [#24830]

**Locking Subsystem Changes**

1. Fixed a crash which could occur when multiple BDB SQL processes repeatedly open a database file. [#24230]

2. Fixed a crash which could occur when maximum count of lockers is reached. [#24470]
Logging Subsystem Changes

1. Avoid incrementing the statistic `DB_LOG_STAT.st_scount` ("Total log file flushes") when `DB_LOG_NOSYNC` is in effect. [#23945]

Memory Pool Subsystem Changes

1. Fixed a bug that may cause a crash when `DB_ENV->memp_stat()` is called. [#24298]

2. Corrected error paths which could cause a spurious panic attempting to unlock a mutex that was not locked. [#24606]

3. When an attempt to flush a file from the memory pool fails, add an error message that includes the file name. It is likely that the file was removed or altered outside of the control of Berkeley DB. [#24606]

4. Fixed a bug where the memory pool statistics for the number of caches and maximum number of caches were incorrect. [#24758]

Mutex Subsystem Changes

1. Added support for built-in atomic (__atomic_*) functions when the compiler supports them. [#20943]

2. Fixed the error "BDB2074 No space available in latch table" for configurations using plain pthreads mutexes and `DB_ENV->set_thread_count()`. Susceptible configurations include the definition of `HAVE_MUTEX_PTHREADS` in `db_config.h` but do not define `HAVE_MUTEX_HYBRID`. [#23831]

3. Fixed `DB_RUNRECOVERY` errors caused by an incompatibility between pure pthreads-based mutexes and failchk broadcasting. [#23965]

4. Fixed a race condition which might allow `DB_ENV->failchk()` to hang after it had freed a locker ID. [#24194]

Transaction Subsystem Changes

1. Fixed a rare case where a TPEPROTO error would be returned incorrectly when using XA transactions. [#24017]

2. Aborting a transaction no longer panics if a deadlock error occurs while freeing up the transaction's cursors. [#24223]

Test Suite Changes

1. The test suite now requires Tcl 8.6 or greater. [#23878]

Utility Changes

1. A 'salvage' (db_dump -r) no longer hangs on a corrupt database which contains a loop in an overflow item's chain of pages. [#18550]
2. Fixed a bug that db_verify did not verify multiple files correctly. [#23692]

3. Fixed a crash with db_dump -d when --enable-diagnostic is on and the environment has been set up for thread tracking (DB_ENV->set_thread_count()) [#23818].

4. Added a `\-v [ov]` flag which can be used to verify the database file before dumping it. [#24217]

5. Fixed a bug where db_printlog improperly handled a “No such file or directory” error condition. [#24265]

6. Improved the way db_verify handle corrupted database files. [#24398]

7. Added additional page verifications into db_verify to prevent possible SEGV errors when verifying extremely corrupted database files. [#24754]

**Configuration, Documentation, Sample Apps, Portability and Build Changes**

1. Improved the Javadoc on EnvironmentConfig.addDataDir and EnvironmentConfig.setCreateDir. [#24093]

2. Improved the configure scripts for Mac OS X. [#24139]

3. The JDBC jar file (sqlite.jar) is now installed in the “lib” directory instead of “jar” on -nix systems. [#24139]

4. Worked around a clang compiler issue on Mac OS X where clang may incorrectly optimize out the check for external file size overflow and cause SIGXFSZ to be raised when calling DB_STREAM->write(). [#24264]

5. Enabled cross compiling the JDBC driver. [#24347]


7. Added a 64-bit Windows binary release. [#24576]

**Example Changes**

1. Added a Java example showing how to use the external file interface and storage format for working with very large records. [#24023]

2. Added the C example ex_servicemgmt, which shows how BDB can be used to implement an automobile after-sale management system. [#24064]

3. Added the SQL example ex_toll_booth, which shows how BDB can be used to implement an automated toll booth. [#24065]

4. Added the SQL example ex_advertising, which shows how BDB can be used to implement location based personal advertising. [#24065]

5. Added the C example ex_priority, which shows how BDB can be used to implement priority message handling. [#24065]
6. Added the C example `ex_firewall`, which shows how BDB can be used to implement a firewall. [#24065]

7. Wrote a guide to the Berkeley DB example programs and posted it to the Oracle Berkeley DB OTN site. [#24072]

8. Added the parking demo Java sample program. The demo program requires JDK 7 or above and DB's JDBC driver to build and run. To build the demo program, run `make dbdemos.jar`. [#24106]

9. Added the C example `ex_rep_gsg_wrfor`, which shows how to use BDB Replication Manager write forwarding. Also added C++ and Java versions of the test. [#24726]

10. Fixed a bug to make the `excxx_repquote_gsg` example programs more reliable. [#24845]

**Miscellaneous Bug Fixes**

1. Fixed a bug where opening or renaming a compressed in-memory database may fail with the error "DB1565 DB->set_bt_compress: method not permitted after handle's open method". [#22055]

2. Fixed a crash when joining compressed databases. [#22058]

3. Fixed a bug where a join cursor may miss the first matching pair when the database is compressed. [#22062]

4. Fixed a build error on Solaris where signal.h does not include a typedef for sig_t. [#22419]

5. Fixed a bug so the metadata database created when external files are enabled for a database will have the same page size as that database. [#23414]

6. Fixed a known bug in SQL where ORDER BY could return results out of order if an index is used to answer the query. [#23469]

7. Fixed a bug that may cause self-deadlock during database compaction. [#23725]

8. Removed the `-u1` compile flag from the HP-UX build. [#23739]

9. Fix a memory leak that can occur when multiple threads try to create a table simultaneously. [#23746]

10. Fixed a bug in the recovery of a truncated hash database. [#23772]

11. Fixed a Windows-specific file handling bug where an ERROR_ACCESS_DENIED error was generated on a CreateFile. [#23785]

12. Clean up attribute entry after calling `pthread_create`. [#23808]

13. Fixed a crash when MVCC and snapshot isolation is used and the system is under heavy load. [#23815]

14. Fixed a bug which was causing memory leaks in the DB utility programs and a segmentation fault. [#23816]
15. Fixed a bug that may cause DB_PAGE_NOTFOUND error during btree compaction. [#23853]

16. Fixed a bug that may cause DBcursor->get to return incorrect data when DB_GET_BOTH_RANGE flag is used. [#24220]

17. Fixed a bug that did not display file names correctly when a file failed to open during backup. [#24340]

18. Fixed a bug that may cause an environment panic when the number of lock table partitions is too large. [#24554]

19. Fixed a bug that MultipleRecnoDataEntry did not encode record-number-data pairs correctly with append(). [#24555]

20. Fixed a bug causing DBcursor->put to crash when DB_AFTER is set. [#24572]

21. Fixed a bug causing Database.setConfig() to incorrectly throw an exception. [#24585]

22. Fixed a bug that may cause DB->get and DBcursor->get to return DB_SECONDARY_BAD when Concurrent Data Store is used. [#24640]

23. Fixed a bug that may cause a segmentation fault when multiple database handles are opened on the same database in the same transaction. [#24857]

**Deprecated Features**

1. Removed support for <varargs.h>; <stdarg.h> is required. [#23895]

2. Removed unneeded historical code that is no longer active in Berkeley DB. [#23930]

3. The lock statistic st_nlockers_hit will be removed in a future release. [#24954]

**Known Bugs**

1. The STL interface does not work correctly on a Mac OS X platform.

2. The Solaris Studio 12.5 C++ compiler can fail to build Berkeley DB because of our support for built-in atomic functions. Falling back to Solaris Studio 12.4 should be a successful workaround.
Chapter 10. Upgrading Berkeley DB 12.1.6.0 applications to Berkeley DB 12.1.6.1

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 12.1.6.0 library (which belongs to Berkeley DB 12c Release 1) to the Berkeley DB 12.1.6.1 library (which belongs to Berkeley DB 12c Release 1). This information does not describe how to upgrade Berkeley DB 1.85 release applications.

For information on the general process of upgrading Berkeley DB installations and upgrade instructions related to historical releases, see the Berkeley DB Upgrade Guide.

Failchk Broadcast Notifications

The DB_ENV->failchk() method now has a new option to notify environment processes of an application failure as soon as possible. This option causes any thread waiting on a mutex owned by a crashed process to return DB_RUNRECOVERY instead of hanging.

This option is enabled at compile time with the new configure option --enable-failchk_broadcast. When enabled, this option provides a new timeout. It also provides new events that notify the application when a thread is about to return DB_RUNRECOVERY.

New Parameter Value

- DB_SET_MUTEX_FAILCHK_TIMEOUT — A new timeout parameter value for the DB_ENV->set_timeout() method to indicate how long a thread will wait for a mutex lock before checking whether DB_ENV->failchk() has marked the mutex as failed.

New Events

- DB_EVENT_FAILCHK_PANIC
- DB_EVENT_MUTEX_DIED

New Logging Option

A new logging configuration option has been added which does not flush the transaction log from the operating system cache to stable storage when a transaction commits or the logging system switches log files. This can further improve performance when all of the applications in an environment use less-than-durable transactions (DB_TXN_NOSYNC and DB_TXN_WRITE_NOSYNC).

New Parameter Value

- DB_LOG_NOSYNC — A new flags parameter value for the DB_ENV->log_set_config() method.
Replication Manager Preferred Master Mode

Replication Manager now supports preferred master mode for two-site replication groups. In preferred master mode, one site is the preferred master site, which operates as the master as much of the time as its availability permits. The other site is the client site, which takes over temporarily as master when the preferred master site is unavailable.

New Parameter Values

- DB_REPMGR_CONF_PREFMAS_MASTER — A new which parameter value for the DB_ENV->rep_set_config() method to indicate that this site is the preferred master site.

- DB_REPMGR_CONF_PREFMAS_CLIENT — A new which parameter value for the DB_ENV->rep_set_config() method to indicate that this site is the client site in a preferred master replication group.

Replication Manager Incoming Queue Size Limit

Replication Manager now limits the amount of dynamic memory that its incoming message queue can consume and provides new functions to set and get this limit. There is also a new event that is fired when the incoming message queue first becomes full.

New Functions

- DB_ENV->repmgr_set_incoming_queue_max()

- DB_ENV->repmgr_get_incoming_queue_max()

New Event

- DB_EVENT_REP_INQUEUE_FULL

New Database Format for BLOBs

The database record format for databases containing BLOBs was changed in 12.1.6.0.30. Any database from version 12.1.6.0.21 or earlier that contains BLOBs must be upgraded before it can be opened by this or any later version of Berkeley DB. Databases from version 12.1.6.0.30 or later that contain BLOBs and databases without BLOBs do not require upgrading.

Updated SQLite Version

Berkeley DB's SQL interface includes code from SQLite. The version of SQLite used has been upgraded, so that Berkeley DB SQL is now compatible with SQLite version 3.8.3.1. Please see the release notes at http://sqlite.org/changes.html for further information.

Upgrading SQL databases to releases after 6.1.19

When upgrading from Berkeley DB releases between 5.0 and 6.1.19 to a release after 6.1.19, you must re-index any SQL databases. Non-SQL databases do not require any action.
Upgrading SQL databases on POSIX/UNIX Systems

The shell script `upgrade61.sh` has been provided to simplify upgrading. Before executing the script, back up all the databases and shut down any applications that access the databases. Then execute the script, passing the path to one or more databases in the command line, as follows:

```
sh db-x.x.xx/lang/sql/upgrade61.sh <db1.dbsql> <db2.dbsql> ...
```

The upgrade process can be done by hand by executing the following commands for each database:

1. Back up the database and shut down any applications accessing it.
2. Run recovery on the database journal directory as follows:
   
   ```
   db_recover -f -h <database-journal>
   ```
3. Enter the dbsql shell and execute a SQL REINDEX command as follows:
   
   ```
   dbsql <database.dbsql>
   dbsql> REINDEX
   dbsql> .quit
   ```

That completes the upgrade process.

Upgrading SQL databases on Windows Systems

The batch script "upgrade61.bat" has been provided to simplify upgrading. Before executing the script, back up all the databases and shut down any applications that access the databases. Then execute the script, passing the path to one or more databases in the command line, as follows:

```
db-x.x.xx\lang\sql\upgrade61.bat <db1.dbsql> <db2.dbsql> ...
```

The upgrade process can be done by hand by executing the following commands for each database:

1. Back up the database and shut down any applications accessing it.
2. Run recovery on the database journal directory as follows:
   
   ```
   db_recover.exe -f -h <database-journal>
   ```
3. Enter the dbsql shell and execute a SQL REINDEX command as follows:
   
   ```
   dbsql.exe <database.dbsql>
   dbsql> REINDEX
   dbsql> .quit
   ```

That completes the upgrade process.

Removed C# Class

The `LockDetectMode` C# class has been removed as redundant. The same functionality is available in other classes.
Dropped Support

Berkeley DB no longer tests or certifies Windows XP or Windows 2003.

Berkeley DB Library Version 12.1.6.1 Change Log

This is the changelog for Berkeley DB 12c Release 1 (library version 12.1.6.1).

Database or Log File On-Disk Format Changes

1. The log file format changed in 12.1.6.1.

New Features/Improvements

1. Demonstrated in a sample app how to use thread-specific structures to associate a DB_EVENT_REP_PERM_FAILED event with the transaction that caused it. [#17686]

2. Replication Manager Preferred Master is a new option for 2-site replication groups. One site is the preferred master, which assumes the master role as much of the time as its availability permits. A preferred master replication group guarantees that transactions committed on the preferred master site will never be rolled back while also providing client failover. [#20230]

3. A new flag DB_FORCESYNCENV is provided for DB_ENV->close(), which forces flushing the memory mapped regions before unmapping them. [#21176]

4. A new API is provided to limit the Replication Manager incoming queue based on the amount of heap memory it uses. [#21481]

5. BLOB support is now available for environments configured for replication. [#22359]

6. A new version of DB_ENV->failchk() helps supports multi-process environments which use DB_REGISTER rather than a monitoring process. It is turned on with --enable-failchk_broadcast. [#22419]

7. Added support for Visual Studio 2012. [#22477]

8. CPU time needed for database backups is now reduced. [#22967]

9. Added BDB utilities for the Android platform. [#23079]

10. Added --enable-error_history support. This diagnostic tool prints a stack trace when a panic occurs. [#23119]

11. The new log configuration option DB_LOG_NOSYNC has been added to avoid flushing the transaction log in order to stabilize storage. This can be useful when durability across system crashes is unnecessary and all the transactions in the environment specify either the DB_TXN_NOSYNC or DB_TXN_WRITE_NOSYNC flag. This configuration setting may be made by DB_ENV->log_set_config() or a DB_CONFIG line "db_log_nosync [on|off]". [#23191]
**Database Environment Changes**

1. Fixed a problem with DB_LOCKDOWN which would cause an mlock error ENOMEM on some operating systems. [#{21379}]

2. DB_ENV->close() now detaches from the environment's mmap and shared memory regions even when a panic error (DB_RUNRECOVERY) has been returned. DB_ENV->open(DB_REGISTER) detects whether a panic has occurred in the environment. If so it returns DB_RUNRECOVERY. Prior to this, the panic state was not detected, which might have caused a hang. [#{22805}]

3. Ensure that the size of memory mapped backing files are a multiple of the system pagesize. Previously the files were aligned up to 8KB boundaries, which was not adequate for systems with a 64KB pagesize. [#{23290}]

**Concurrent Data Store Changes**

1. None

**Access Method Changes**

1. Fixed a bug that left behind obsolete queue extents. [#{21988}]

2. Fixed a bug that occurred if the range keys did not fit within one database page during database partition opening. [#{22280}]

3. Corrected a buffer mutex deadlock involving off-page duplicate sets and less than full transactional isolation levels. [#{22532}]

4. Fixed a byte swapping bug that could corrupt the header of heap database pages. [#{23318}]

**API Changes**

1. Make flag checking in sequence stat print API consistent with the other stat print APIs. [#{22513}]

**SQL-specific API Changes**

1. Fixed a bug that was preventing old log files from being removed while using the SQL API. [#{22521}]

2. Updated SQL API build to be compatible with SQLite 3.8.3.1. [#{23178}]

3. Updated ADO.NET API to be compatible with System.Data.SQLite 1.0.91.0. [#{23446}]

**Tcl-specific API Changes**

1. Exposed flag DB_SET_MUTEX_FAILCHK_TIMEOUT in Tcl API. [#{23163}]

2. Added msgfile and msgfile_close in the Tcl API. [#{31881}].
Java-specific API Changes

1. Exposed `DB->set_msgfile()`, `DB_ENV->set_msgfile()` and all `xxx_stat_print()` in Java API. [#22691]
2. Exposed flags `DB_REPMGR_CONF_PREFMAS_MASTER` and `DB_REPMGR_CONF_PREFMAS_CLIENT` in Java API. [#23163]
3. Exposed new flag `DB_LOG_NOSYNC` in Java API. [#23191]
4. Updated DPL changes from BDB JE. [#23251]
5. Exposed new flag `DB_EVENT_REP_INQUEUE_FULL` in Java API. [#23280]
6. Fixed a bug that partition directories could not be set correctly via Java API. [#23410]

C#-specific API Changes

1. The class `LockDetectMode` has been removed. It was duplicating functionality provided in other classes and not being used. [#21248]
2. Exposed `DB_STAT_ALLOC` in `PrintLockingSystemStats()`, `PrintLoggingSystemStats()`, `PrintMPoolSystemStats()`, `PrintMutexSystemStats()`, `PrintStats()` and `PrintTransactionSystemStats()`. [#22513]
3. Exposed `DB->set_msgcall()`, `DB->set_msgfile()`, `DB_ENV->set_msgcall()` and `DB_ENV->set_msgfile()` to C# API. [#22799]
4. Exposed flags `DB_REPMGR_CONF_PREFMAS_MASTER` and `DB_REPMGR_CONF_PREFMAS_CLIENT` in C# API. [#23163]
5. Exposed new flag `DB_LOG_NOSYNC` in C# API. [#23191]
6. Exposed new flag `DB_EVENT_REP_INQUEUE_FULL` in C# API. [#23280]
7. Changed definition of all structs to make SWIG 2.0.12 can generate code successfully. [#23295]

C++-specific API Changes

1. Fixed a bug that `db_stl` library could not build using gcc-4.7 or later versions. [#22615]

Replication Changes

1. Fixed a bug where, after some rare multiple-site failure scenarios including transaction rollbacks, Replication Manager could have incorrectly determined that new transactions fulfilled the acknowledgement policy. [#16624]
2. Acknowledgements are counted more precisely when an unelectable client is first joining the replication group and also when a master has been set to be unelectable. [#20825]
3. If there is already a replication process running, do not require a flag value when calling `DB_ENV->repmgr_start()` to start a subordinate process. [#20980]

4. Fixed a bug that a site could not rejoin the replication group on Windows after being removed from the group. [#22460]

