Empress is designed for use with Linux, UNIX, FreeBSD, Mac OS, Windows, QNX, VxWorks and other Real-Time operating systems. The availability of some features is operating system dependent.


All other product names contained in this book are trademarks of their respective owners.

© Copyright Empress Software Inc. 2003, 2009

All rights reserved. Reproduction of this document in whole or part, by electronic or any other means, is prohibited without written consent from Empress Software Incorporated.

Information in this document is subject to change without notice and does not represent a commitment on the part of Empress Software Inc. Empress Software Inc. assumes no responsibility for any errors that may appear in this document.

Not all features are available on all operating system and hardware platforms.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph © (1) (ii) of the Rights in Technical Data and Computer Software clause at 52.227-7013.

02/2009

Part #: 011-2009-217
The Empress Product Profile at a glance:

<table>
<thead>
<tr>
<th></th>
<th>The Empress Product Profile at a glance:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What is Empress Ultra Embedded Database?</td>
</tr>
<tr>
<td>2</td>
<td>What is Empress Software Inc?</td>
</tr>
<tr>
<td>3</td>
<td>Why Use Empress?</td>
</tr>
<tr>
<td>4</td>
<td>How to Use Empress?</td>
</tr>
<tr>
<td>5</td>
<td>Where is Empress Being Used?</td>
</tr>
<tr>
<td>6</td>
<td>What are Empress’ Major Features and Benefits?</td>
</tr>
<tr>
<td>7</td>
<td>What Does Empress Run on?</td>
</tr>
<tr>
<td>8</td>
<td>Empress Maximum Sizes</td>
</tr>
<tr>
<td>9</td>
<td>What are the Empress Products?</td>
</tr>
<tr>
<td>10</td>
<td>What is in the Empress Ultra Embedded Package?</td>
</tr>
<tr>
<td>11</td>
<td>Empress’ Features by Module</td>
</tr>
<tr>
<td>12</td>
<td>Where Can I Get Empress?</td>
</tr>
<tr>
<td></td>
<td>Appendix 1 A List of MR Routines</td>
</tr>
<tr>
<td></td>
<td>Appendix 2 A List of Empress System Variables</td>
</tr>
<tr>
<td></td>
<td>Appendix 3 White Paper: The Concept of I=MC²</td>
</tr>
<tr>
<td></td>
<td>Appendix 4 White Paper: Securing Data a Rest</td>
</tr>
<tr>
<td></td>
<td>Index</td>
</tr>
</tbody>
</table>
# Table of Contents

1. **What is Empress Ultra Embedded Database?** .................. 9

2. **What is Empress Software Inc?** ................................. 13

3. **Why Use Empress?** .......................................................... 15
   - It’s Powerful .............................................................................. 17
   - It’s Fast .................................................................................. 18
   - It’s Reliable ........................................................................... 19
   - It’s Cost Effective .................................................................. 20

4. **How to Use Empress?** .......................................................... 21
   - Stand-alone Scenarios .......................................................... 23
   - Scenarios using Empress Connectivity Server for Client/Server ......................................................................... 27
   - Performance Comparison among Empress Interfaces .................................................................................. 30
   - Replication Scenarios .......................................................... 31

5. **Where is Empress Being Used?** ......................................... 35

6. **What are Empress’ Major Features and Benefits?** .......... 43
   - Application Development ................................................... 45
   - Embeddable ........................................................................... 45
   - Deterministic Response ......................................................... 45
   - Fast Performance .................................................................... 45
   - Flexible Development ............................................................ 46
   - Friendly Storage Requirements ........................................... 46
   - Small Footprint ...................................................................... 46
   - Wide Range of Platforms ......................................................... 46
   - Cost Effective Partnership Programs .................................. 46
   - Advanced Product Distribution ........................................... 47
   - Innovative Solutions ............................................................... 47
   - Production Deployment ........................................................ 48
   - Significant Cost Savings ........................................................ 48
   - High Availability .................................................................. 48
   - Deployment Flexibility ............................................................ 48
   - Scalability ............................................................................. 48
   - In-Depth Database Expertise .................................................. 49
   - Easy Access to User Data ......................................................... 49
   - High Performance ................................................................. 49
7 What Does Empress Run on? ................................................................. 51
   Supported Platforms .............................................................................. 53