5. When a removed site rejoins a two-site replication group with `DB_REPMGR_CONF_2SITE_STRICT` turned off, defer its election so that the rejoining site does not prematurely elect itself master with only its own vote when the other site is already master. [#22683]

6. Fixed a bug where a Replication Manager subordinate process could fail its automatic takeover as the replication process because it is unnecessarily waiting for acknowledgements from other sites before the connections to these sites are properly established. [#22998]

7. Fixed a bug where a client synchronization could get stuck because certain error paths were failing to release an internal lockout. [#23169]

8. Fixed a bug where replication group-aware log archiving might have unnecessarily kept log files after certain connection sequences between sites. [#23420]

**Locking Subsystem Changes**

1. None

**Logging Subsystem Changes**

1. None

**Memory Pool Subsystem Changes**

1. None

**Mutex Subsystem Changes**

1. Removed an environment handle mutex which was no longer needed. [#23099]

2. Removed the last source files of the defunct fcntl mutex implementation. [#23119]

**Transaction Subsystem Changes**

1. None

**Test Suite Changes**

1. None

**Utility Changes**

1. None
Configuration, Documentation, Sample Apps, Portability and Build Changes

1. Fixed a bug that ex_rep_base could not quit gracefully after receiving 'exit' command. [#16871]

2. Extended the ex_rep_mgr Replication Manager sample application to demonstrate the use of a thread-specific structure to associate a DB_EVENT_REP_PERM_FAILED event with its originating transaction. [#17686]

3. Parameter "-module" is removed from task libsqilte_jni in JDBC makefile. So libsqilte_jni library could be built in `.dylib' format on Mac OS X. [#22937]

4. Additional information about the cause of an error may now be displayed by configuring with --enable-error_history. [#23119]

5. Add one public error code 'DB_META_CHKSUM_FAIL'. [#23304]

6. Added build support on latest MinGW. [#23332]

7. A configuration warning ' Ignoring --with-cryptography flag value' is added for the non-crypto package. [#23333]

8. Parameter LIBSQLSO_LIBS is added to the task libsqlso_target to workaround a linking issue with the gcc 4.8.1 distributed by Ubuntu. [#23372]

9. Extended the Mac OS X section in the chapter 7 of the installation guide. [#23388]

10. Fixed PHP module to consistently use the ZEND_REGISTER_RESOURCE macro, preventing a build failure with PHP 5.5. Updated INSTALL file to note the changed location of sample output. [#23391]

11. Fixed a few typos in the JDBC/ODBC sections of the Windows/Unix installation guide. [#23404]

Example Changes

1. Added an example using BDB SQL with the JDBC driver and ADF. [#22569]

Miscellaneous Bug Fixes

1. Fixed a byte swapping bug with log file operations on blob files. [#22215]

2. Fixed test failures on QNX with small mutex regions. [#22310]

3. Database containing BLOBS are now portable between machines of opposite endianness. [#22455]

4. Added a -D option to display output for db_dump and db_printlog. This value takes priority over DB_ENV->set_data_len() if that was called to set the length. [#22485]

5. Updates to PHP API to handle functional changes in PHP 5.4. [#22549]
6. Fixed heap databases so they can work on both big and little endian machines. [#22929]

7. Restructure the blob delete code to improve performance. [#22935]

8. Properly clean up blob meta database after aborting the creation of its parent database. [#23000]

9. Fixed memory alignment issues with the DB_MPOOL stats. [#23150]

10. Fixed a byte swapping bug with heap blob log records. [#23319]

11. Fixed two race condition bugs with DB_ENV->failchk(). [#23347]

12. Fixed memory alignment issue on Solaris. [#23571]

13. Fixed a bug that using Visual Studio 2010 and JDK 1.5 could generate corrupted jar files. [#23574]

14. Fixed a bug that specifying --with-cryptography=yes did not build the jdbc library correctly. [#23606]

**Platform Certification Changes**

1. Berkeley DB no longer tests or certifies Windows XP or Windows 2003.

**Deprecated Features**

1. The DB_AUTO_COMMIT flag is present in the DB_SEQUENCE APIs for Java and C#. This flag is no longer meaningful for these APIs and will be deprecated in the next release.

**Known Bugs**

1. [#23469] The SQL ORDER BY command on the primary key of a table can in rare cases return the results out of order if an index search is used in executing the query. For example: Create a single column table. Insert two rows with identical values. A query on the row id and the column will produce results in a different order depending on whether an index is used.

2. [#23275] Enabling BLOB support and master leases can result in unexpected DB_REP_LEASE_EXPIRED errors.
Chapter 11. Upgrading Berkeley DB 11.2.5.3 applications to Berkeley DB 12.1.6.0

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 11.2.5.3 library (which belongs to Berkeley DB 11g Release 2) to the Berkeley DB 12.1.6.0 library (which belongs to Berkeley DB 12c Release 1). This information does not describe how to upgrade Berkeley DB 1.85 release applications.

For information on the general process of upgrading Berkeley DB installations and upgrade instructions related to historical releases, see the Berkeley DB Upgrade Guide.

Binary Large Object Support

Berkeley DB now has improved facilities for managing binary large objects (blobs). There is a new streaming interface to efficiently read and write blobs. There are new configuration interfaces to set an optional size threshold above which an object is created as a blob, and to manage the location of blob storage. A new SQL pragma enables SQL users to set a similar optional size threshold above which rows are stored internally in Berkeley DB as blobs.

New Functions

- DBC->db_stream()
- DB_STREAM->close()
- DB_STREAM->read()
- DB_STREAM->size()
- DB_STREAM->write()
- DB_ENV->set_blob_threshold()
- DB_ENV->get_blob_threshold()
- DB->set_blob_threshold()
- DB->get_blob_threshold()
- DB_ENV->set_blob_dir()
- DB_ENV->get_blob_dir()
New Flags Accepted by Blob Streaming Interfaces

- DB_STREAM_READ — accepted by DBC->db_stream()
- DB_STREAM_WRITE — accepted by DBC->db_stream()
- DB_STREAM_SYNC_WRITE — accepted by DBC->db_stream() and DB_STREAM->write()

Other New Flags

- DB_DBT_BLOB — Used on the data DBT accepted by DB->put() and DBC->put() to indicate that the DBT is a blob.
- DB_LOG_BLOB — Accepted by DB_ENV->log_set_config() and DB_ENV->log_get_config() to indicate full logging of blobs.

New Pragma

- PRAGMA large_record_opt [=n]

Replication Views

A replication site can now be created as a replication view by using the DB_ENV->rep_set_view() method. A replication view is a read-only copy of some or all of the replicated databases that cannot become master, participate in elections, or contribute to transaction durability.

New Function

- DB_ENV->rep_set_view()

New Flag

- DB_REPMGR_ISVIEW - A new value for the flags field of the DB_REPMGR_SITE structures returned by the DB_ENV->repmgr_site_list() method to indicate that a site is a replication view.

Enhanced BDB SQL Replication Options

Berkeley DB SQL replicated applications now support multiprocess access. Also, there are several new replication pragmas. The replication_ack_policy, replication_ack_timeout and replication_priority pragmas provide ways to configure a site's acknowledgement criteria and election priority. The other new pragmas provide additional information about the current state of the replication group or the local site.

New Pragmas

- PRAGMA replication_ack_policy [=all|all_available|none|one|quorum]
- PRAGMA replication_ack_timeout [=n]
• PRAGMA replication_priority [=n]
• PRAGMA replication_get_master
• PRAGMA replication_num_sites
• PRAGMA replication_perm_failed
• PRAGMA replication_site_status

Replication Manager Automatic Takeover

When running Replication Manager in multiple processes, if the replication process terminates successfully and there is an available subordinate process, a subordinate process now automatically takes over as the new replication process. The new DB_EVENT_REP_AUTOTAKEOVER_FAILED event informs the application when a takeover fails, leaving the application without a replication process.

New Event

• DB_EVENT_REP_AUTOTAKEOVER_FAILED

New Berkeley DB SQL API Pragma

The new Berkeley DB SQL API specific pragma bdbsql_log_buffer has been added to set the size of the Berkeley DB in-memory log buffer, in bytes. This is useful for long-running transactions or applications producing large amounts of data because a larger log buffer can increase throughput.

New Pragma

• PRAGMA bdbsql_log_buffer [=n]

UNIX/POSIX SQL Encryption Default

Encryption is no longer enabled by default when building the SQL interface, either with or without the JDBC driver, on UNIX/POSIX platforms. To enable encryption, use the configure argument --with-cryptography=yes.

Sequence Datatype Changes

The datatype for several pieces of information associated with sequences has changed from int32_t to u_int32_t:

• DB_SEQUENCE->get() delta parameter
• DB_SEQUENCE->set_cachesize() size parameter
• DB_SEQUENCE->get_cachesize() sizep parameter
Comparison Function Signature Change

A new parameter has been added to the comparison callback set by the following methods:

- `DB->set_bt_compare()`
- `DB->set_dup_compare()`
- `DB->set_h_compare()`

The new parameter is the `locp` parameter, and it is the last parameter required in the callback's signature. It is intended for internal use only. In customer code, the parameter must be set to NULL.

License Change

Starting with the 6.0 / 12c releases, all Berkeley DB products are licensed under the GNU AFFERO GENERAL PUBLIC LICENSE (AGPL), version 3. This license is published by the Free Software Foundation (FSF) (1) and approved by the Open Source Initiative (2). Please review the terms of the license to ensure compliance before upgrading to the 12c release. Previous releases of Berkeley DB software will continue to be distributed under the Sleepycat license.

Updated SQLite Version

Berkeley DB's SQL interface includes code from SQLite. The version of SQLite used has been upgraded, so that Berkeley DB SQL is now compatible with SQLite version 3.7.16.2. Please see the release notes at http://sqlite.org/changes.html for further information.

Sun Workshop Compiler Optimization Level

Berkeley DB now automatically uses optimization level "-xO2" with the Sun Workshop compiler. Test failures have been observed with higher optimization levels. This can include the default "-O", which equates to the higher "-xO3" optimization level for some Sun compilers.

Dropped Support

Berkeley DB no longer supports Tornado 2.0 or versions of VxWorks earlier than 6.9.

Berkeley DB Library Version 12.1.6.0 Change Log

This is the changelog for Berkeley DB 12c Release 1 (library version 12.1.6.0).

Database or Log File On-Disk Format Changes

1. The log file format changed in 12.1.6.0.
New Features

1. Added an alternate storage format called blobs, which improves the read and update performance of very large data items. Interface available through C, C#, C++ and Java APIs. [#19332]

2. Added several pragmas to the SQL layer that configure and give information on HA SQL applications, such as the number of sites in the replication group, or configuring election priority of a site. [#20227]

3. Added a BDDBSQL_CONCURRENT_CONNECTIONS flag to the SQL API to allow database locking to be disabled. Useful when there are only readers in the database. [#20928]

4. When running Replication Manager in multiple processes, if the replication process quits and there is an available subordinate process, the subordinate process automatically takes over as the replication process. [#20969]

5. Added replication views/partial replication feature. Feature available through C, C#, C++ and Java APIs. [#20162]

6. DPL enhancements to work with JDK7. [#21101]

7. Added encryption support for the Android JDBC driver. [#21129]

8. Applications which use snapshot isolation and long-running transactions will, in many instances, see reduced use of the cache. This leads to improved performance by reducing the need for temporary freezer files. [#21324]

9. Added mpool stat fields st_mvcc_resued, st_oddfsize_detect and st_oddfsize_resolve to C#. [#21325]

10. Added support database partitioning in C# APIs. [#21474]

11. Added a per-thread cache for the locker struct which stores lock identifiers, such as transaction ids or the locker id returned by DB_ENV->lock_id(). This can improve concurrency of programs which use DB_ENV->set_thread_count(). [#21591]

12. Port utilities to Windows Mobile. [#21636]

13. Expose 3 new repmgr stat values and a new DB_REPMGR_SITE flag value to C# and Java APIs. [#21687]

14. Added support for localizing error messages in UTF-8 characters, and added a section about how to build localized library and an ICU example in Programmer’s Reference Guide. [#21280]

Database Environment Changes

1. Redundant whitespaces are now ignored in DB_CONFIG lines pertaining to directories. [#20158]

2. Fixed bugs when attempting to open non-DB files as if they were databases. If encryption was active, an assertion failure would occur. If encryption was not active, the error
message "BDB0178 Encrypted database: no encryption flag" would incorrectly imply that a password was required. [#21408]

Concurrent Data Store Changes

1. None

Access Method Changes

1. Fixed a bug that could allocate a heap data page in a region after the region creation has been undone. [#20939]

2. Fixed a bug in heap that prevented it from working in mixed-endian configurations. [#21016]

3. Enhance the search performance for Btree access method when the keys share a common prefix. [#21216]

4. Fixed bugs with the byte swapping of heap pages. Now heap databases can be moved between little-endian and big-endian architectures. [#21387]

5. Fix a race condition where a cursor in the process of being cleaned up could be accessed by another thread of control. [#21714]

6. Duplicating the position of a cursor on a partitioned database will no longer result in a crash. [#22195]

7. Fixed an assertion error when closing a joined cursor. [#22398]

API Changes

1. Change the sequence’s cachesize and the delta parameter of DB_SEQUENCE->get to be of type u_int32_t instead of int32_t. [#21552]

2. Fixed a bug with the DB_MPOOLFILE->set_maxsize API. In previous releases the file was allowed to grow one page larger than the configured limit. [#21867]

3. Change the comparison function in DB->set_BT_compare, DB->set_dup_compare and DB->set_H_compare to accept an extra size_t *locp parameter. [#22235]

SQL-specific API Changes

1. Added the PRAGMA large_record_opt, which when enabled can improve the read and update performance of very large records. [#19332]

2. Added support for multi-process access to the database in SQL HA applications. [#21050]

3. In-memory databases can now be created without passing the flag SQLITE_OPEN_CREATE. [#21456]

4. Fix a bug in the SQL API where setting the page size to 64 K would cause an error. Also fixed an associated memory leak in SQL online backup. [#21460]
5. Add bdbsql_log_buffer pragma to allow setting of log buffer size. [#21471]

6. Upgraded SQLite to version 3.7.13. [#21641]

7. Fixed a mutex leak with multi-process access to the database in SQL HA applications. [#21714]

8. Fixed a bug that could cause jdbc build to fail on ARM-Linux using gnueabi gcc. [#22123]

9. Upgraded SQLite to version 3.7.16.2. [#22349]

10. Upgraded ADO.NET to version 1.0.85.0. [#22405]

**Tcl-specific API Changes**

1. Support blob storage APIs in Tcl. [#19332]

**Java-specific API Changes**

1. Fixed a bug where EnvironmentConfig.setCreateDir() was not correctly configuring the environment. [#21127]

2. Fixed a bug where Database.verify() was not correctly configuring the db. [#22401]

3. Fixed a false error message that could be printed when creating a transaction on a replication client in Java. [#21671]

**C#-specific API Changes**

1. Clean up many C# build warnings. [#22285]

2. Fixed SWIG typemap defintions for C# interface. [#18509]

**Replication Changes**

1. Fixed several assertion failures that could result when using failchk and replication. [#19765]

2. Provide an error when the local site defined during internal initialization is inconsistent with an earlier definition. [#20870]

3. Fixed a bug where replication clients were incorrectly flushing the log when the DB_TXN_WRITE_NOSYNC flag was specified. [#20964]

4. Fixed a bug where a valid invocation of the DB_ENV->repmgr_start() method would fail after a previous invocation that supplied an invalid nthreads parameter value. A BDB3640 error was being incorrectly generated. [#21289]

5. Fixed a bug where the calculation of the number of required master lease grants enforced that a transaction must be present on a strict majority of replication group sites. It is sufficient that a transaction be present on at least half of the replication group sites, which in some cases reduces the number of required lease grants. [#21329]
6. Fixed a recovery issue on the client where an incorrect checkpoint was being selected causing recovery to go to an incorrect point. [#21342]

7. Fixed a bug that could result in a crash due to an incorrect re-request on a site that just won an election. [#21344]

8. Fixed a bug where logs would never get archived if there are 2 sites and various combinations of failures. [#21448]

9. Fixed a bug where we were not detecting the fact that DB_TXN_SNAPSHOT was being used with an HA client. Clients are by definition read only, so snapshot transactions are not permitted. [#21601]

10. Fixed a bug where the determination of whether or not a site is peer was incorrectly determined. [#21722]

11. Fixed a bug where a Replication Manager master could stop functioning after accepting an obsolete group membership site list from another site. [#21804]

12. Reduce the multiple data streams that can sometimes occur if rerequest times are shorter than the time it takes to switch log files or perform a checkpoint at a client site. [#21843]

13. Fixed a bug that a site is removed unexpectedly from the replication group when the replication group loses the old master's confirmation of adding the site during a master change. [#21875]

14. Fixed a bug where a master site could crash when removing a site if the removed site closes its connections before the master completes its final steps of the operation. [#22050]

15. Fixed a bug where an election never happens after the loss of the master when master leases are in use. [#22120]

**Locking Subsystem Changes**

1. Fixed a bug that could cause a trap if a lock timeout occured while opening a database. [#21098]

2. Fixed a hang which could occur if there were no more space for lockers. [#21774]

3. Fixed a potential race condition when building the deadlock detection graph which, under rare conditions, might cause a invalid address fault. [#22261]

**Logging Subsystem Changes**

1. Fixed a bug that DB_ENV->log_get_config did not work correctly when it was called before DB_ENV->open. [#21359]

**Memory Pool Subsystem Changes**

1. Fixed a bug in the DB_ENV->get_cache_max API that caused it to return an incorrect value and a bug in the DB_ENV->set_cachesize API that caused incorrect behavior when changing the size of an existing cache. [#21146]
2. Fixed a race condition occurring when trying to get information about a file and the file is being extended by the OS. [#21193]

3. Corrected error messages in several memory pool configuration functions. [#21779]

4. Fixed several memory leak items identified by Purify runs. [#22283]

**Mutex Subsystem Changes**

1. Fixed an undefined symbol error (MUTEX_IS_OWNED) when configuring with --enable-diagnostic --disable-mutexsupport. [#21145]

2. Fix a bug where we were not properly aligning mutexes in DB_PRIVATE regions when the application configured an alignment. [#21491]

3. Fix a race condition between DB_ENV->failchk() and the allocation of a mutex. [#21796]

4. Limit the default mutex spin count to be at most 200, to avoid excessive spinning on systems with hundreds of hyperthreads. [#21463]

**Transaction Subsystem Changes**

1. None

**Test Suite Changes**

1. Added support for get/set_heapsize. [#21510]

**Utility Changes**

1. Fixed a bug where calling DB->set_partition with a key array required users to keep the memory array valid until after the database was opened. [#18350]

2. Fixed a bug that caused DB_ENV->backup to stop early if DB_BACKUP_FILES was not set and a non-DB file was in the data directory. [#21076]