8 Empress Maximum Sizes .................................................................... 55
   Operational Parameters ........................................................................ 57

9 What are the Empress Products? .......................................................... 59
   9.1 The Empress Software Products ......................................................... 60
   9.2 Support and Upgrades ........................................................................ 63
      Support by phone, fax and e-mail ......................................................... 63
      Upgrades and Updates ......................................................................... 63
   9.3 Technical Services ............................................................................. 64
      Installation ............................................................................................ 64
      Consulting ............................................................................................ 64
      Source Level Customization ................................................................. 64
      Training ............................................................................................... 64

10 What is in the Empress Ultra Embedded Package? ...................... 67
   10.1 The Empress Ultra Embedded Package ............................................ 68
   10.2 The Empress CD .............................................................................. 69
   10.3 The Empress Installation Key ............................................................ 70

11 Empress’ Features by Module .............................................................. 71
   11.1 Empress RDBMS Engine ................................................................. 72
      Product Functionality ........................................................................... 73
      Supported Data Types ........................................................................ 75
         Character Data .................................................................................. 78
         Text Data .......................................................................................... 78
         National Language Support (NLS) Character Data ......................... 78
         National Language Support (NLS) Text Data .................................. 78
         Multimedia Data ................................................................................ 78
         Date Data .......................................................................................... 79
         Time Data .......................................................................................... 79
         MicroSecond Timestamp Data ........................................................ 79
         Decimal Data ..................................................................................... 80
         Dollar Data ......................................................................................... 80
         Floating Point Data ........................................................................... 81
         Integer Data ....................................................................................... 81
   11.2 Empress Servers ............................................................................... 82
      Connectivity Server (ODBC, JDBC) ..................................................... 83
      Replication Server ................................................................................ 84
      Stand-alone Mode ................................................................................ 85
   11.3 Empress APIs .................................................................................. 86
      C-API MR Routines .............................................................................. 87
      C-API Embedded SQL ........................................................................ 88
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-API MSCALL</td>
<td>89</td>
</tr>
<tr>
<td>ODBC Interface</td>
<td>90</td>
</tr>
<tr>
<td>JDBC Interface</td>
<td>91</td>
</tr>
<tr>
<td>C++ Interface</td>
<td>92</td>
</tr>
<tr>
<td>11.4 Empress Utilities</td>
<td>93</td>
</tr>
<tr>
<td>Empress System Variables</td>
<td>94</td>
</tr>
<tr>
<td>Empress Database Administrative Variables</td>
<td>95</td>
</tr>
<tr>
<td>Database Administration</td>
<td>96</td>
</tr>
<tr>
<td>Database Integrity Checking and Maintenance</td>
<td>97</td>
</tr>
<tr>
<td>Database Backup and Recovery</td>
<td>98</td>
</tr>
<tr>
<td>Database Version Upgrade</td>
<td>99</td>
</tr>
<tr>
<td>Data Import and Export</td>
<td>100</td>
</tr>
<tr>
<td>Empress Monitors and Statistics</td>
<td>101</td>
</tr>
<tr>
<td>Shared Memory</td>
<td>102</td>
</tr>
<tr>
<td>Batch SQL</td>
<td>103</td>
</tr>
<tr>
<td>12 Where Can I Get Empress?</td>
<td>105</td>
</tr>
<tr>
<td>North America</td>
<td>106</td>
</tr>
<tr>
<td>Europe</td>
<td>106</td>
</tr>
<tr>
<td>Asia</td>
<td>108</td>
</tr>
<tr>
<td>South America</td>
<td>109</td>
</tr>
<tr>
<td>Appendix 1 - A List of MR Routines</td>
<td>111</td>
</tr>
<tr>
<td>Appendix 2 - A List of Empress System Variables</td>
<td>121</td>
</tr>
<tr>
<td>Appendix 3 - White Paper: The Concept of I=MC&lt;sup&gt;2&lt;/sup&gt;</td>
<td>133</td>
</tr>
<tr>
<td>Appendix 4 - White Paper: Securing Data at Rest</td>
<td>147</td>
</tr>
</tbody>
</table>
What is Empress Ultra Embedded Database?
Empress Ultra Embedded Database is “The Embedded Real-Time Database Management System”. Empress Ultra Embedded Database is one of the most powerful and cost-effective database management systems available for organizations developing embedded, real-time applications using different types of operating systems including but not excluding, Linux, UNIX, FreeBSD, Mac OS, Windows, QNX and VxWorks. Support both process based and thread/task based environments, Empress Ultra Embedded Database is compact, agile and maintenance-free and is suited for embedded systems, real-time, communications & networking, military & defense, process control and scientific & engineering applications.
Empress Ultra Embedded Database consists of Empress Embedded Database, APIs, Connectivity Server, Replication Server and Utilities.