3. Fixed a problem with db_hotbackup's -F option, which could cause a segmentation violation. Also replaced an unclear error message. [#21184]

4. Fixed three problems with db_hotbackup's backup of transaction logs. A hot backup did not use any configured log directory, but would try to open the logs in the environment home. The second fix corrected an error path, in which the memory was freed by the wrong function, possibly causing a guard byte error. The third fix fixed the issue that a wrong message would be displayed when only "-l" was specified. [#21313]

5. Fixed a spurious "No such file or directory" (ENOENT) error when running db_archive in an environment with an empty queue database. [#21541]

6. Fixed bugs in compaction of large keys in the upper levels of btrees. [#21569]

7. Most utilities no longer operate on replication clients which are being automatically initialized. The DB_REP_LOCKOUT error is now returned. [#21593]
8. Fixed a resource leak in the db->verify() function for btrees. The bug would slow down verification and possibly cause it to run out of memory. [#21917]

9. Removed a potential hang when compacting databases with many duplicates. [#21975]

10. Fixed a bug where we were not returning an error when trying to compact a heap database. [#22063]

11. Fixed a bug in the error handling of db_log_verify which would cause a segmentation violation if there were not enough space to create its temporary databases. [#22187]

**Configuration, Documentation, Sample Apps, Portability and Build Changes**

1. Fix an odbc configure issue which was causing ODBC build failure with SQL API. [#21490]

2. Disabled encryption by default in SQL API on Linux, completed build options for encryption in SQL and SQL JDBC on Linux, Windows and Android, and updated build process for the three platforms in Installation Guide. [#21872]

3. Use "-xO2" as the default optimization level for the Sun Workshop compiler. [#22057]

4. Fix an "opcode not supported" error on MIPS platforms whose assembler defaults to the MIPS1 instruction set. [#22065]

5. Provide a consistent way to turn on/off encryption for db,sql,jdbc libraries on Unix. [#22158]

6. Added the compile option CONFIG_TEST when using debug mode in the db_csharp Windows project, to be consistent with the db Windows project. [#22300]

**Example Changes**

1. Port the C bulk example to C++. [#19541]

**Miscellaneous Bug Fixes**

1. Fixed a race condition that could cause a crash if 2 processes opened the same SQL database at the same time. [#21041]

2. Fixed an assert failure in the btree comparison function which caused memory to be allocated to the wrong thread. [#21232]

3. Fixed a race condition in the SQL handle cache locking code. [#21265]

4. Fixed a bug in the SQL code that prevented the table from being created when BDBSQL_FILE_PER_TABLE is defined. [#21417]

5. Fixed a bug that some fields of the partition key DBTs returned from DB->get_partition_keys may not be initialized. [#21474]

6. Fixed a bug that prevented a sub-database from being created under the directory identified in DB->set_create_dir. [#21603]
7. Fixed a bug where dbenv->mp_mtxcount was not being used correctly. [#21768]

8. Cleaned a few variable initialization problems that were causing failures on 64-bit Windows. [#21850]

9. Improved the randomness of the 20 byte unique fileid generator on Windows. [#22470]

10. Fixed a possible assertion failure in the XA Transaction Manager feature. [#22450]

Deprecated Features

1. Tornado 2.0 is not supported, and VxWorks versions older than 6.9 are not supported. [#21877]

Known Bugs

1. Prepared transactions may not remove databases. An error EINVAL message will be returned by TXN->prepare() if the transaction removed any databases.

2. Databases containing BLOBs are not portable between machines of opposite endianness.

3. The latest version of the ADO.NET driver uses a new database connection close function that does not clean up the database completely, leaving transactions active. In our build projects that we distribute we added the preprocessor definition INTEROP_LEGACY_CLOSE so the older version will be used. Removing this can cause hangs.

4. The db_log_verify utility does not currently implement support for the "-D" option. Also, the "-d" option does not work for an environment with multiple databases. For environments with sub-databases, some unexpected warnings will be thrown. [#22136]

5. Aborting a prepared transaction that removes a database or does operations on a blob cannot undo the operations (although the call to txn->prepare throws an error if a user tries to do that). [#22226]

6. Using the DB_FAILCHK flag with just the DB_RECOVER flag can cause processes to hang. DB_RECOVER will cause the environment to be rebuilt, and when DB_FAILCHK is specified, it runs fail check as the last step in the open. It is done after the environment has been rebuilt. There isn't anything for fail check to really do in this case. The hang can occur if a separate process also uses this combination which would mean the environment is rebuilt again and it could conflict with the first process if it is doing the fail check. [#22419]

7. A rare race condition in the XA transaction manager can return the error XAER_PROTO in a one-phase commit. [#22450]
Chapter 12. Upgrading Berkeley DB 11.2.5.2 applications to Berkeley DB 11.2.5.3

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 11.2.5.2 library to the Berkeley DB 11.2.5.3 library (both of which belong to Berkeley DB 11g Release 2). This information does not describe how to upgrade Berkeley DB 1.85 release applications.

For information on the general process of upgrading Berkeley DB installations and upgrade instructions related to historical releases, see the Berkeley DB Upgrade Guide.

Changes to the build_windows Folder

Visual Studios projects in the build_windows folder have been moved into two new subfolders build_windows/VS8 and build_windows/VS10. Visual Studios 2010 projects have been moved into the folder VS10, and Visual Studios 2008 projects have been moved into the folder VS8.

Replication Connection Status in the Java API

The Java function ReplicationManagerSiteInfo.isConnected() is now deprecated. To get the replication connection status, use ReplicationManagerSiteInfo.getConnectionStatus(), which returns the new class ReplicationManagerConnectionStatus, which has the values CONNECTED, DISCONNECTED, and UNKNOWN.

New Function

- ReplicationManagerSiteInfo.getConnectionStatus()

New Class

- ReplicationManagerConnectionStatus

Deprecated Function

- ReplicationManagerSiteInfo.isConnected()

Exclusive Database Handles

Database handles can now be configured to allow exclusive access to the database. To enable exclusive access, call DB->set_lk_exclusive() before calling DB->open(). Set nowait_onoff to non-zero to have DB->open() return immediately, with the error DB_LOCK_NOTGRANTED if it cannot immediately get exclusive access to the database, and to 0 to have DB->open() block until it can gain exclusive access.
New Functions

- DB->set_lk_exclusive()
- DB->get_lk_exclusive()

Configure the Region Size of Heap Databases

The region size of heap databases is now configurable. Configuring the region size is useful in controlling the growth of a heap database. To set the region size, call DB->set_heap_regionsize() with the number of pages that the region should have, before the database is created. The function is ignored if it is called after the database is created.

New Functions

- DB->set_heap_regionsize()
- DB->get_heap_regionsize()

New Hotbackup Interface

Two new functions have been added to the API that perform hotbackups, DB_ENV->backup() and DB_ENV->dbbackup(). DB_ENV->backup() creates a hotbackup of all databases in the specified environment, and DB_ENV->dbbackup() creates a hotbackup of the specified database. The functions DB_ENV->set_backup_callbacks() and DB_ENV->set_backup_config() can be called to customize the behavior of hotbackup. Note that this interface must be used to create a hotbackup on all non-BSD or Unix based systems.

New Functions

- DB_ENV->backup()
- DB_ENV->dbbackup()
- DB_ENV->set_backup_callbacks()
- DB_ENV->set_backup_config()

Flags Accepted by DB_ENV->backup()

- DB_BACKUP_CLEAN
- DB_BACKUP_FILES
- DB_BACKUP_NO_LOGS
- DB_BACKUP_SINGLE_DIR
- DB_BACKUP_UPDATE
• DB_CREATE
• DB_EXCL
• DB_VERB_BACKUP

Flags Accepted by DB_ENV->dbbackup()

• DB_CREATE
• DB_EXCL

Enumerations Accepted by DB_ENV->set_backup_config()

• DB_BACKUP_WRITE_DIRECT
• DB_BACKUP_READ_COUNT
• DB_BACKUP_READ_SLEEP
• DB_BACKUP_SIZE

Updated JDBC Version

The JDBC library version included with Berkeley DB has been updated to version 20110827. The new version supports the Embedded Java JSR 169 standard, which is the official specification for JDBC with embedded Java platforms.

Configure Directory to Store Metadata Files

The directory in which persistent metadata files are stored can now be configured. By default persistent metadata files are stored in the environment home directory. The files that will be stored in the metadata directory are __db.rep.system, __db.rep.gen, __db.rep.egen and __db.rep.init.

To set the metadata file directory, call DB_ENV->set_metadata_dir() with the path to the directory in which to store metadata files. The metadata directory can also be set in the DB_CONFIG file using set_metadata_dir.

New Functions

• DB_ENV->set_metadata_dir()
• DB_ENV->get_metadata_dir()

Changes in the SQL API Build

Several changes have been made to the SQL API build. Encryption is now disabled by default on Windows, and the compile time flag BDBSQL_OMIT_SHARING has been changed to BDBSQL_SINGLE_PROCESS.
Encryption is now disabled by default in the Windows SQL API build. This makes it consistent with builds of the SQL API on other systems. To enable encryption in Visual Studios right click the db_sql project, and select Properties->Configuration Properties->C/C++->Preprocessor and add SQLITE_HAS_CODEC to Preprocessor Definitions.

New Berkeley DB SQL API PRAGMAs

Two new Berkeley DB SQL API specific pragmas have been added, bdbsql_shared_resources and bdbsql_set_lock_tablesize. bdbsql_shared_resources is used to set the maximum amount of memory, in bytes, to be used by shared structures in the main environment region, which is useful in applications with a large number of tables, transactions, or threads. bdbsql_set_lock_tablesize is used to set the number of buckets in the lock object hash table in the Berkeley DB environment, which is useful if an application has many concurrent long running transactions.

New PRAGMAs

- PRAGMA bdbsql_shared_resources[=N]
- PRAGMA bdbsql_set_lock_tablesize[=N]

Replication for Existing Databases in the SQL API

Replication can now be enabled on existing SQL databases, and replication is now disabled permanently instead of temporarily. Replication is enabled on an existing database the same way it is enabled on a new database, with one restriction. The existing database must configure itself as the initial master of a new replication group. To disable replication on a database permanently, use pragma replication=OFF;.

PRAGMAs With Permanent Effects

- pragma replication=OFF;

PRAGMAs That Can Now Operate on Existing Databases

- pragma replication_local_site="host:port"
- pragma replication_initial_master=ON
- pragma replication=ON

Berkeley DB X/Open Compliant XA Resource Manager and Transaction Snapshots

The transactions managed by the Berkeley DB X/open compliant XA resource manager can now be enabled for transaction snapshots. To enable snapshots open an XA managed database with
the flag, DB_MULTIVERSION, and all XA managed transactions that operate on that database will use transaction snapshots.

**Berkeley DB Library Version 11.2.5.3 Change Log**

This is the changelog for Berkeley DB 11g Release 2 (library version 11.2.5.3).

### Database or Log File On-Disk Format Changes

1. Existing database file formats were unchanged in library version 11.2.5.3.
2. The log file format changed in library version 11.2.5.3.

### New Features

1. Added support for verifying named in-memory dbs. [#16941]
2. Added an integer key comparison function to improve performance through the SQL API. [#19609]
3. Support build on the platforms where pthread_t is a struct. [#19876]
4. Added an API call so the user can specify the size of the region in a heap db. [#19914]
5. Improved Replication Manager's ability to recover from the (perhaps rare) phenomenon of two sites trying to connect to each other simultaneously, which used to result in loss of both connections, requiring a retry after the CONNECTION_RETRY timeout period. [#19980]
6. Enhanced the interface for copying databases for a hot backup. Added configure support for --enable-atomicfileread. [#20129]
7. Enhaced the log reading routine to detect that a log file is missing rather than returning that a zero length record was found. [#20130]
8. Added pragma bdbsql_shared_resources to set or report the maximum amount of memory to be used by shared structures in the main environment region and bdbsql_lock_tablesize to set or report the number of buckets in the lock object hash table. These are advanced tuning features for applications with large number of tables or needs to reduce locking on concurrent long running transactions. [#20156]
9. Added set_metadata_dir() and get_metadata_dir() to enable storage of persistent metadata files in a location other than the environment home directory. [#20174]
10. Improved the error handling through the SQL API. Errors can be sent to a file with the use of the BDBSQL_ERROR_FILE pragma. [#20213]
11. Database handles can now be configured to give exclusive access to the database. [#20331]
12. XA transactions will now use transaction snapshots if the XA databases they operate on were configured with DB_MULTIVERSION. [#20332]
13. Added additional stats fields into the C# API [#20693]

14. Added pragma bdbsql_single_process to keep the Berkeley DB environment information on the heap instead of in shared memory. This option cannot be used if the database is accessed from multiple processes. [#20789]

15. Improved the ability of DB->compact to move DB_HASH database pages to the beginning of the file. [#20815]

**Database Environment Changes**

1. Fixed a bug that could cause a segmentation violation when closing an environment handle which has open database handles on partition databases. [#20281]

2. Fixed a bug that could cause a segmentation violation if a region was extended leaving a very small fragment at the end. [#20414]

3. Changed the behavior of the DB_REGISTER | DB_RECOVER flag combination, so that recovery is always run if the environment panic bit is set. [#20767]

**Access Method Changes**

1. Fixed a bug were database configuration settings could be lost when the database was opened if the open operation was blocked for any amount of time. [#20860]

2. Fixed a bug that bulk update operations did not work correctly on compressed databases. [#19017]

3. Improved the log flushing performance when ftruncate() is not available on a system. [#19725]

4. When performing partial puts in a heap database, empty pieces will no longer be left in a split record chain. [#20052]

5. Fixed a bug where, on systems without FTRUNCATE, db_verify will return an error for truncated heap databases. [#20195]

6. Fixed a bug where BDB could run out of mutexes when many databases are renamed. [#20228]

7. Fixed a bug where the metadata page in hash databases would not be flushed to disk. [#20265]

8. Fixed a bug that could leave deleted pages from a HEAP database in the buffer cache. [#20309]

9. Fixed a bug where the library would fail to put records with overflow keys into hash duplicate database. [#20329]

10. Fixed a bug in DB->compact of btrees that could cause a bad pointer reference. [#20360]
11. Fixed a bug that could cause the last page number stored on the metadata page to be wrong after rolling forward a db->compact operation that freed more pages than will fit in a single log record. [#20646]

12. Fixed a bug that could cause DB->stat to block on a mutex while holding a lock on the metadata page. [#20770]

13. Fixed a bug that could cause DB->compact of a DB_HASH database to fail to mark a page it updated as dirty. [#20778]

14. Fixed a bug where internal HEAP structures were not rebuilt during database handle refresh. [#20821]

15. Fixed a bug with secondary indices, off-page duplicates and DB_READ_COMMITTED which could erroneously release the lock on the page holding a returned record. [#20853]

16. Fixed a bug that could cause a hang or improperly report an empty queue when the queue record numbers wrapped around at 2^32. [#20956]

17. Fixed a bug on Linux or Windows that could generate a checksum error if a database file was being opened while the meta data page happened to be flushed from the cache. [#20996]

**SQL API Changes**

1. Fixed several memory leaks in the Online Backup API. [#19850]

2. Fixed a bug in the SQL API when using large blob items and multiple concurrent connections. [#19945]

3. To avoid a race condition that could cause a snapshot reader to see a wrong version it is now not permitted to open a DB handle specifying DB_MULTIVERSION if that database is currently opened by a handle which did not specify DB_MULTIVERSION. [#19940]

4. Pragma replication=on can now enable replication on an existing database. Turning replication off is now permanent. [#20180]

5. Fixed a bug in the SQL API where it was possible for a schema update to be ignored when accessing a database from multiple processes. [#20319]

6. Fixed a bug where aborting an exclusive transaction followed by an auto-commit read operation causes an assert failure. [#20567]

7. Fixed a bug in the SQL API where using the journal_mode pragma could cause a crash when used as the first operation in a connection on an existing database. [#20620]

8. Turn off the DBSQL encryption option on Windows/WinCE by default to match the behavior on the other platforms. [#20671]

9. Renamed the BDBSQL_OMIT_SHARING preprocessor flag to BDBSQL_SINGLE_PROCESS. [#20789]
10. Fixed a bug dealing with handle lock modes not reflecting the correct state which was causing a deadlock in the SQL API. [#20862]

**Java-specific API changes**

1. Added ReplicationManagerConnectionStatus class and ReplicationManagerSiteInfo.getConnectionStatus(). Deprecated ReplicationManagerSiteInfo.isConnected(). [#18068]

2. Updated EID_MASTER to be "public static final" so that it would be exposed in Java docs. [#20184]

3. Fixed a bug where calls that return Stat objects could cause a segfault. [#20377]

**Replication Changes**

1. Fixed quorum computation when most sites are unelectable. [#15251]

2. Made Replication more resilient to random input on its port. [#15712]

3. Fixed a bug where the datadir structure was not maintained during an internal init. [#19041]

4. Fixed a repmgr memory leak when using DB_PRIVATE. [#19363]

5. Fixed a minor bug to handle ENOMEM when using an in-memory temp database. [#20197]

6. Fixed a bug where multiple long running transactions across checkpoints could cause Log Sequence errors on client systems. [#20421]

7. Fixed a bug where multiple Replication Manager processes would sometimes not all conform to replication-group-aware log archiving. [#20342]

**Locking Subsystem Changes**

1. Fixed a bug that could cause an early lock timeout if a previous error left a lock timeout value set. [#19973]

**Logging Subsystem Changes**

1. Fixed a bug which could cause an incompletely written log record to be recognized as valid, resulting in recovery failing with the message “Illegal record type <integer, usually 0> in log”. [#17851]

2. Fixed a bug where printlog would fail on in-memory heap databases. [#20269]

**Memory Pool Subsystem Changes**

1. Fixed a bug which overstated the number of clean and dirty pages evicted from the cache. [#20410]

2. Fixed a bug that left a small fragment at the end of a region when extending. [#20414]
3. Fixed a bug where the file bucket was always zero when creating a mpoolfile using the mpool API. [#20468]

4. Fixed a bug with multiversion concurrency control which could cause versions of pages to remain in the cache even though they are no longer needed. [#20570]

5. The memory pool allocator will now start freezing MVCC versions of buffers if it sees more than 1/4 of the available buffers are taken up by versions. [#20836]

**Mutex Subsystem Changes**

1. Fixed a bug in which DB_ENV->mutex_set_align() could cause DB_ENV->mutex_stat_print(dbenv, DB_STAT_ALL) to display only the first mutex. [#20522]

2. Fixed a bug with DB_ENV->mutex_stat_print() in which the information on some mutexes would not be displayed, if any mutex had been freed and not yet reallocated. [#20533]

**Transaction Subsystem Changes**

1. Fixed a bug where a malloc failure could result in a segfault when doing a put on a database with secondaries. [#20641]

**Utility Changes**

1. Fixed a bug that would cause verify to call the wrong compare function if there are user defined compare functions used and the database has multilevel off page sorted duplicate trees. [#20284]

2. Fixed a bug that could cause recovery to fail if DB->compact moved the meta data page of a HASH subdatabase. [#20708]

**Configuration, Documentation, Sample Apps, Portability and Build Changes**

1. The DB_CONFIG configuration commands which specify directory pathnames ("set_data_dir", "set_lg_dir", and "set_tmp_dir") now accept names containing whitespace characters. [#20158]

**Known Bugs**

1. If two SQL processes are concurrently altering the schema of the same tables in a database, there is a race condition that can cause the application to hang. [#20513]

2. Replication groups including machines of different endianness do not support the heap access method. [#21016]

3. If a txn that is attempting to remove a region page from a heap database is aborted and another txn is trying to update that same page then it can cause the original txn to abort. This is timing dependant. [#20939]
Chapter 13. Upgrading Berkeley DB 11.2.5.1 applications to Berkeley DB 11.2.5.2

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 11.2.5.1 library to the Berkeley DB 11.2.5.2 library (both of which belong to Berkeley DB 11g Release 2). This information does not describe how to upgrade Berkeley DB 1.85 release applications.

For information on the general process of upgrading Berkeley DB installations and upgrade instructions related to historical releases, see the Berkeley DB Upgrade Guide.

SQLite Interface Upgrade

Berkeley DB’s SQL interface includes code from SQLite. The version of SQLite used has been upgraded, so DB SQL is compatible with SQLite version 3.7.6.2. Please see the release notes at http://sqlite.org/changes.html for further information.

32bit/64bit Compatibility on Windows

Berkeley DB can now be compiled on Windows so that 32 bit and 64 bit applications can concurrently access a BDB environment. To enable this feature, build both the 32 bit BDB library and application and the 64 bit library and application with the flag /D HAVE_MIXED_SIZE_ADDRESSING. Note that private environments are disabled under the compatibility mode.

Read Only flag for DBT

A DBT can now be set as read-only, when passed to the DB->get() method, using the flag DB_DBT_READONLY. This is useful when using a static string as a key value, because this flag will prevent Berkeley DB from updating the DBT.

New Flag

• DB_DBT_READONLY

Dynamic Environment Configuration

Memory is now allocated incrementally as needed, instead of all at once during environment initialization, for structures that support locks, transactions, threads, and mutexes. With this change new functions have been added that configure how much memory is allocated initially, and how much that memory is allowed to grow. The old memory configuration functions have been deprecated.

New Functions

• DB_ENV->set_memory_init()
• DB_ENV->get_memory_init()
• DB_ENV->set_memory_max()
• DB_ENV->get_memory_max()
• DB_ENV->set lk_tablesize()
• DB_ENV->get lk_tablesize()
• DB_ENV->mutex_set_init()
• DB_ENV->mutex_get_init()

Deprecated Functions
• DB_ENV->mutex_set_max()
• DB_ENV->mutex_get_max()
• DB_ENV->set lk_max_lockers()
• DB_ENV->get lk_max_lockers()
• DB_ENV->set lk_max_locks()
• DB_ENV->get lk_max_locks()
• DB_ENV->set lk_max_objects()
• DB_ENV->get lk_max_objects()
• DB_ENV->set thread_count()
• DB_ENV->get thread_count()
• DB_ENV->set tx_max()
• DB_ENV->get tx_max()

Exclusive Transactions in the SQL Layer
Issuing the SQL command BEGIN TRANSACTION EXCLUSIVE will now cause any other transactions accessing the database to block, or return a SQLITE_BUSY or SQLITE_LOCK error, until the exclusive transaction is committed or aborted. Previously, non-exclusive transactions could execute concurrently with an exclusive transaction.

Group Membership in Repmgr
Replication Manager now manages group membership much more closely, making it much easier for applications to add and remove sites from a replication group without risk of transaction loss. In order to accomplish this, the API for configuring group membership has
changed significantly. The repmgr_set_local_site() and repmgr_add_remote_site() methods no longer exist; they are replaced by a new handle type, DB_SITE. The repmgr_get_local_site() method has been replaced by DB_ENV->repmgr_site(), which now returns a DB_SITE handle instead of a raw host/port network address.

Replication Manager applications may no longer call the DB_ENV->rep_set_nsites() method, because the Replication Manager now tracks the number of sites in the replication group for you. Replication Manager applications may still call DB_ENV->rep_get_nsites(), but only after a successful call to DB_ENV->repmgr_start().

For applications using the replication Base API there is no change, except that they may now call DB_ENV->rep_set_nsites() to change the group size even when Master Leases are in use.

The new Replication Manager group membership functionality is described in the Managing Replication Manager Group Membership chapter in the Berkeley DB Programmer’s Reference Guide.

Replication Manager no longer prints an error message on a connection failure. Instead it generates an event with the equivalent information (invoking the application’s event-handling call-back function).

Upgrading

An existing application running a previous version of BDB can do a “live upgrade” so that only one site at a time has to be shut down. To do this, restart each site in the group, with the old master being shutdown last. When each site is restarted, use DB_SITE to configure the local site with the flag DB_LEGACY, and create a DB_SITE handle with a full specification of all the remote site addresses for all other sites currently in the group, and configure each handle with the DB_LEGACY flag. When the old master is restarted and a new master has been established, the new master is ready to manage membership changes, and new sites can be added as usual. But the application must not try to add new sites, or remove existing sites, during the mixed-version transitional phase.

To do a non-live upgrade shutdown the entire replication group. Then restart the group with each site configured with the DB_LEGACY flag, and in DB_REP_ELECTION mode.

New Functions

• DB_ENV->repmgr_site()
• DB_ENV->repmgr_site_by_eid()
• DB_SITE->set_config()
• DB_SITE->get_config()
• DB_SITE->remove()
• DB_SITE->get_eid()
• DB_SITE->get_address()
• DB_SITE->close()

**Modified Functions**

• DB_ENV->rep_set_nssites() is no longer used by the Replication Manager, but is still used by the Base API. It can now be used to change the number of sites dynamically, even when master leases are in use.

**New Events**

• DB_EVENT_REP_SITE_ADDED
• DB_EVENT_REP_SITE_REMOVED
• DB_EVENT_REP_LOCAL_SITE_REMOVED
• DB_EVENT_REP_CONNECT_BROKEN
• DB_EVENT_REP_CONNECT_ESTD
• DB_EVENT_REP_CONNECT_TRY_FAILED
• DB_EVENT_REP_INIT_DONE

**Removed Functions**

• DB_ENV->repmgr_set_local_site()
• DB_ENV->repmgr_add_local_site()
• DB_ENV->repmgr_add_remote_site()
• DB_ENV->repmgr_get_local_site()

**New Parameters**

The following new parameters are passed to DB_SITE->set_config().

• DB_BOOTSTRAP_HELPER
• DB_GROUP_CREATOR
• DB_LEGACY
• DB_LOCAL_SITE
• DB_REPMGR_PEER

**New Structure**

• DB_REPMGR_CONN_ERR encapsulates an EID and an integer system error code.
Heap Access Method

Databases can now be configured as heaps by passing the access type DB_HEAP to DB->open(). Heap size can be configured with DB->set_heapsize(), and DB->stat() now returns heap statistics in the structure DB_HEAP_STAT when applied to a heap database.

New Functions

- DB->set_heapsize()
- DB->get_heapsize()

Modified Functions

- DB->open() now accepts DB_HEAP as an access type.
- DB->stat() now returns heap statistics in the structure DB_HEAP_STAT.

New Definition

- DB_HEAP_RID is the defined heap key value.

Enabling Transaction Snapshots in the SQL Layer

Read/write concurrency can now be enabled in the SQL API by using PRAGMA multiversion=on before accessing any tables in the database. After multiversion has been enabled, it can be temporarily disabled using the PRAGMA transaction_snapshots=on/off.

New Pragmas

- PRAGMA multiversion=ON|OFF;
- PRAGMA snapshot_isolation=ON|OFF

2SITE STRICT Enabled by Default in Replication

The 2SITE STRICT replication configuration parameter is now turned on by default. This configuration parameter is controlled using the DB_REPMGR_CONF_2SITE STRICT. flag on the DB_ENV->rep_set_config() method.

Enabling Replication in the SQL Layer

Replication can now be enabled and configured in the SQL layer using pragmas. The pragmas replication_local_site, replication_initial_master, and replication_remote_site can be used to configure the replication group. Note that when the BDB SQL replicated application is initially started, a specific master site must be explicitly designated. After configuring the replication group, start replication using PRAGMA replication=ON.

To display replication statistics in the dbsql shell, use:

dbsql> .stat :rep
New Pragmas

- PRAGMA replication=ON|OFF
- PRAGMA replication_initial_master=ON|OFF
- PRAGMA replication_local_site="hostname:port"
- PRAGMA replication_remote_site="hostname:port"
- PRAGMA replication_remove_site="host:port"
- PRAGMA replication_verbose_output=ON|OFF
- PRAGMA replication_verbose_file=filename

Repmgr Message Channels

Application components running at various sites within a replication group can now use the Replication Manager's existing TCP/IP communications infrastructure to send and process messages among themselves, using the DB_CHANNEL handle. DB_ENV->repmgr_channel() is used to create the DB_CHANNEL handle. DB_CHANNEL->send_msg() and DB_CHANNEL->send_request() are used to send synchronous and asynchronous messages that are handled by the function set by DB_ENV->repmgr_msg_dispatch(). DB_CHANNEL->set_timeout() is used to configure channel time out, and DB_CHANNEL->close() closes the channel and frees resources held by it.

New Functions

- DB_ENV->repmgr_msg_dispatch()
- DB_ENV->repmgr_channel()
- DB_CHANNEL->send_msg()
- DB_CHANNEL->send_request()
- DB_CHANNEL->set_timeout()
- DB_CHANNEL->close()

Sequence Support in the SQL Layer

A partial implementation of the sequence API defined in the SQL 2003 specification has been added to the SQL layer. A sequence is created using the syntax:

```
SELECT create_sequence("sequence_name"...)
```

The sequence numbers are accessed using

```
SELECT nextval("sequence_name")
```
and

```
SELECT currval("sequence_name")
```

Finally, a sequence can be dropped using

```
SELECT drop_sequence("sequence_name")
```

**New Functions**

- `create_sequence`
- `seq_nextval`
- `seq_currval`
- `seq_drop_sequence`

**Berkeley DB X/Open Compliant XA Resource Manager**

The Berkeley DB X/Open compliant XA resource manager has been restored. (It was removed from the product after the 4.7 release.) The new implementation includes support for multi-threaded servers. Consult the documentation of your chosen transaction manager to learn how to implement a multi-threaded server.

**Constraints**

Applications that use a BDB XA resource manager must now take into account the following constraints.

- No in-memory logging.
- No application-level child transactions.
- All database-level operations (open, close, create and the like) must be performed outside of a global transactions (i.e., they can be performed in local BDB transactions, but not while a distributed XA transaction is active).
- Environment configuration must be done using a DB_CONFIG file.
- Cursors must be closed before a service invocation returns.

**New Flag**

- `DB_XA_CREATE` - This flag is passed to `db_create()` to create a DB handle that supports XA transactions.

**Modified Function**

- `DB->stat()` now returns the field `DB_TXN_STAT->DB_TXN_ACTIVE->xa_status`, which contains information on the XA transactions.
Hot Backup Changes

Because non-UNIX systems do not support atomic file system reads, the db_hotbackup utility has been modified to read data through the environment. If your application is running on a UNIX based system such as Solaris, HPUX, BSD or Mac OS, you can specify the -F flag to read directly from the filesystem. Please refer to Recovery procedures in the Berkeley DB Programmer’s Reference Guide for more information on safely backing up your databases.

Berkeley DB Library Version 11.2.5.2 Change Log

This is the changelog for Berkeley DB 11g Release 2 (library version 11.2.5.2).

Database or Log File On-Disk Format Changes

1. Existing database file formats were unchanged in library version 11.2.5.2. However, a new database file format, “heap”, was introduced.

2. The log file format changed in library version 11.2.5.2.

New Features

1. Replication Manager now manages Group Membership. This allows sites to be added to and removed from the replication group dynamically. Replication Manager also now automatically keeps track of the group size (nsites). [#14778]

2. Initial allocations for various non-pagebuffer (mpool) system resources may now be specified, as well as a total maximum of memory to use, rather than specifying a maximum value for each resource. [#16334]

3. Implemented Berkeley DB globalization support architecture to enable localized and stripped error and output messages. [#16863]

4. Added a new access method, DB_HEAP. Heap aims for efficient use (and re-use) of disk space. Keys in a heap database are automatically generated by BDB, it is recommended that one or more secondary indexes be used with a heap database. For full details on DB_HEAP, see the Programmer’s Reference Guide. [#17627]

5. Added a compatible mode for 32bit and 64bit Windows environment. [#18225]

6. For the SQL API, concurrency between read and write transactions can now be enabled using “PRAGMA multiversion”. Added several pragmas that can be used to configure the Berkeley DB datastore. [#18521]

7. Add several new pragmas to provide in-process support for replication in the SQL API. [#18528]

8. The Berkeley DB X/Open compliant XA resource manager has been restored, including support for multi-threaded servers. [#18701]

9. Improved the ability to recover from an application crash on connections through the SQL API. Berkeley DB will try to automatically clean up locks, mutexes and transactions from the failed process. [#18713]
10. Add support for sequence usage in the SQL API using SQLite custom functions. [#19007]

11. Add a pragma in the SQL API to allow execution of a cache trickle command. [#19202]

12. Add a pragma in the SQL API to allow configuration of DB_SYSTEM_MEM environments. [#19249]

13. The new db_env_set_win_security(SECURITY_ATTRIBUTES *) function allows an application to specify the particular Microsoft Windows security attributes to be used by Berkeley DB. This helps support applications which reduce their privileges after opening the environment. [#19529]

**Database Environment Changes**

1. None

**Concurrent Data Store Changes**

1. None

**Access Method Changes**

1. Modified the queue access method so that it only uses latches on the metadata page rather than a latch and a lock. This was done to improve performance. [#18749]

2. Fixed several bugs that could cause an update running with MVCC to get the wrong version of a page or improperly update the metadata last page number. [#19063]

3. The database open code will no longer log the open and close of the master database in a file when opening a sub database in that file. [#19071]

4. Fixed a bug where an error during an update to a hash database with DB_NOOVERWRITE set could return DB_KEYEXIST rather than the correct error. [#19077]

5. Fixed a bug that could cause the wrong page number to be on a root or metadata page if DB->compact moved the page and the operation was later rolled forward. [#19167]

6. Fixed a bug that could cause the close of a secondary index database to fail if the transaction doing the open aborted. [#19169]

7. Fixed a bug that could prevent an update to a primary recno or queue database with DB_NOOVERWRITE set. [#19230]

8. Fixed a bug when an update to a database with DB_NOOVERWRITE set could incorrectly return DB_KEYEXIST rather than the correct error (e.g., DB_LOCK_DEADLOCK). [#19345]

9. Fixed a bug preventing the use of the set_re_len and set_re_pad methods with a RECNO database when configuring with --disable-queue. [#19367]

10. Fixed a bug in DB->compact on BTREE databases that did not check if the last page in the database could be moved to a lower numbered page. [#19394]
11. Fixed a bug that could cause a Log Sequence Error when recovering the deallocation of a multiple page overflow chain. [#19474]

12. Fixed a bug that could cause a diagnostic assertion if MVCC was in use and multiple levels of a btree needed to be split. [#19481]

13. Fixed a few error paths that could cause a Panic with an "unpinned page returned" error. [#19493]

14. Fixed a bug that closed a race condition that under heavy mult-threaded appending to a queue database could cause some records to be lost. [#19498]

15. Fixed a bug that might cause DB->compact to mis-estimate the size of an overflow record when merging two pages. This may cause the page to have more data than desired. [#19562]

16. Fixed a bug in DB_ENV->fileid_reset that did not update the fileid's on the metadata pages of subdatabases if the database file was not in native byte order. [#19608]

17. Fixed a bug that caused the first directory specified in the create of a partitioned database to get too many partitions. [#20041]

**SQL API Changes**

1. Fixed a race condition that would cause a corruption error in one process when two processes created the same SQL database. [#18929]

2. Fixed a bug that would cause a constraint violation when updating the primary key with the same value. [#18976]

3. Overwriting an old backup with a new backup using the SQL online backup API will no longer double the size of the database. [#19021]

4. Implemented index optimizations for indexes on large values. [#19094]

5. Fixed a bug that could cause an undetected deadlock between a thread which moved a metadata or root page via a DB->compact operation and another thread trying to open the database if the old page was being removed from the file. [#19186]

6. Fix a bug in the BDBSQL_FILE.PER_TABLE option, to allow absolute path names. [#19190]

7. Add a pragma to allow configuration of DB_SYSTEM_MEM environments. [#19249]

8. Exclusive transactions will now block new transactions and will prevent existing transactions from making forward progress. [#19256]

9. Fixed a bug that would cause assert error when opening an in-memory hash database with thread count configured when compiled with --enable-diagnostic. [#19357]

10. Upgrade the bundled version of SQLite to 3.7.6.2 [#19376]
11. Fixed a performance bug with the cache victim selection algorithm when there were multiple cache regions. [#19385]

12. Fixed a bug which could cause two SQL threads to have an undetected deadlock when opening or closing tables. [#19386]

13. Fix a bug that could cause a hang when deleting a table if there are multiple connections to a database from different processes. [#19419]

14. Fixed a bug which could cause multiple threads performing DB->compact on the same database file to overrun the in-memory freelist, which could potentially lead to memory corruption. [#19571]

15. Fixed a bug in DB->compact that could cause a loop if an attempt to move a sub-database meta data page deadlocked. [#20028]

**C API Changes**

1. Fixed a bug where encryption could not be enabled for individual databases in an encrypted environment. [#18891]

2. Removed two unused error codes, DB_NOSERVER_HOME and DB_NOSERVER_ID. [#18978]

3. Added a DB_DBT_READONLY flag so that users can pass in a non-usermem key (DB_DBT_USERMEM) for get operations. [#19360]

4. Fixed a bug in DB/DBC->get/pget that the partial flags are silently ignored with positional flags and return inconsistent DBT. [#19540]

5. Fixed a bug which prevented items from being deleted on a secondary database. [#19573]

6. Fixed a bug to correctly handle the DB_BUFFER_SMALL case on delete operations when compression is enabled. [#19660]

**Tcl-specific API Changes**

1. None.

**C#-specific API Changes**

1. Added support for partial put/get in the C# API. [#18795]

2. Fixed a bug in compare delegate for secondary db. [#18935]

**Replication Changes**

1. Replication Manager now allows differing ack policies at different sites throughout the group, and supports dynamic changes to the ack policy. (The ack policy in force is determined by the current master.) [#14993]