Empress Embedded Database complies to the ANSI standard with its own extensions and ACID with several transaction isolation levels and two phase commit. Empress Embedded Database is capable of managing all traditional data as well as text, binary and multimedia data in the most demanding applications from weather forecasting, space exploration, flight simulation, process control, medical primary care system and geographical information systems to digital imaging medical devices, POS, voice recorder, intelligent routers, video editing machine, print manager and sensor data collector.

Empress Ultra Embedded Database has API’s for C, C++, SQL, ODBC, and JDBC. Empress Ultra Embedded Database can work in stand-alone mode with C/C++ API, Embedded SQL, ODBC Local Interface, JDBC Interface and Interactive SQL.

Empress Ultra Embedded Database has two servers as options. Connectivity Server supports ODBC and JDBC in a client/server mode with clients and servers for Linux, UNIX, FreeBSD, Mac OS, Windows, QNX and VxWorks environments. Replication Server provides highly flexible database replication functionality for LAN and WAN applications.

Empress provides a set of utilities for system administration ranging from setting the database and operating system environments to tuning the database for maximum performance. The Empress Utilities can be categorized as follows: Empress System Variables, Empress Database Administrative Variables, Database Administration, Database Integrity Checking and Maintenance, Database Warm Restart, Database Backup and Recovery, Database Version Upgrade, Data Import and Export, Empress Monitors and Statistics, Shared Memory and Batch SQL.
Layered Architecture of Empress

1. What is Empress Ultra Embedded Database?

Empress Database Kernel with Optional Encryption
LINUX, UNIX, WINDOWS, or REAL TIME O/S LAYER
What is Empress Software Inc?
Empress Software Inc develops, markets and supports the *Empress* Ultra Embedded Database and associated software products worldwide. In the USA, Empress Software Inc markets directly to end-users and through Original Equipment Manufacturers (OEMs), Value-Added Resellers (VARs) and System Integrators. International sales and support are carried out through distributor channels as well as Value Added Resellers and System Integrators.

Over 30 years, *Empress* products have proven themselves in the most demanding applications. Software developers requiring power over embedded database applications have relied on *Empress* as the foundation for systems in such diverse fields as medical diagnostics, weather forecasting, space exploration, military and defense, flight simulation, process control and image and voice management.

Founded in 1979, Empress Software is a privately held company that serves a growing global community from offices in the Washington, D.C. suburb of Beltsville, Maryland. International sales and support are directed from Markham, Canada. *Empress* Embedded Database was designed by John Kornatowski and Ivor Ladd both of whom founded Empress Software Inc.

As embedded database application developers’ needs and expectations become more sophisticated, *Empress* continues to evolve. Empress Software account and technical representatives, support programs and user groups provide a valuable, two-way channel of information between company and customer. The input and experiences of users and developers are reflected in the ongoing enhancement of *Empress* products at the company's Research and Development Center. Our goal at *Empress* is to foster long-term business partnerships through which we can grow together with our customers. It's how we both ensure a powerful future.
3 Why Use Empress?
Why use Empress Ultra Embedded Database?