2. Replication Manager "channels" feature allows applications to share repmgr's communication facilities. [#17228]
3. Add example program for RepMgr "channels" feature: ex_rep_chan. [#17387]

4. Replication Manager now allows dynamic changes to a site’s "electability" (changes between zero and non-zero priority). This feature should be used with care, because electability changes can in boundary cases invalidate durability guarantees granted for previous transactions. [#17497]

5. Changed election criteria so that later group transactions won’t get overwritten by earlier generations with more log. [#17815]

6. Added changes to master lease checks that result in improved performance when using master leases. [#18960]

7. A log write failure on a replication master will now cause a panic since the transaction may be committed on some clients. [#19054]

8. Fixed a few memory leak conditions on error paths. [#19131]

9. Change lease code so that zero priority sites do not count in lease guarantees since they cannot be elected. [#19154]

10. Repmgr rerequest processing is moved from a dedicated thread to heartbeat messages. Repmgr clients using heartbeats can now detect and rerequest missing final master log records without master activity. [#19197]

11. Repmgr statistics are now included in full statistics output for an environment. [#19198]

12. Fix an inefficiency in mixed version elections. We now check if an election is won via the EID instead of priority. [#19254]

13. Changed election LSNs to use the last txn commit LSN instead of the end of the log. [#19278]

14. Create replication internal database files in the environment home directory rather than the data directory so that they are in the same location as the other internal replication files. [#19403]

15. Fix a bug that was preventing repmgr from calling an election when starting a site with the DB_REP_ELECTION flag. [#19546]

16. Fixed a bug which could cause a segfault at a replication master if a named in-memory database was being created around the same time as a client site were synchronizing (in “internal init”) with the master. [#19583]

17. Adjust lease code to consider timeout length when retrying. [#19705]

18. Fixed a bug that could cause a crash in replication groups of more than 10 sites, with multiple processes sharing each DB environment concurrently. [#19818]

19. Fix a bug where an assertion failure could happen if pages in a database were deallocated during a client internal initialization.[#19851]
20. Fix a bug where an internal initialization of a queue database with non-contiguous extent files could return an error. [#19925]

21. The 2SITE_STRICT replication configuration parameter is now turned on by default. It can be turned off via a call to DB_ENV->rep_set_config(). [#19937]

22. Repmgr heartbeats can now help detect a duplicate master without the need for application activity. [#19950]

**Locking Subsystem Changes**

1. Fixed a bug where an updater supporting DB_READ_UNCOMMITED might downgrade its lock too soon if there was an error during the update. [#19155]

2. Fixed a bug where transaction timeouts could have been specified in a database environment where the locking subsystem was disabled. [#19582]

3. Fixed a bug in a diagnostic assertion that was improperly triggered by the removal of a sub-database. [#19683]

4. Fixed a bug that would cause DB_ENV->failcheck to free locks for a locker associated with a database handle after the thread that opened the handle exited. [#19881]

**Logging Subsystem Changes**

1. Enhanced recovery so that it will not output extra checkpoint or transaction id recycle log records if there was no activity since the last checkpoint. [#15330]

2. Log checksums can now be disabled using the compile argument --disable-log-checksum. This will give a performance increase at the risk of undetectable corruption in the log records, which would make recovery impossible. [#19143]

3. Fixed a bug that could cause a page that should have been removed from the end of a file still be in the copy of the file in a hot backup. [#19996]

**Memory Pool Subsystem Changes**

1. Fixed a bug in MPOOLFILE->get that did not permit the DB_MPOOL_DIRTY flag to be used with other flags. [#19421]

**Mutex Subsystem Changes**

1. Fixed a bug when the mutex region needs to be larger than 4GB, the region size was incorrectly adjusted to be slightly too small to fit the mutexes. [#18968]

2. Fixed a performance problem with hybrid shared latches in which a request for exclusive access would busy-wait (rather than put itself to sleep) if the latch were held by a shared reader. This also fixed the timeout handling of hybrid mutexes. In some cases the timeout would not be honored, resulting in delays for the replication “read your writes” feature which were longer than requested. [#18982]
3. Fixed the timeout handling of the pthreads mutexes used by the replication "read your writes" feature. When a timeout occurred there was a race condition which might result in a hang. [#19047]

Transaction Subsystem Changes

1. Fixed a leak of log file ids when a database is closed before the end of a transaction that references it. [#15957]

2. Fixed a bug that would cause a panic if a child transaction performed a database rename, then aborted, and then the parent transaction committed. [#18069]

3. Fixed a bug where we released the metadata page lock too early if a non-transactional update was being done. [#19036]

4. Removed the possibility that checkpoints will overlap in the log, decreasing the time to recover. [#19062]

Test Suite Changes

1. Require Tcl 8.5 or greater.

Utility Changes

1. Added a new utility, db_tuner, which analyzes the data in a btree database, and suggests a reasonable pagesize. [#18910]

2. Fixed some bugs in log_verify when there are in-memory database logs and subdb logs. [#19157]

3. Modified db_hotbackup to not read from the file system as required on non-UNIX systems. Also provided the db_copy function for this purpose. [#19863]

4. Fixed db_hotbackup so that when -d/-l or -D is not specified, DB_CONFIG is used to determine the locations of the databases and logs in the source environment. [#19994]

Configuration, Documentation, Sample Apps, Portability and Build Changes

1. Changed SQL API library built on *nix to link with libpthreads when necessary. [#19098]

2. Added CPPFLAGS into our --enable-jdbc configuration. [#19234]

3. Added encryption support into the Windows CE build project for SQL API. [#19632]

4. Fixed a bug in the STAT_INC_VERB() dtrace probe that was causing compiler warnings. [#19707]

5. Fixed a bug that could cause a trap in db_dump using salvage mode if a page was found that was not associated with any database in the file. [#19974]

6. On Cygwin, circumvented a bug in libtool that is exposed when building the BDB SQL API in a directory path containing whitespace characters. [#19812]
Example Changes

1. Update repmgr C, C#, C++, Java examples(ex_rep_mgr, ex_rep_gsg_repmgr, ex_rep_chan, excs_repquote, excxx_repquote, excxx_epquote_gsg, repquote, repquote_gsg) with their related API changes for group membership. [#19586][#19622]

2. Port ex_rep_chan, ex_rep_gsg_repmgr, ex_rep_gsg_simple, excxx_repquote_gsg_repmgr, excxx_repquote_gsg_simple to Window.[#19890]

Miscellaneous Bug Fixes

1. Fixed a bug where memory copied from the Java API could leak if flags were not correctly configured. [#19152]

Deprecated Features

1. None

Known Bugs

1. The SQL API has a known issue when using a blob field with a lot of content and multiple concurrent connections to the database. [#19945]

2. Rollback of a dropped table in the SQL layer contains a mutex leak, which can consume all mutex resources if enough rollbacks of table drops are performed. [#20077]

3. The DB_CONFIG configuration parameters which specify path names currently do not support names containing any whitespace characters. [#20158]

4. The BFile module has a known crash issue when using BFile handle for SQL expressions interface on 64bit platforms. [#20193]

5. On systems without FTRUNCATE, db_verify will return an error for truncated heap databases. This is a bug in db_verify, the database has been truncated correctly and can be used in the future. [#20195]

6. An application using queue extents which is append mostly could see a decrease in the buffer pool hit rate due to the failure to remove pages from closed extents from the buffer pool. [#20217]
Chapter 14. Upgrading Berkeley DB 11.2.5.0 applications to Berkeley DB 11.2.5.1

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 11.2.5.0 library to the Berkeley DB 11.2.5.1 library (both of which belong to Berkeley DB 11g Release 2). This information does not describe how to upgrade Berkeley DB 1.85 release applications.

For information on the general process of upgrading Berkeley DB installations and upgrade instructions related to historical releases, see the Berkeley DB Upgrade Guide.

DPL Applications must be recompiled

Applications that use the Java interface's Direct Persistence Layer must be recompiled, due to a change in the return type of the setter methods in StoreConfig and EvolveConfig classes. The setter methods now return this instead of void.

Source Tree Rearranged

The source code hierarchy has been reorganized. Source files that belong to Berkeley DB core are now in a top-level directory named src. Files related to language interfaces are in lang, and all examples are collected under examples.

SQLite Interface Upgrade

Berkeley DB's SQL interface includes code from SQLite. The version of SQLite used has been upgraded, so DB SQL is compatible with SQLite version 3.7.0.1. Please see the release notes at http://sqlite.org/changes.html for further information.

Mod_db4 Support Discontinued

The mod_db4 apache module is no longer included in the release.

Berkeley DB Library Version 11.2.5.1 Change Log

This is the changelog for Berkeley DB 11g Release 2 (library version 11.2.5.1).

Database or Log File On-Disk Format Changes

1. The database file format was unchanged in 11gR2 library version 11.2.5.1.

2. The log file format was unchanged in 11gR2 library version 11.2.5.1.

New Features

1. Added Performance event monitoring support for DTrace and SystemTap which can be enabled during configuration. Static probes have been defined where statistics values
are updated, where mutex or transactional consistency lock waits occur, and where some other potentially lengthy operations may be initiated. [#15605]

2. Added a new acknowledge policy - DB_REPMGR_ACKS_ALL_AVAILABLE. [#16762]

3. Added transactional bulk loading optimization for non-nested transactions. [#17669]

4. Added exclusive transaction support for the SQL API. [#17822]

5. Added support for bulk update and delete in C# API. [#18011]

6. Added a db_replicate utility. [#18326]

7. Added an implementation of the Online Backup API. [#18500]

8. Added support in Berkeley DB SQL for the vacuum and incremental vacuum pragmas [#18545]

9. Added an option to automatically convert SQLite databases to Berkeley DB on opening. [#18531]

10. Added BDBSQL_SHARE_PRIVATE, an option to enable inter-process sharing of DB_PRIVATE environments using multiple-reader. [#18533]

11. Added database-level locking to optimize single-threaded operations and remove locking limitations for database load operations. [#18549]

12. Added support for DB_INIT_REP, DB_PRIVATE and DB_THREAD in DB_CONFIG file. [#18555]

13. Added support for the BDBSQL_DEFAULT_PAGE_SIZE pragma to override Berkeley DB's choice of page size depending on the filesystem. Use SQLITE_DEFAULT_PAGE_SIZE rather than a hard-coded default. [#18577]

14. Added an extension that allows access to binary files stored outside of the database. What is stored in the database is a pointer to the binary file. [#18635]

15. Added .stat command to dbsql shell to print environment, table, and index statistics. [#18640]

16. Added enhancements to reduce the size of indexes in the SQL API by allowing duplicates in the index database and moving the rowid from the index key into the index data. [#18653]

17. Added a compile time flag BDBSQL_FILE_PER_TABLE that causes each table to be created in a separate file. This flag replaces the BDBSQL_SPLIT_META_TABLE flag. [#18664]

18. Added the handling of read only and read write open of the same database in BDB SQL [#18672]

19. Added an encryption implementation to the SQL API [#18683]
Database Environment Changes

1. Fixed failchk behavior on QNX. [#17403]
2. Fixed a bug that prevented the same process from connecting to the database after recovery is performed. [#18535]
3. Fixed a bug which would occur when recovery checkpoint was not written because the cache ran out of space attempting to flush the memory pool cache. The environment would be recovered and all database where made available, but some databases would incorrectly closed. This would cause a subsequent recovery to fail on its backward pass with the error “PANIC: No such file or directory”. [#18590]
4. Fixed a bug that could cause recovery to fail with the error “DB_LOGC->get: log record LSN %u/%u: checksum mismatch” if the last log file was nearly full and ended with a partially written log record which was smaller than a checkpoint record. It now erases the invalid partial record before switching to the new log file. [#18651]

Concurrent Data Store Changes

1. None

Access Method Changes

1. Fixed a bug such that segmentation fault does not occur if DB->set_partition_dirs is called before DB->set_partition. [#18591]
2. Fixed a bug such that the error "unknown path" does not occur if you put duplicate records into a duplicated sorted HASH database with DB_OVERWRITE_DUP option. [#18607]
3. Added the ability to specify that data should not be logged when removing pages from a database. This can be used if the ability to recover the data is not required after the database has been removed. [#18666]
4. Fixed a bug that caused an aborting transaction to fail if it aborted while a DB->compact of the same HASH database was compacting the dynamic hash table [#18695]
5. Fixed a bug that could cause DB->compact to loop on a DB_RECIDNO database or a database with an multilevel unsorted off page duplicate tree. [#18722]
6. Fixed a bug that could cause an illegal page type error when using a HASH database with MVCC and the HASH table was contracted and then extended. [#18785]
7. Fixed locking bugs: [#18789] The Db->compact method of BTREE with MVCC would return an unpinned page. The RECNO option would fail to lock the next page when splitting a leaf page.
8. Fixed a bug that could cause data to not be returned in a HASH database that was one of multiple databases in a file and was opened prior to running DB->compact method on that database in another thread of control [#18824]
9. Fixed a bug where doing a bulk insert with secondaries could return an error incorrectly. [#18878]

10. Fixed a bug that would return DB_NOTFOUND instead of DB_BUFFER_SMALL when the first item in a HASH database is larger than the user supplied buffer. [#18829]

API Changes

1. Fixed various items uncovered by extending DB_CONFIG support: [#18720] - Added missing set_cache_max method, and fixed name of log_set_config (was set_log_config). - Added new DB_ENV->repmgr_get_local_site method. - Fixed a bug which could fail to allocate enough mutexes when specifying a maximum cache size. - Fixed a bug that could allocate multiple caches when a small cache size was specified.

SQL-Specific API Changes

1. Allowed SQL applications to attach to the same database multiple times unless shared cache mode is explicitly requested. [#18340]

2. Fixed a bug where auto-removal of log files after writing a checkpoint was not functioning correctly. [#18413]

3. Fixed a race between opening and closing SQL databases from multiple threads that could lead to the error "DB_REGISTER limits processes to one open DB_ENV handle per environment". [#18538]

4. Optimized the SQL adapter for joins. Reduce the number of Berkeley DB operations in a join by caching the maximum key in the primary. [#18566]

5. A SQLITE_LOCKED or SQLITE_BUSY error returned by a statement in an explicit transaction will no longer invalidate the entire transaction, but just the statement that returned the error. [#18582]

6. Changed how multiple connections to the same database are detected. Used a fileid so that different paths can be used without error. [#18646]

7. Fixed a bug where the journal (environment) directory was being created prior to the actual environment. [#18656]

8. Added a new PRAGMA to allow tuning of when checkpoints are run. [#18657]

9. Fixed spurious “column <x> not unique” error messages. [#18667]

10. Fixed a segmentation fault that could happen when memory could not be allocated for the index key in the SQL API. [#18783]

11. Fixed a bug causing a segfault when releasing a savepoint that was already released. [#18784]

Tcl-Specific API Changes

1. Changed to link tcl8.5 by default on Windows [#18244]
Java-Specific API Changes

1. Fixed a bug where getAllowPopulate and getImmutableSecondaryKey method always returned false for SecondaryConfig objects returned by SecondaryDatabase.getSecondaryConfig method. [#16018]

2. Fixed a bug which made it impossible to (re)set VerboseConfig.REPLICATION_SYSTEM on the Java API. [#17561]

3. Fixed a bug where populating a SecondaryDatabase on open could lead to an OutOfMemoryException. [#18529]

4. Fixed a bug such that segmentation fault does not occur when putting records into callback-partitioned database. [#18596]

5. Fixed a bug where DatabaseConfig.getUnsortedDuplicates method returned true when the database had been configured for sorted duplicates. [#18612]

6. Initialized DatabaseConfig.pageSize so that it can be queried. [#18691]

7. Fixed a bug by opening a write cursor for Direct Persistent Layer(DPL) entity's put operation in the Concurrent Data Store product. [#18692]

8. Synchronized Java persistence code and tests from Java Edition to Berkeley DB. [#18711]

9. Introduced the EnvironmentConfig.setReplicationInMemory method as a way to configure in-memory internal replication files before opening the Environment handle on the Java API. [#18719]

10. Fixed a bug in the bulk DatabaseEntry class, where it was possible to overflow the buffer. [#18850]

11. Added LEASE_TIMEOUT field to the ReplicationTimeoutType class that enables configuring the amount of time a client grants its Master Lease to a master. [#18867]

C#-Specific API Changes

1. Fixed a bug in BTreeprefix comparison method such that there is no problem when the application needs to save a number larger than or equal to $2^{31}$. The BTreeprefix comparison function now returns an unsigned int instead of a signed int. [#18481]

2. Fixed a bug which caused the HasMultiple method to throw an exception when there were multiple databases in a single database file. [#18483]

3. Fixed a bug to ensure the CachePriority is set for Database and Cursor objects. [#18716]

4. Fixed a bug that use leading to the error: "Transaction that opened the DB handle is still active" when applications used different transactional handles in the associate and open methods in a secondary database. [#18873]
Direct Persistence Layer (DPL), Bindings and Collections API

1. All setter methods in the DPL `StoreConfig` and `EvolveConfig` now return `this` rather than having a `void` return type. This change requires that applications using the DPL be recompiled. [#17021]

2. Improve performance of `StoredCollection.removeAll()`. This method no longer iterates over all records in the stored collection. [#17727]

3. Several new tuple formats and binding classes have been added in the `com.sleepycat.bind.tuple` package:
   - Packed integer formats have been added that support default natural sorting. These are intended to replace the old unsorted packed integer formats.
   - Two new `BigDecimal` formats have been added. One format supports default natural sorting. The other format is unsorted, but has other advantages: trailing zeros after the decimal place are preserved, and a more compact, faster serialization format is used. See the `com.sleepycat.bind.tuple` package description for an overview of the new bindings and a comparative description of all tuple bindings. [#18379]

4. The following classes are now certified to be serializable. [#18738]
   - `com.sleepycat.persist.IndexNotAvailableException`
   - `com.sleepycat.persist.StoreExistsException`
   - `com.sleepycat.persist.StoreNotFoundException`
   - `com.sleepycat.persist.evolve.DeletedClassException`
   - `com.sleepycat.persist.evolve.IncompatibleClassException`

Replication Changes

1. Replication Manager now uses the standard system implementation of `getaddrinfo()` when running on Windows, which means that it can support IPv6 addresses if support is present and configured in the operating system. [#18263]

2. Fixed a bug which caused a "full election" to fail if a majority of sites were not ready when the election started. [#18456]

3. Fixed a bug which could occur when using bulk transfer with Replication Manager. When closing a `DB_ENV` handle, any remaining bulk buffer contents are flushed, and Replication Manager could have tried to send the resulting messages even though its connections had already been closed, leading in rare circumstances to spurious EBADF error reports, or possibly even arbitrary memory corruption. [#18469]