1. It’s Powerful
2. It’s Fast
3. It’s Reliable
4. It’s Cost Effective

That’s why!
1. It’s Powerful  

Adaptable, Scalable and Extensible

- For Linux, UNIX, FreeBSD, Mac OS, Windows, QNX, VxWorks and other Real-Time operating systems
- 32 and 64 bit O/S support
- **Empress** Kernel-level “C” API for total control
- API’s for C, C++, Java, ODBC, JDBC
- In-memory or on-disk
- Triggers, stored procedures & PSMs
- Fast, small footprint
- Two servers and stand-alone mode
- Handle text, binary and multimedia data
- Store and manipulate huge amounts of data
- Distribute data across networks
- Distribute data across different platforms and operating systems
- Full supercomputer to SOC(System On a Chip) functionality
- Unlimited user definable functions and operators
- Supports English, European, Japanese, Chinese, Unicode (UTF-8)
- Layered architecture makes **Empress** extensible and highly portable
2. It’s Fast

Achieve Microsecond Performances

- Use shared memory for faster performance
- Put part or all of the database “in memory”
- Use hundreds of system and tuning variables
- Use data type tuning parameters for best speed
- Use MicroSecond timestamp data type
- Choose how fast you want your SQL: DSQL, ESQL, Java SQL and iSQL
- Choose how fast you want your API: “C” MR Routines, C++, DSQL, ODBC, JDBC
- Choose how fast you want your server: stand-alone mode, Connectivity, Replication
- Best choices/tuning may give 3x to 5x better speed
3. **It’s Reliable**  

Robust, Always-On

- Unattended operations 24/7
- Dynamically self regulating
- Keeps on working for days, months, years
- Still in use since 1984
- Hot swappable replicates
- Automatic two-phase commit for transactions
- Referential constraints and range checks in the data dictionary
- Used in mission critical applications: banking, space programs, printer drivers, USAF fighter operations, battlefield deployment, nuclear reactors, electricity grids, gas station POS, internet telephony (VOIP), medical records, factory automation
3. Why Use Empress?

4. It’s Cost Effective  Affordable, Time-to-Market

- More features and performance for less cost
- **Empress** delivers significant savings on licenses and support
- Does not require dedicated DBAs
- Routine installation, backups and upgrades tasks can be automated, reducing administration costs
- Best ranked support services at remarkably affordable cost
- Adaptable pricing to work with OEM, VAR, SI business models
- Very affordable alternative for tight budgets
- On GSA schedule GSA# GS-35F-4989H
4  How to Use Empress?
**Empress** can be configured in a variety of different ways to achieve many different business objectives.

Two basic scenarios are:

- Stand-alone
- Client/Server

In a stand-alone scenario there is no need to run an **Empress** Server. The **Empress** Embedded Database engine library is linked directly with a target user application. Stand-alone scenarios are very attractive for embedded database applications running in a single address space (one process).

In a client/server scenario, the **Empress** Connectivity Server is required. A user application is linked with an **Empress** client library to access the **Empress** Connectivity Server at application runtime. Client/Server scenarios are required for remote access to an **Empress** database but may also be used locally.

**Both the stand-alone and client/server scenarios can run on the same system at the same time.**
Stand-alone Scenarios

**Stand-alone Scenario - Empress C & C++ API MR Routines**

*Fig. 4.1* Stand-alone Scenario – Empress C/C++ API MR Routines

*Empress* C/C++ API MR Routines is a unique feature of *Empress* that gives users access to the Embedded Database kernel libraries. This *Empress* API provides the fastest means of accessing *Empress* databases. MR Routines give the developer maximum control over time and space in developing real-time embedded database applications.