4. Fixed a bug which caused Replication Manager to wait for acknowledgement from client, even if it had failed to send a log record, due to "queue limit exceeded". Replication Manager now returns immediately, with a PERM_FAILED indication, to avoid a pointless delay to the commit() operation. [#18682]
5. Fixed a bug where changes made in one process to Replication Manager configuration values (such as ack policy or ack timeout) were not observed in other processes sharing the same database environment. [#18839]

6. Fixed bugs that could prevent client synchronization from completing due to a failure to request missing log records. [#18849]

7. Fixed a bug where a client that had rolled back transactions to synchronize with a new master, failed to invalidate existing database handles later used for cursor operations based on an explicitly provided transaction. [#18862]

8. Fixed a bug where Replication Manager called for an election after a DUPMASTER event, even when using Master Leases. In such a case it now simply accepts the new (remote) master. [#18864]

9. Fixed a bug which would cause failure if client env attempted to perform sync-up recovery to a point in the log that happened to fall exactly on a log file boundary. [#18907]

Locking Subsystem Changes

1. Moved the wait mutex from the lock structure to the locker structure, reducing the number of mutexes required in the system. [#18685]

Logging Subsystem Changes

1. None.

Memory Pool Subsystem Changes

1. Fixed a race condition that was causing the error: "Unable to allocate space from the buffer cache". The error can only be triggered when multiple memory pool regions are used and there is a periodic gathering and clearing of statistics. This also fixes a second bug where if you compile without statistics and explicitly set the memory pool default pagesize, other environment handles to that environment would not see the correct memory pool default pagesize. [#18386]

2. Fixed a bug where the get_cachesize method and the mpool_stat method returned the initial cache size, even if the cache size had been changed. [#18706]

3. Changed memory pool allocation so that the EIO error is returned rather than the ENOMEM error when the memory cannot be allocated because dirty pages cannot be written. [#18740]

Mutex Subsystem Changes

1. Fixed problems with the printed statistics for DB_MUTEX_SHARED latches. The DB_STAT_CLEAR flag (as specified by db_stat -Z) did not clear the counts of the number of times a shared latch either had to wait to get access or was able to get the latch without waiting. Also, the ownership state of a test-and-set latch (not a hybrid one) was always displayed as not owned, even when it was held. [#17585] [#18743]
Transaction Subsystem Changes

1. Fixed bugs that could caused PANIC or DB_RUNRECOVERY errors when the synchronization of the transaction log failed. [#18588]
2. Fix javadoc to note the exception to the rule that a transaction handle may not be accessed after commit() operaton (getCommitToken() is allowed). [#18730]

Test Suite Changes

1. None. [#18831]

Utility Changes

1. Modified the db_printlog and db_dump -da so that they use the same formatting. The db_logprint utility now uses the message stream. Both db_dump -da and db_printlog accept a -D flag to indicate the numer of bytes of data items to display. You can set this value when calling the DB_ENV->set_data_len method or in the DB_CONFIG. [#18365]
2. Fixed a bug that caused a segmentation violation when using the db_printlog utility. [#18694]
3. Fixed a bug in db_hotbackup that would cause a trap if the -D flag is used and the DB_CONFIG file does not specify a log directory. [#18841]

Configuration, Documentation, Sample Apps, Portability, and Build Changes

1. Added support and documentation for iPhone OS. [#18223]
2. Fix a bug to make the configuration option --enable-debug work when CFLAGS is set. [#18432]
3. Updated Visual Studio project files to enable ADO.NET support. [#18448]
4. Enhanced the source tree layout making it easier to navigate. [#18492]
5. Fixed the --enable-dbm argument to configure. [#18497]
6. Fixed Visual Studio project files so that they can load into Visual Studio 2010. [#18505]
7. Updated Windows CE build files to be consistent with desktop Windows build files. Added Windows Mobile 6.5.3 Professional as a target platform. [#18516]
8. Added a fix so that an error message is displayed when the ’ar’ utility is missing when configured. [#18619]
9. Added tighter integration of JDBC on POSIX/autoconf by including an argument --enable-jdbc to configure. [#18621]
10. Fix build conflicts in log verify with other configurations. [#18658]
11. Upgraded Berkeley DB SQL to SQLite version 3.7.0 [#18857]
Example Changes

1. Renamed examples/c/bench_001 to examples/c/ex_bulk. [#18537]

Miscellaneous Bug Fixes

1. Provided a functionality on the Windows platform to choose a default page size based on the underlying file system sector size. [#16538]

2. Changed DB_NOSYNC from an "operation" constant to a flag value. [#17775]

3. Changed the default permissions for files in the release tree to allow write access. [#17974]

4. Fixed a bug which caused database verification to hang when verifying a database in a Concurrent Data Store environment that performs locking on an environment-wide basis (DB_CDB_ALLDB). [#18571]

Deprecated Features

1. [#18871] Removed the mod_db4 PHP/Apache wrapper. It only supported Apache 1.3 and has not been actively supported. Use php_db4 instead.

Known Bugs

1. None
Chapter 15. Upgrading Berkeley DB 4.8 applications to Berkeley DB 11.2.5.0

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 4.8 release interfaces to the Berkeley DB 11g Release 2 interfaces. (Library version 11.2.5.0). This information does not describe how to upgrade Berkeley DB 1.85 release applications.

`db_sql` Renamed to `db_sql_codegen`

The `db_sql` utility is now called `db_sql_codegen`. This command line utility is not built by default. To build `db_sql_codegen`, specify `--enable-sql_codegen` when configuring Berkeley DB.

`DB_REP_CONF_NOAUTOINIT` Replaced

In this release, the `DB_REP_CONF_NOAUTOINIT` flag is replaced by the `DB_REP_CONF_AUTOINIT` flag. This option is ON by default. To turn off automatic internal initialization, call the `DB_ENV->rep_set_config` method with the `which` parameter set to `DB_REP_CONF_AUTOINIT` and the `onoff` parameter set to zero.

Support for Multiple Client-to-Client Peers

A Berkeley DB Replication Manager application can now designate one or more remote sites (called its "peers") to receive client-to-client requests.

In previous releases, there could be only one peer at a time. If you called the `DB_ENV->repmgr_add_remote_site` method specifying site "A" as a peer and you made another call specifying site "B" as a peer, site "B" would become the only peer, and site "A" would no longer be a peer.

Starting with Berkeley DB 11gR2, the same sequence of calls results in both site "A" and site "B" being possible peers. Replication Manager may select any of a site's possible peers to use for client-to-client requests. If the first peer that the Replication Manager selects cannot be used (for example, it is unavailable or it is the current master), Replication Manager attempts to use a different peer if there is more than one peer.

To get the pre-11gR2 peer behavior in this example, you must make an additional call to the `DB_ENV->repmgr_add_remote_site` method, specifying site "A" and a flag value that excludes the `DB_REPMGR_PEER` bit value to remove site "A" as a possible peer.

Cryptography Support

In this release, the configuration options, `--disable-cryptography` and `--enable-cryptography` are deprecated. `--disable-cryptography` is replaced by `--with-cryptography=no` and `--enable-cryptography` is replaced by `--with-cryptography=yes`.
To build Berkeley DB with support for cryptography, enter `--with-cryptography=yes` as an argument to configure instead of `--enable-cryptography`.

To build Berkeley DB without support for cryptography, enter `--with-cryptography=no` as an argument to configure instead of `--disable-cryptography`.

Berkeley DB now supports encryption using Intel's Performance Primitive (IPP) on Linux. To build Berkeley DB with support for cryptography using Intel's Performance Primitive (IPP) library, enter `--with-cryptography=ipp` as an argument to configure.

Note: The `--with-cryptography=ipp` argument works only on Linux.

**DB_NOSYNC Flag to Flush Files**

Applications must now pass the `DB_NOSYNC` flag to the methods - `DB->remove`, `DB->rename`, `DB_ENV->dbremove`, and `DB_ENV->dbrename`, to avoid a multi-database file to be flushed from cache. This flag is applicable if you have created the database handle in a non-transactional environment.

By default, all non-transactional database remove/rename operations cause data to be synced to disk. This can now be overridden using the `DB_NOSYNC` flag so that files can be accessed outside the environment after the database handles are closed.

**Dropped Support**


Berkeley DB no longer supports Win9X, Windows Me (Millenium edition), and Windows NT 4.0. The minimum supported windows platform is Windows 2000.

**Changing Stack Size**

Prior to the 11gR2 release, Berkeley DB limited the stack size for threads it created using the POSIX thread API to 128 KB for 32-bit platforms and 256 KB for 64-bit platforms. In this release, the system default stack size is used unless you run the Berkeley DB configure script with the `--with-stacksize=SIZE` argument to override the default.

**Berkeley DB 11g Release 2 Change Log**

Changes between 11.2.5.0.26 and 11.2.5.0.32

1. Added Visual Studio 2010 support. Users can find Visual Studio 2010 solutions and projects on `build_windows`. [#18889]
2. Fixed a leak of log file ids when a database is closed before the end of a transaction that references it. [#15957]
3. Fixed a race condition that was causing an "unable to allocate space from the buffer cache" error. The error can only be triggered when multiple mpool regions are used.
and there is a periodic gathering and clearing of statistics. This also fixes a second bug where if you compile without statistics and explicitly set the mpool default pagesize, other environment handles to that environment would not see the correct mpool default pagesize. [#18386]

4. Fix failure to flush pages to disk. [#18760]

5. Fix a general I/O problem on Windows where system doesn't always return ENOENT when file is missing. [#18762]

6. Fixed locking bugs: [#18789]
   - Db->compact of BTREE with MVCC could return an unpinned page.
   - RECNO would fail to lock the next page when splitting a leaf page.

7. Don't await ack if message not sent due to queue limit exceeded. [#18682]

8. Fixed a bug that could cause data to not be returned in a HASH database that was one of multiple databases in a file and it was opened prior to running DB->compact on that database in another thread of control [#18824]

9. Return HANDLE_DEAD on cursor creation that names a specific txn after client callback. [#18862]

10. Remove parting_shot rep_start(CLIENT) in election thread because it can occasionally conflict with rep_start(MASTER) in another thread. [#18946]

11. Fixed a bug that would cause handle locks to be left referencing the wrong metadata page if DB->compact moved the metadata page of a sub-database. [#18944]

12. Fixed a bug that might cause an update to a HASH database to fail with an "unpinned page returned" error if it first gets an I/O error while logging. [#18985]

13. Fixed a bug that failed to dirty a page when DB->compact moved records within a hash bucket [#18994]

14. Fixed a bug in page allocation where if a non-transactional update was being done, then we release the metadata page lock too early possibly leading to the corruption of the in memory page list used by DB->compact. [#19036]

15. A log write failure on a replication master will now cause a panic since the transaction may be committed on some clients. [#19054]

16. Removed the possibility that checkpoints will overlap in the log, decreasing the time to recover [#19062]

17. Fixed a bug that could leave a hash bucket overflow page not linked to the bucket if the unlink of that page aborted. [#19001]

18. Fixed a bug that would leave the next page pointer of a hash bucket that was removed pointing to an invalid page. [#19004]
19. Fixed several bugs that could cause an update running with MVCC to get the wrong version of a page or improperly update the metadata last page number. [#19063]

20. Fixed a bug where an error during an update to a hash database with DB_NOOVERWRITE set could return DB_KEYEXIST rather than the correct error. [#19077]

21. Fixed a bug where an updater supporting DB_READ_UNCOMMITTED might downgrade its lock too soon if there was an error during the update [#19155]

22. Fixed a bug that could cause the wrong page number to be on a root or metadata page if DB->compact moved the page and the operation was later rolled forward [#19167]

23. Fixed a bug that could cause the close of a secondary index database to fail if the transaction doing the open aborted [#19169]

24. The database open code will no longer log the open and close of the master database in a file when opening a sub database in that file [#19071]

**Changes between 11.2.5.0.21 and 11.2.5.0.26**

1. Fixed a bug that might cause recovery to fail if processed part of the log that had previously been recovered and a database which was not present was opened in the log and not closed. [#18459]

2. Fixed a bug which could occur when using bulk transfer with Replication Manager. When closing a DB_ENV handle, any remaining bulk buffer contents are flushed, and Replication Manager could have tried to send the resulting messages even though its connections had already been closed, leading in rare circumstances to spurious EBADF error reports, or possibly even arbitrary memory corruption. [#18469]

3. Fixed a bug in C# HasMultiple() that this function always throws exceptions when there are multiple databases in a single db file. [#18483]

4. Fixed the `--enable-dbm` argument to configure. [#18497]

5. Fixed a bug in the Java API where populating a SecondaryDatabase on open could lead to an OutOfMemoryException. [#18529]

6. Fixed a bug where DB SQL reports "The database disk image is malformed" in "group by" operations. [#18531]

7. Fixed a bug that prevented the same process from reconnecting to the database when DB_REGISTER is being used. [#18535]

8. Fix a race between opening and closing SQL databases from multiple threads that could lead to the error "DB_REGISTER limits processes to one open DB_ENV handle per environment". [#18538]

9. Fixed some bugs that could cause a panic or a DB_RUN_RECOVERY error if the sync of the transaction log failed. [#18588]

10. Fixed a bug which would occur when recovery checkpoint was not written because the cache ran out of space attempting to flush the mpool cache. The environment was
recovered and all database where made available, but some databases were incorrectly closed. This would cause a subsequent recovery to fail on its backward pass with the error 'PANIC: No such file or directory'. [#18590]

11. Fixed a bug that segmentation fault would occur if DB->set_partition_dirs was called before DB->set_partition. [#18591]

12. Fixed a bug that the error of "unknown path" would occur if putting duplicate records to duplicated sorted hash database with DB_OVERWRITE_DUP.[#18607]

13. Fixed a bug where DatabaseConfig.getUnsortedDuplicates() returned true when the database had been configured for sorted duplicates. [#18612]

14. Fixed a bug that could cause recovery to fail with the error "DB_LOGC->get: log record LSN %u/%u: checksum mismatch" if the last log file was nearly full and ended with a partially written log record which was smaller than a checkpoint record. It now erases the invalid partial record before switching to the new log file. [#18651]

15. Initialize DatabaseConfig.pageSize so that it can be queried from Java. [#18691]

16. Fixed a bug that might cause an aborting transaction to fail if it aborted while a DB->compact of the same HASH database was compacting the dynamic hash table [#18695]

### Changes between 4.8 and 11.2.5.0.21

#### Database or Log File On-Disk Format Changes

1. The log file format changed in 11.2.5.0.21

#### New Features

1. Replication Manager sites can specify one or more possible client-to-client peers. [#14776]

2. Added resource management feature in all Berkeley DB APIs to automatically manage cursor and database handles by closing them when they are not required, if they are not yet closed.[#16188]

3. Added a SQL interface to the Berkeley DB library. The interface is based on - and a drop-in-replacement for - the SQLite API. It can be accessed via a command line utility, a C API, or existing APIs built for SQLite. [#16809]

4. Added hash databases support to the DB->compact interface. [#16936]

5. Renamed the “db_sql” utility to “db_sql_codegen”. This utility is not built by default. To build this utility, enter --enable-sql_codegen as an argument to configure. [#18265]

6. Added transactional support in db_sql_codegen utility. Specify TRANSACTIONAL or NONTRANSACTIONAL in hint comments in SQL statement, db_sql_codegen enable/disable transaction in generated code accordingly. [#17237]

7. Added the feature read-your-writes consistency that allows client application to check, or wait for a specific transaction to be replicated from the master before reading database. [#17323]
8. Added DB log verification feature, accessible via the API and a new utility. This feature can help debugging and analysis.  [#17420]

9. Added support for applications to assign master/client role explicitly at any time. Replication Manager can now be configured not to initiate elections.  [#17484]

10. Enhanced the DB->compact method so that it can reassign metadata and root pages from subdatabases to lower numbered pages while compacting a database file that contains multiple databases. This feature helps to free the higher numbered pages and truncate the file.  [#17554]

11. Added system diagnostic messages that are ON by default.  [#17561]

12. Added the feature to assign a priority level to transactions. When resolving a deadlock:
   - if the transactions have differing priority, the lowest priority transaction is aborted
   - if all transactions have the same priority, the same policy that existed before priorities were introduced is used  [#17604]

13. Added a feature in which log_archive uses group-wide information for archiving purposes if Replication Manager is in use. [#17664]

14. Added a feature by which the Replication Manager application clients now automatically request any missing information, even when there is no master transaction activity.  [#17665]

15. Added support for sharing logs across mixed-endian systems.  [#18032]

16. Added an option to specify the first and last pages to the db_dump utility. You can do this by providing -F and -L flags to the db_dump -d option.  [#18072]

17. Added Intel Performance Primitive (IPP) AES encryption support.  [#18110]

18. Removed support for the configuration option --with-mutex=UNIX/fcntl as of version 4.8. If Berkeley DB was configured to use this type of mutex in an earlier release, switch to a different mutex type or contact Oracle for support.  [#18361]

**Database Environment Changes**

1. Fixed a bug to reflect the correct configuration of the logging subsystem when the DB_ENV->log_set_config method is called with the DB_LOG_ZERO flag in a situation where a DB_ENV handle is open and an environment exists.  [#17532]

2. Fixed a bug to prevent memory leak caused when the environment is closed by the named in-memory database in a private database environment which has open named in-memory databases.  [#17816]

3. Fixed a race condition in an internal directory-scanning function that returns the ENOENT ("No such file or directory") error, if a file is removed just before a call to stat() or its equivalent.  [#17850]
## Access Method Changes

1. Fixed a bug to prevent a page in the hash database from carrying the wrong header information when a group allocation is rolled forward by recovery. [#15414]
2. Improved the sort function for bulk put operations. [#17440]
3. Fixed a bug in the DB->compact method to ensure locking of leaf pages when merging higher level interior nodes or when freeing interior nodes when descending to find a non-zero length key. [#17485][#16466]
4. Fixed a bug to prevent a trap if a cursor is opened or closed when another thread is adjusting cursors due to an update in the same database. [#17602]
5. Fixed a bug that incorrectly lead to the error message "library build did not include support for the Hash access method" [#17672]
6. Fixed a bug to ensure that the DB->exists method accepts the DB_AUTO_COMMIT flag. [#17687]
7. In the past, removing a database from a multi-database file that was opened in an environment always caused dirty pages in the file to be flushed from the cache. In this release, there is no implicit flush as part of a DB->remove for handles opened in an environment. Applications that expect the database file to be flushed will need to add an explicit flush. [#17775]
8. Fixed a bug so that the code does not loop if a DB->compact operation processed a 3 or more level non-sorted off page duplicate tree. [#17831]
9. Fixed a bug that could leave pages pinned in the cache if an allocation failed during a DB->compact operation. [#17845]
10. Fixed a bug to ensure sequences are closed when an EntityStore is closed. [#17951]
11. Fixed a bug that prevented retrieval of a non-duplicate record with DB_GET_BOTH_RANGE in hash sorted duplicate db. In a database configured with sorted duplicate support, when the DBcursor->get method is passed the DB_GET_BOTH_RANGE flag, the data item should be retrieved that is the smallest value greater than or equal to the value provided by the data parameter (as determined by the comparison function). [#17997]
12. Fixed a bug that causes the wrong file to be removed if multiple cascading renames are done in the same transaction. [#18069]
13. Fixed a bug to prevent the DB->compact method specified with the DB_AUTO_COMMIT flag from acquiring too many locks. [#18072]
14. Fixed a bug that might cause DB->compact on a DB_BTREE database to get a spurious read lock on the metadata page. If the database was opened non-transactionally the lock would get left behind. [#18257]
15. Fixed a bug that could lead to btree structure corruption if the DB->compact method ran out of locks [#18361]
16. Fixed a bug that would generate an error if a non-BDB file was used to create a database and the DB_TRUNCATE flag was specified. [#18373]

17. Fixed a bug that might cause a trap reading uninitialized memory when backing out a merge of a duplicate tree leaf page during DB->compact. [#18461]

**Locking Subsystem Changes**

1. Fixed a bug to ensure deadlock detection works even when there are blocked transactions, configured with and without timeouts. [#17555]

2. Fixed a bug to ensure a call to the DB->key_range method from outside a transaction does not lock pages. [#17930]

3. Fixed a bug that could cause a segmentation fault if the lock manager ran out of mutexes. [#18428]

**Logging Subsystem Changes**

1. Limited the size of a log record generated by freeing pages from a database, so that it fits in the log file size. [#17313]

**Memory Pool Subsystem Changes**

1. Fixed a bug to ensure multiple versions of a buffer are not created when MVCC is not set. [#17495]

2. Fixed a bug to detect if cache size is being set when the cache is not configured. [#17556]

3. Fixed a bug to ensure the error message “unable to allocate space from the buffer cache” generated when there is still some space available, can be cleared by running recovery. [#17630]

4. Fixed a race condition that causes an operation to return EPERM when the buffer cache is nearly filled with pages belonging to recently closed queue extents. [#17840]

5. Fixed a bug that could cause a page needed by a snapshot reader to be overwritten rather than copied when it was freed. [#17973]

6. Enabled set_mp_pagesize to be specified in the DB_CONFIG file. [#18015]

7. Fixed a bug to ensure single-version or obsolete buffers were selected over any intermediate version. [#18114]

**Mutex Subsystem Changes**

1. Fixed a bug on HP-UX when specifying --with-mutex=HP/msem_init during configure. It would generate the error “TAS: mutex not appropriately aligned” at runtime, when initializing the first mutex. [#17489]

2. Fixed a race condition which could cause unnecessary retrying of btree searches when several threads simultaneously attempted to get a shared latch. [#18078]
3. Exclusive Transactions have been implemented for the SQL API. See the documentation for details on the behavior of this feature. [#17822]

**Tcl-specific API Changes**

1. Fixed a bug in Tcl API to prevent a segmentation fault from occurring when the get dbname method is called to get the db name and the db handle is opened without providing either the filename or dbname. [#18037]

**C#-specific API Changes**

1. Fixed a bug in C# to prevent a System.AccessViolationException from occurring on Windows7 when trying to open new database. [#18422]

2. Fixed a bug in the C# API to make DB_COMPACT consistent with __db_compact in the C API. [#18246]

**API Changes**

1. Added the dbstl_thread_exit method to release thread specific resources on thread exit. [#17595]

2. Fixed the parser to allow configuration API flags set in the DB_CONFIG file to accept an optional ON/OFF string. The DB_REP_CONF_NOAUTOINIT flag has been removed. It is replaced by DB_REP_CONF_AUTOINIT. However, replication’s default behavior remains the same. [#17795]

**Replication Changes**

1. Fixed bug where a not-in-sync client could service a peer request. [#18279]

2. Fixed bug where page gaps, once filled, would not immediately request the next page gap it finds. This was already fixed for logs. [#18219]

3. Fixed a bug so that only one thread waits for the meta-page lock during internal initialization and broadcasts out the information rather than all threads waiting. Removed the former retry code. [#17871]

4. Added a feature by which the DB->open method now allows the DB_CREATE flag on a replication client. It is ignored, but this allows a replication application to make one call that can work on either master or client. It fixes a possible race that could develop in a Replication Manager application if a call to DB->open is made around the same time as a master/client role change. [#15167]

5. The DB_ENV->repmgr_site_list method now returns an indication on whether the site is a client-to-client peer. [#16113]

6. Fixed a bug that could occasionally lead to elections failing to complete. [#17105]

7. Fixed a bug that could cause DB_ENV->txn_stat to trap. [#17198]

8. Added a new JOIN_FAILURE event to notify Replication Manager applications which refuse auto-initialization. [#17319]
9. Fixed a bug where a failed master lease check at a client site causes an ASSERT when processing a master lease grant at the master site. [#17869]

10. Fixed a bug to ensure a second simultaneous call to the DB_ENV->rep_elect method does not incorrectly clear a bit flag. [#17875]

11. Fixed a bug in client-side autoremoval of log files. [#17899]

12. Removed the likelihood of dual data streams to enhance network traffic. [#17955]

13. Fixed a bug such that non-txn dup cursors are accounted for in the replication API lockout. [#18080]

14. Fixed a bug to ensure checking for other sync states for the rerequest thread. [#18126]

15. Fixed a bug to avoid getting stuck in an election forever. [#18151]

16. Fixed a bug where using client-to-client synchronization with Master Leases could have resulted in failure of a new master to get initial lease grants from sufficient number of clients, resulting in a master environment panic. [#18254]

17. Fixed a bug which had prevented Replication Manager socket operations from working on HP/UX systems. [#18382]

18. Fixed a bug where starting as a client in multiple threads after receiving dupmaster messages could have resulted in a failure to find a new log file, resulting in a panic. [#18388]

19. The default thread stack size is no longer overridden by default for Berkeley DB threads. [#18383]

Transaction Subsystem Changes

1. Fixed a bug that caused transactions to deadlock on the mutex in the sequence object. [#17731]

2. Fixed a bug to ensure that the failure checking mechanism reconstructs child transactions correctly when a process dies with active sub-transactions. [#18154]

3. Removed a memory leak during recovery related to a deleted database [#18273]

Utility Changes

1. Fixed compiler warnings in the db_sql_codegen utility. [#17503]

2. Enhanced the db_recover -v utility to display the message, "No log files found", if no logs are present. [#17504]

3. Modified the db_verify utility to verify all files instead of aborting on the first failure. [#17513]

4. Modified the db_verify utility to display a message after verification is completed. [#17545]
5. Fixed a bug in the db_sql_codegen utility where the primary key is stored in both key and data fields. Removed it from the data field. [#17925]

Example Changes

1. Fixed a bug that causes the ex_txn C# example to hang. [#17376]
2. Fixed Solaris alignment issues in Stl port test code. [#17459]
3. Added GCC 4.4 compatibility support for all examples. [#17584]
4. Added new command line arguments (-h and -d) to the env examples. [#17624]
5. Fixed configuration problems related to running java API tests. [#17625]
6. Updated the bench_001 example to include bulk testing examples. [#17766]
7. Added a new Stl example to demo advanced feature usage. The Stl test cases referred earlier are replaced by these new examples in the Stl reference document. [#18175]

Deprecated Features

1. The configuration options --disable-cryptography and --enable-cryptoraphy are being deprecated. [#18110]

Configuration, Documentation, Sample Apps, Portability and Build Changes

1. Remove build files for Windows Visual Studio 6.0. [#16848]
2. Added an API, DBENV->db_full_version, to return db full version.
3. Berkeley DB no longer supports Win9X, Windows Me (Millenium edition) and NT 4.0. The minimum supported windows platform is Win 2k.
5. Added "+u1" to CFLAGS for HP ANSI C Compiler on HP-UX(IA64) to fix the alignment issue found with the allocation functions DB->set-alloc and DB_ENV->set_alloc. [#17257]
6. Fixed a bug such that the thread local storage (TLS) definition modifier is correctly deduced from the m4 script on all platforms. [#17609][#17713]
7. Fixed a bug such that TLS key is not initialized on platforms which do not support thread local storage (TLS) keywords, such as MAC OSX, and where TLS is implemented using pthread API. [#18001]
8. Fixed a bug to ensure that when using Intel C++ compiler (icpc), the TLS code builds successfully. A stricter criteria is adopted to deduce the TLS keyword, and hence pthread API is more likely to be used to implement TLS. [#18038]
9. Adding new configuration option, --with-cryptography={yes|no|ipp}. Using --with-cryptography=yes, will give equivalent behavior to the old --enable-cryptoraphy
option. Using \texttt{--with-cryptography=no}, will give equivalent behavior to the old \texttt{--disable-cryptography} option. Using \texttt{--with-cryptography=ipp} will enable Intel's Performance Primitive (IPP) encryption on Linux. [#18110]

**Known Bugs**

1. The configure option \texttt{--with-uniquename} may cause macro redefinition warnings on platforms where BDB implements parts of the standard C library. These warnings (e.g., \texttt{"db_int_def.h"}, line 586: warning: macro redefined: strsep') may occur when functions in the \texttt{"clib"} directory are included during configuration. This cosmetic affect does not affect the correct operation of the library. [#17172]

2. A multithreaded application using a private environment and multi-version concurrency control could, on very rare occasions, generate an illegal pointer access error during the final steps of a clean environment shutdown. [#17507]

3. Although rare, it is possible for a partial log record header at the end of a transaction log to be erroneously accepted as if it were valid, causing the error "Illegal record type 0 in log" during recovery. [#17851]

4. It is possible to get the error "unable to allocate space from the buffer cache" when there are disk errors on the freezer files used by multi-version concurrency control. [#17902]

5. Java API does not support partitioning by keys and the C# API doesn't support partitioning. [#18350]

6. If a database is removed from an environment and it was still opened transactionally and recovery is run, then a future recovery that must process that part of the log may fail. [#18459]

7. Replication "bulk transfer" does not work if Berkeley DB is unable to determine, at environment open time, whether the Replication Manager will be used. To work around this problem, an application using the Replication Manager should call \texttt{DB_ENV->repmgr_set_local_site()} before opening the environment. An application using the replication Base API should call \texttt{DB_ENV->rep_set_transport()} before opening the environment. [#18476]

8. The BTree prefix comparison function behaves slightly differently in the C API vs the C# API. In the C# API it returns a signed int and in the C API it returns an unsigned int. This can be a problem if the application needs to save more than $2^{31}$ bytes.
Chapter 16. Upgrading Berkeley DB 4.7 applications to Berkeley DB 4.8

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 4.7 release interfaces to the Berkeley DB 4.8 release interfaces. This information does not describe how to upgrade Berkeley DB 1.85 release applications.

Registering DPL Secondary Keys

Entity subclasses that define secondary keys must now be registered prior to storing an instance of the class. This can be done in two ways:

- The EntityModel.registerClass() method may be called to register the subclass before opening the entity store.
- The EntityStore.getSubclassIndex() method may be called to implicitly register the subclass after opening the entity store.

Failure to register the entity subclass will result in an IllegalArgumentException the first time an attempt is made to store an instance of the subclass. An exception will not occur if instances of the subclass have previously been stored, which allows existing applications to run unmodified in most cases.

This behavioral change was made to increase reliability. In several cases, registering an entity subclass has been necessary as a workaround. The requirement to register the subclass will ensure that such errors do not occur in deployed applications.

Minor Change in Behavior of DB_MPOOLFILE->get

DB 4.8 introduces some performance enhancements, based on the use of shared/exclusive latches instead of locks in some areas of the internal buffer management code. This change will affect how the DB_MPOOL interface handles dirty buffers.

Because of these changes, DB_MPOOLFILE->get will now acquire an exclusive latch on the buffer if the DB_MPOOL_DIRTY or DB_MPOOL_EDIT flags are specified. This could lead to an application deadlock if the application tries to fetch the buffer again, without an intervening DB_MPOOLFILE->put call.

If your application uses the DB_MPOOL interface, and especially the DB_MPOOL_DIRTY and DB_MPOOL_EDIT flags, you should review your code to ensure that this behavior change does not cause your application to deadlock.

Dropped Support for fcntl System Calls

Berkeley DB no longer supports mutex implementations based on the fcntl system call. If you have been configuring Berkeley DB to use this type of mutex, you need to either switch to a different mutex type or contact the Berkeley DB team for support.
Upgrade Requirements

The log file format changed in the Berkeley DB 4.8 release.

No database formats changed in the Berkeley DB 4.8 release.

The Berkeley DB 4.8 release does not support live replication upgrade from the 4.2 or 4.3 releases, only from the 4.4 and later releases.

For further information on upgrading Berkeley DB installations, see the Berkeley DB Upgrade Guide.

Berkeley DB 4.8.28 Change Log

Changes between 4.8.26 and 4.8.28:

1. Limit the size of a log record generated by freeing pages from a database so it fits in the log file size. [#17313]
2. Fix a bug that could cause a file to be removed if it was both the source and target of two renames within a transaction. [#18069]
3. Modified how we go about selecting a usable buffer in the cache. Place more emphasis on single version and obsolete buffers. [#18114]

Known bugs in 4.8

1. Sharing logs across mixed-endian systems does not work. [#18032]

Changes between 4.8.24 and 4.8.26:

1. Fixed a bug where the truncate log record could be too large when freeing too many pages during a compact. [#17313]
2. Fixed a bug where the deadlock detector might not run properly. [#17555]
3. Fixed three bugs related to properly detecting thread local storage for DbStl. [#17609] [#18001] [#18038]
4. Fixed a bug that prevented some of our example code from running correctly in a Windows environment. [#17627]
5. Fixed a bug where a “unable to allocate space from buffer cache” error was improperly generated. [#17630]
6. Fixed a bug where DB->exists() did not accept the DB_AUTO_COMMIT flag. [#17687]
7. Fixed a bug where DB_TXN_SNAPSHOT was not getting ignored when DB_MULTIVERSION not set. [#17706]
8. Fixed a bug that prevented callback based partitioning through the Java API. [#17735]
9. Fixed a replication bug where log files were not automatically removed from the client side. [#17899]

10. Fixed a bug where code generated from db_sql stored the key in both the data and key DBTs. [#17925]

11. Fixed a bug that prevented a sequence from closing properly after the EntityStore closed. [#17951]

12. Fixed a bug where gets fail if the DB_GET_BOTH_FLAG is specified in a hash, sorted duplicates database.[#17997]

Changes between 4.8.21 and 4.8.24:

1. Fixed a bug in the C# API where applications in a 64-bit environment could hang. [#17461]

2. Fixed a bug in MVCC where an exclusive latch was not removed when we couldn't obtain a buffer. [#17479]

3. Fixed a bug where a lock wasn't removed on a non-transactional locker. [#17509]

4. Fixed a bug which could trigger an assertion when performing a B-tree page split and running out of log space or with MVCC enabled. [#17531]

5. Fixed a bug in the repquote example that could cause the application to crash. [#17547]

6. Fixed a couple of bugs when using the GCC 4.4 compiler to build the examples and the dbstl API. [#17504] [#17476]

7. Fixed an incorrect representation of log system configuration info. [#17532]

Changes between 4.7 and 4.8.21:

Database or Log File On-Disk Format Changes:

1. The log file format changed in 4.8.

New Features:

1. Improved scalability and throughput when using BTree databases especially when running with multiple threads that equal or exceed the number of available CPUs.

2. Berkeley DB has added support for C#. In addition to the new C# api, C# specific tests and sample applications were also added. [#16137]

3. Berkeley DB has added an STL API, which is compatible with and very similar to C++ Standard Template Library (STL). Tests and sample applications and documentation were also added. [#16217]

4. Berkeley DB has added database partitioning. BTree or Hash databases may now be partitioned across multiple directories. Partitioned databases can be used to increase
concurrency and to improve performance by spreading access across disk subsystems. [#15862]

5. Berkeley DB now supports bulk insertion and deletion of data. Similar to the bulk get interface, the bulk put and bulk delete allow the developer to populate a buffer of key-value pairs and then pass it to the BDB library with a single API call.


7. Berkeley DB introduces a new utility named db_sql which replaces db_codegen. Similar to db_codegen, db_sql accepts an input file with DDL statements and generates a Berkeley DB application using the C API that creates and performs CRUD operations on the defined tables. The developer can then use that code as a basis for further application development.

8. The Replication Manager now supports shared access to the Master database environment from multiple processes. In earlier versions, multiple process support on the Master required use of the Base Replication API. [#15982]

9. Foreign Key Support has been added to Berkeley DB.

10. Several enhancements were made to DB_REGISTER & DB_ENV->failchk().

11. Berkeley now supports 100% in-memory replication.

12. Berkeley DB now has the ability to compare two cursors for equality. [#16811]

Database Environment Changes:

1. Fixed a bug that could cause an allocation error while trying to allocate thread tracking information for the DB_ENV->failcheck system. [#16300]

2. Fixed a bug that could cause a trap if an environment open failed and failchk thread tracking was enabled. [#16770]

Concurrent Data Store Changes:

None.

General Access Method Changes:

1. Fixed a bug where doing an insert with secondary indices and the NOOVERWRITE flag could corrupt the secondary index. [#15912]

2. Fixed a possible file handle leak that occurred while aborting the create of a database whose metadata page was not initialized. [#16359]

3. Fixed a bug so that we now realloc the filename buffer only if we need it to grow. [#16385] [#16219]

4. Fixed a race freeing a transaction object when using MVCC. [#16381]
5. Added missing get methods for the DB and DB_ENV classes where there already was a corresponding set method. [#16505]

6. Fixed a bug to now ensure that DB_STAT_SUBSYSTEM is distinct from other stat flags. [#16798]

7. Fixed a bug related to updating multiple secondary keys (using DB_MULTIPLE). [#16885]

8. Fixed a bug so that verify (db->verify, db_verify) will now report when it cannot read a page rather than just saying the database is bad. [#16916]

9. Fixed a bug that could cause memory corruption if a transaction allocating a page aborted while DB->compact was running on that database. [#16862]

10. Fixed a bug where logging was occurring during remove of an in-memory database when the DB_TXN_NOT_DURABLE flag was set. [#16571]

11. Fixed a bug to remove a race condition during database/file create. [#17020]

12. Fixed a bug where a call to DB->verify and specifying DB_SALVAGE could leak memory when the call returned. [#17161]

13. Fixed a bug to avoid accessing freed memory during puts on primaries with custom comparators. [#17189]

14. Fixed a bug that could cause old versions of pages to be written over new versions if an existing database is opened with the DB_TRUNCATE flag. [#17191]

**Btree Access Method Changes:**

1. Fixed a bug which could cause DB->compact to fail with DB_NOTFOUND or DB_PAGE_NOTFOUND if the height of the tree was reduced by another thread while compact was active. The bug could also cause a page split to trigger splitting of internal nodes which did not need to be split. [#16192]

2. Fixed a bug that caused Db->compact to loop if run on an empty RECNO database when there were pages in the free list. [#16778]

3. Added a new flag, DB_OVERWRITE_DUP, to DB->put and DBC->put. This flag is equivalent to DB_KEYLAST in almost all cases: the exception is that with sorted duplicates, if a matching key/data pair exists, we overwrite it rather than returning DB_KEYEXIST. [#16803]

**Hash Access Method Changes:**

1. Fixed a bug to now force a group allocation that rolls forward to reinit all the pages. Otherwise a previous aborted allocation may change the header. [#15414]

2. Fixed a bug to now return the expected buffer size on a DB_BUFFER_SMALL condition. [#16881]
Queue Access Method Changes:

1. Fixed a bug that would cause the LSN reset functionality to not process queue extents. [#16213]

2. Fixed a bug that prevented a partial put on a queue database with secondaries configured. [#16460]

3. Fixed a bug to now prevent an unpinned page to be returned if a delete from a HASH database deadlocked. [#16371]

4. Fixed a bug that could cause a queue extent to be recreated if an application deleted a record that was already deleted in that extent. [#17004]

5. Added the DB_CONSUME flag to DB->del and DBC->del to force adjustment of the head of the queue. [#17004]

Recno Access Method Changes:

1. Fixed a bug which could cause DB->compact of a RECNO database to loop if the number of pages on the free list was reduced by another thread while compact was active. [#16199]

2. Fixed a bug that occurs when deleting from a Recno database and using DB_READ_UNCOMMITTED where we could try to downgrade a lock twice. [#16347]

3. Fixed a bug to now disallow passing DB_DUP and DB_RECNUM together to __db_set_flags. [#16585]

C-specific API Changes:

1. Add get functions for each set functions of DB and DB_ENV structures which didn’t have one. [#16505]

C++-specific API Changes:

1. The get and set lk_partitions methods are now available.

2. Add get functions for each set functions of Db and DbEnv classes which didn’t have one. [#16505]

3. Fixed a memory leak when using nested transactions. [#16956]

Java-specific API Changes:

1. Fixed a bug where the replication finer-grained verbose flags were not available in the Java API. [#15419]

2. Fixed a bug in the BTree prefix compression API when called from the Java API. DBTs were not properly initialized. [#16417]

3. Fixed a bug so that LogCursor will work correctly from the Java API. [#16827]
4. Fixed a bug so that position(), limit() and capacity() of ByteBuffers are obeyed by DatabaseEntry objects. [#16982]

**Direct Persistence Layer (DPL), Bindings and Collections API:**

1. The StoredMap class now implements the standard java.util.concurrent.ConcurrentMap interface. [#15382]

2. Report a meaningful IllegalArgumentException when @Persistent is incorrectly declared on an enum class. Before, the confusing message Persistent class has non-persistent superclass: java.lang.Enum was reported. [#15623]

3. Report a meaningful IllegalArgumentException when @Persistent is incorrectly declared on an interface. Before, a NullPointerException was reported. [#15841]

4. Several validation checks have been added or corrected having to do with entity subclasses, which are @Persistent classes that extend an @Entity class. [#16077]

5. Optimized marshaling for large numbers of embedded objects improving performance. [#16198]

6. The StoredMap class now implements the Java 1.5 ConcurrentMap interface. [#16218]

7. Fix a DPL bug that caused exceptions when using a class Converter for an instance containing non-simple fields. [#16233]

8. Add EntityCursor.setCacheMode and getCacheMode. See the com.sleepycat.je.CacheMode class for more information. [#16239]

9. Fix a bug that prevents evolution of @SecondaryKey information in an entity subclass (a class that extends an @Entity class). [#16253]

10. Report a meaningful IllegalArgumentException when @Persistent or @Entity is incorrectly used on an inner class (a non-static nested class). Before, the confusing message No default constructor was reported. [#16279]

11. Improved the reliability of Entity subclasses that define secondary keys by requiring that they be registered prior to storing an instance of the class. [#16399]

12. Fix a bug that under certain circumstances causes "IllegalArgumentException: Not a key class" when calling EntityStore.getSubclassIndex, EntityStore.getPrimaryConfig, EntityStore.getSecondaryConfig, or PrimaryIndex.put, and a composite key class is used. [#16407]

13. Fixed a bug so that one can now compile DPL in the Java API on Windows. [#16570]

14. The com.sleepycat.collections.TransactionRunner.handleException method has been added to allow overriding the default transaction retry policy. See the javadoc for this method for more information. [#16574]

15. Fix a bug that causes an assertion to fire or a NullPointerException (when assertions are disabled) from the EntityStore constructor. The problem occurs only when the previously
created EntityStore contains an entity with a secondary key definition in which the key name has been overridden and is different than the field name. [#16819]

16. Key cursors have been optimized to significantly reduce I/O when the READ_UNCOMMITTED isolation mode is used. See EntityIndex.keys for more information. [#16859]

17. Report a meaningful IllegalArgumentException when NULLIFY is used with a @SecondaryKey and the field is a primitive type. Before, the confusing message Key field object may not be null was reported. [#17011]

18. Enum fields may now be used as DPL keys, including primary keys, secondary keys, and fields of composite key classes. Comparators are supported for composite key classes containing enum fields. [#17140]

19. Fix a bug that prevented the use of custom key comparisons (composite key classes that implement Comparable) for secondary keys defined as ONE_TO_MANY or MANY_TO_MANY. [#17207]

20. The db.jar file now contains a Premain class which enables bytecode enhancement using the JVM instrumentation commands. The built-in proxy classes are also now enhanced in the db.jar file, which enables off-line bytecode enhancement. For more information on DPL bytecode enhancement and how to use both instrumentation and off-line enhancement, please see the com.sleepycat.persist.model.ClassEnhancer javadoc. [#17233]

Tcl-specific API Changes:

1. The mutex API is now available when using Tcl. [#16342]

RPC-specific Client/Server Changes:

- RPC support has been removed from Berkeley DB. [#16785]

Replication Changes:

1. Improved testing of initial conditions for rep and repmgr APIs and added heartbeat timeouts to rep_get_timeout. [#14977]

2. Added DB_REP_CONF_INMEM replication configuration flag to store replication information exclusively in-memory without creating any files on-disk. [#15257]

3. Added repmgr support for multi-process shared env [#15982]

4. Fixed a bug where opening a cursor from a database handle failed to check whether the database handle was still fresh. If the database handle had been invalidated by a replication client synchronizing with a new master, it could point to invalid information. [#15990]

5. Fixed a bug so that if LOG_REQ gets an archived LSN, replication sends VERIFY_FAIL. [#16004]
6. Added timestamp and process/thread id to replication verbose messages. [#16098]

7. Fixed a bug where, in very rare circumstances, two repmgr sites could connect to each other at the exact same time, the connection attempts “collide” and fail, and the same collision repeats in time synchronization indefinitely. [#16114]

8. Fixed a bug where a missing database file (FILE_FAIL error condition) can interrupt a client synchronization without restarting it. [#16130]

9. Fixed a bug by adding REP_F_INREPSSTART flag to prevent racing threads in rep_start. [#16247]

10. Fixed a bug to not return HOLDELECTION if we are already in the middle of an election. Updated the egen so the election thread will notice. [#16270]

11. Fixed a bug in buffer space computation, which could have led to memory corruption in rare circumstances, when using bulk transfer. [#16357]

12. Fixed a bug that prevented replication clients from opening a sequence. The sequence is opened for read operations only. [#16406]

13. Fixed a bug by removing an assertion about priority in elections. It is not correct because it could have changed by then. Remove unused recover_gen field. [#16412]

14. Fixed a bug to now ignore a message from client if it is an LSN not recognized in a LOG_REQ. [#16444]

15. Fixed a bug so that on POSIX systems, repmgr no longer restores default SIGPIPE action upon env close, if it was necessary to change it during start-up. This allows remaining repmgr environments within the same process, if any, to continue operating after one of them is closed. [#16454]

16. After a replication client restarts with recovery, any named in-memory databases are now re-materialized from the rest of the replication group upon synchronization with the master. [#16495]

17. Fixed a bug by adding missing rep_get_config flags. [#16527]

18. Instead of sleeping if the bulk buffer is in transmission, return so that we can send as a singleton. [#16537]

19. Fixed a bug by changing __env_refresh to not hit assert on -private -rep env with an in-memory database. [#16546]

20. Fixed a bug in the Windows implementation of repmgr where a large number of commit threads concurrently awaiting acknowledgments could result in memory corruption, and leaking Win32 Event Objects. [#16548]

21. Fixed a bug by changing repmgr to count a dropped connection when noticing a lacking heartbeat; fixed heartbeat test to check for election, rather than connection drop count, and more reasonable time limit; fixed test to poll until desired result, rather than always sleeping max possible time. [#16550]
22. Fixed "master changes" stat to count when local site becomes master too. [#16562]

23. Fixed a bug where a c2c client would send UPDATE_REQ to another client [#16592]

24. Removed code to proactively expire leases when we don't get acks. Leases maintain their own LSNs to know. [#16494]

25. Fixed a bug where a client may not sync pages during internal init. [#16671]

26. Fixed a bug where a client that received and skipped a log record from the master during an election, then won the election, could then try to request a copy of the skipped log record. The result was an attempt to send a request to the local site, which is invalid: this could confuse a replication Base API application, or cause the Replication Manager to crash. [#16700]

27. Fixed a bug which could have caused data loss or corruption (at the client only) if a replication client rolled back existing transactions in order to synchronize with a new master, and then crashed/recovered before a subsequent checkpoint operation had been replicated from the master. [#16732]

28. Fixed a bug so that replication now retries on DB_LOCK_NOTGRANTED. [#16741]

29. Fixed a potential deadlock in rep_verify_fail. [#16779]

30. Fixed a bug so that an application will no longer segv if nsites given was smaller than number of sites that actually exists. [#16825]

**XA Resource Manager Changes:**

1. The XA Resource Manager has been removed from Berkeley DB. [#6459]

**Locking Subsystem Changes:**

1. Fixed a bug to prevent unlocking a mutex twice if we ran out of transactional locks. [#16285]

2. Fixed a bug to prevent a segmentation trap in __lock_open if there were an error during the opening of an environment. [#16307]

3. Fixed a bug to now avoid a deadlock if user defined locks are used only one lock partition is defined. [#16415]

4. Fixed concurrency problems in __dd_build, __dd_abort by adding LOCK_SYSTEM_LOCK() calls to __dd_build and __dd_abort. [16489]

5. Fixed a bug that could cause a panic if a transaction which updated a database that was supporting READ_UNCOMMITTED readers aborted and it hit a race with a thread running the deadlock detector. [#16490]

6. Fixed a race condition in deadlock detection that could overwrite heap. [#16541]

7. Fixed a bug so that DB_STAT_CLEAR now restores the value of st_partitions. [#16701]
Logging Subsystem Changes:

1. Fixed a bug so that the header checksum is only ignored when the log is from a previous version [#16281]
2. Fixed a bug by removing a possible race condition with logc_get(DB_FIRST) and log archiving. [#16387]
3. Fixed a bug that could cause a recovery failure of a create of a database that was aborted. [#16824]
4. An in-memory database creation has an intermediate phase where we have a semi-open DBP. If we crash in that state, then recovery was failing because it tried to use a partially open database handle. This fix checks for that case, and avoids trying to undo page writes for databases in that interim step. [#17203]

Memory Pool Subsystem Changes:

1. Fixed a bug that occurred after all open handles on a file are closed. Needed to clear the TXN_NOT_DURABLE flag (if set) and mark the file as DURABLE_UNKNOWN in the memory pool. [#16091]
2. Fixed a possible race condition between dirtying and freeing a buffer that could result in a panic or corruption. [#16530]
3. Fixed a memory leak where allocated space for temporary file names are not released. [#16956]

Mutex Subsystem Changes:

1. Fixed a bug when using mutexes for SMP MIPS/Linux systems. [#15914]
2. POSIX mutexes are now the default on Solaris. [#16066]
3. Fixed a bug in mutex allocation with multiple cache regions. [#16178]
4. Fixed MIPS/Linux mutexes in 4.7. [#16209]
5. Fixed a bug that would cause a mutex to be unlocked a second time if we ran out of space while tracking pinned pages. [#16228]
6. Fixed a bug Sparc/GCC when using test-and-set mutexes. They are now aligned on an 8-byte boundary. [#16243]
7. Fixed a bug to now prevent a thread calling DB_ENV->failcheck to hang on a mutex held by a dead thread. [#16446]
8. Fixed a bug so that __db_pthread_mutex_unlock() now handles the failchk case of finding a busy mutex which was owned by a now-dead process. [#16557]
9. Removed support for the mutex implementation based on the “fcntl” system call. Anyone configuring Berkeley DB to use this type of mutex in an earlier release will need to either switch to a different mutex type or contact Oracle for support. [#17470]
Test Suite Changes

1. Fixed a bug when using failchk(), where a mutex was not released. [#15982]
2. Added a set of basic repmgr tests to run_std and run_all. [#16092]
3. Added control wrapper for db_reptest to test suite. [#16161]
4. Fixed a bug to now skip tests if db_reptest is not configured. [#16161]
5. Changed name of run_db_in_mem to run_inmem_db, and run_inmem to run_inmem_log and made the arg orders consistent. [#16358]
6. Fixed a bug to now clean up stray handles when rep_verify doesn't work. [#16390]
7. Fixed a bug to avoid db_reptest passing the wrong flag to repmgr_start when there is already a master. [#16475]
8. Added new tests for abbreviated internal init. Fixed test not to expect in-memory database to survive recovery. [#16495]
9. Fix a bug, to add page size for txn014 if the default page size is too small. Move files instead of renaming directory for env015 on QNX. [#16627]
10. Added new rep088 test for log truncation integrity. [#16732]
11. Fixed a bug by adding a checkpoint in rep061 to make sure we have messages to process. Otherwise we could hang with client stuck in internal init, and no incoming messages to trigger rerequest. [#16781]

Transaction Subsystem Changes:

1. Fixed a bug to no longer generate an error if DB_ENV->set_flags (DB_TXN_NOSYNC) was called after the environment was opened. [#16492]
2. Fixed a bug to remove a potential hang condition in replication os_yield loops when DB_REGISTER used with replication by adding PANIC_CHECKS. [#16502]
3. Fix a bug to now release mutex obtained before special condition returns in __db_cursor_int and __txn_record_fname. [#16665]
4. Fixed a leak in the transaction region when a snapshot update transaction accesses more than 4 databases. [#16734]
5. Enabled setting of set_thread_count via the DB_CONFIG file. [#16878]
6. Fixed a mutex leak in some corner cases. [#16665]

Utility Changes:

1. The db_stat utility with the -RA flags will now print a list of known remote replication flags when using repmgr. [#15484]
2. Restructured DB salvage to walk known leaf pages prior to looping over all db pages. [#16219]

3. Fixed a problem with upgrades to 4.7 on big endian machines. [#16411]

4. Fixed a bug so that now db_load consistently returns >1 on failure. [#16765]

5. The db_dump utility now accepts a “-m” flag to dump information from a named in-memory database. [#16896]

6. Fixed a bug that would cause db_hotbackup to fail if a database file was removed while it was running. [#17234]

Configuration, Documentation, Sample Application, Portability and Build Changes:

1. Fixed a bug to now use the correct Perl include path. [#16058]

2. Updated the version of the Microsoft runtime libraries shipped. [#16058]

3. Upgraded the Visual Studio build files to be based on Visual Studio 8 (2005+). The build is now simplified. Users can still upgrade the Visual Studio 6.0 project files, if they want to use Visual Studio .NET (7.1) [#16108]

4. Expanded the ex_rep example with checkpoint and log archive threads, deadlock detection, new options for acknowledgment policy and bulk transfer, and use of additional replication features and events. [#16109]

5. Fixed a bug so that optimizations on AIX are re-enabled, avoiding incorrect code generation. [#16141]

6. Removed a few compiler warnings and three type redefinitions when using vxworks and the GNU compiler. [#16341]

7. Fixed a bug on Sparc v9 so that MUTEX_MEMBAR() now uses membar_enter() to get a #storeload barrier rather than just stbar’s #storestor. [#16468]


9. Fixed lock_get and lock_vec examples from the Java (and C#) API. Updated the Java lock example. [#16506]

10. Fixed a bug to correctly handle the TPC-B history record on 64-bit systems. [#16709]

11. Add STL API to Linux build. Can be enabled via the --enable-stl flag. [#16786]

12. Add STL API to Windows build, by building the db_stl project in the solution. There are also stl’s test and examples projects in this solution. [#16786]

13. Add support to build dll projects for WinCE, in order to enable users to build DB into a dll in addition to a static library.[#16625]
14. Fixed a weakness where several malloc/realloc return values are not checked before use. [#16664]

15. Enabled DB->compact for WinCE.[#15897]

16. HP-UX 10 is no longer supported.
Chapter 17. Test Suite

Running the test suite

Once you have started tclsh and have loaded the test.tcl source file (see Running the test suite under UNIX (page 43) and Running the test suite under Windows (page 26) for more information), you are ready to run the test suite. At the tclsh prompt, to run the standard test suite, enter the following:

```bash
% run_std
```

A more exhaustive version of the test suite runs all the tests several more times, testing encryption, replication, and different page sizes. After you have a clean run for run_std, you may choose to run this lengthier set of tests. At the tclsh prompt, enter:

```bash
% run_all
```

Running the standard tests can take from several hours to a few days to complete, depending on your hardware, and running all the tests will take at least twice as long. For this reason, the output from these commands are redirected to a file in the current directory named ALL.OUT. Periodically, a line will be written to the standard output, indicating what test is being run. When the test suite has finished, a final message will be written indicating the test suite has completed successfully or that it has failed. If the run failed, you should review the ALL.OUT file to determine which tests failed. Errors will appear in that file as output lines, beginning with the string "FAIL".

Tests are run in the directory TESTDIR, by default. However, the test files are often large, and you should use a filesystem with at least several hundred megabytes of free space. To use a different directory for the test directory, edit the file include.tcl in your build directory, and change the following line to a more appropriate value for your system:

```tcl
set testdir ./TESTDIR
```

For example, you might change it to the following:

```tcl
set testdir /var/tmp/db.test
```

Alternatively, you can create a symbolic link named TESTDIR in your build directory to an appropriate location for running the tests. Regardless of where you run the tests, the TESTDIR directory should be on a local filesystem. Using a remote filesystem (for example, an NFS mounted filesystem) will almost certainly cause spurious test failures.

Test suite FAQ

1. The test suite has been running for over a day. What's wrong?

   The test suite can take anywhere from some number of hours to several days to run, depending on your hardware configuration. As long as the run is making forward progress and new lines are being written to the ALL.OUT files, everything is probably fine.