Highly demanding applications can take full advantage of the efficiency and flexibility of the *Empress* C/C++ API MR Routines.
Stand-alone Scenario - Empress Embedded SQL

The Empress C/C++ API Embedded SQL is an ANSI standard SQL interface. Empress Embedded SQL Interface is less detailed in comparison to the MR Routines. There is some overhead in performance and space utilization compared to the MR Routines. Empress Embedded SQL Interface is also well suited for usage in real-time embedded database applications.
Stand-alone Scenario - Empress ODBC

Fig. 4.3  Stand-alone Scenario – Empress ODBC Local Access Interface

_Empress_ ODBC Local Access Interface enables many 3rd party ODBC capable software packages to access a local _Empress_ database. _Empress_ ODBC Local Access Interface is a good choice if there is a need for ODBC access to data residing on the same machine as the database application.
Stand-alone Scenario - Empress JDBC

Empress JDBC Local Access Interface enables users to develop Java programs or to use already written 3rd party Java software packages to access a local Empress Database. The Empress JDBC Local Access Interface is another good choice when there is a need to access data residing on the same machine as the database application.
Scenarios using Empress Connectivity Server for Client/Server

Scenario using Empress ODBC Interface

Figure 4.5 shows three possible ways of using Empress ODBC Interface. A database application is running on Machine B and accessing the database on Machine A via Empress Connectivity Server. This is a typical client/server scenario. The other two applications running on Machine A are accessing the database locally, via Empress Connectivity Server and via Empress ODBC Local Access Interface.
Figure 4.6 shows three possible ways to use *Empress* JDBC Interface. The Java application on Machine B uses *Empress* JDBC Interface to remotely access the *Empress* database on Machine A via *Empress* Connectivity Server. This is a typical client/server scenario. Two Java applications running in Machine A are accessing the database locally, either via *Empress* Connectivity Server or via *Empress* JDBC Local Access Interface.
4. How to Use Empress?

**Scenario using Web Technologies and Empress**

An *Empress* database may be accessed via a variety of Web technologies, such as Java Server Pages (JSP), PHP scripting language, or Microsoft Active Server Pages (ASP) in conjunction with *Empress* ODBC or JDBC Interfaces.

The Common Gateway Interface (CGI) is another mechanism that allows Web servers to execute *Empress* database programs and incorporate their output into the text, graphics, and audio sent to a Web browser. For Internet/Intranet access to *Empress* databases,

In addition developers can create their own CGI programs using different *Empress* interfaces, such as *Empress* MR Routines, *Empress* Embedded SQL or *Empress* ODBC Interface.
Performance Comparison among Empress Interfaces

When Empress interfaces are compared in regard to performance, the following list is, in general, an ordered list from the fastest to the slowest interface:

1. Empress C/C++ API MR Routines
2. Empress C/C++ API Embedded SQL
3. Empress ODBC Local Access Interface
4. Empress JDBC Local Access Interface
5. Empress ODBC Interface
6. Empress JDBC Interface
Replication Scenarios

Replication Scenario - High Data Availability

*Empress* Replication may be utilized in scenarios for high data availability as shown in Figure 4.9. An application on Machine B can access a local source of data instead of a remote one on Machine A. By accessing an alternative data source, an application could bypass situations when Machine A is not available.

Furthermore, access to a local copy of data, reduces network traffic which contributes to better performance.

Machine A can run an operating system that is different from Machine B.
4. How to Use Empress?

Replication Scenario - Distributing Data

Figure 4.10 shows a scenario for distributing data across multiple locations. Queries can be performed at different locations each containing a copy of the data, so that the workload for doing a query through a single site is reduced.

Furthermore, by providing copies of data through the network, data is automatically distributed to remote sites in a network.
Figure 4.11 represents a scenario for alleviating Lock Contention situations. By off loading database resource lock contention between database applications, an application can access an alternative data source instead of only one. This allows many applications to access copies of the same data using different lock resources.
Figure 4.12 represents a scenario where *Empress* Replication is used to resolve recovery from system failures. In the case of a system failure on a machine, Machine A or Machine B can shut down its operation while the other keeps running.
Where is Empress Being Used?
5. Where is Empress Being Used?

*Empress* is targeting its embedded database at the following embedded and real-time markets:

- Networking and Telecom
- Control and Automation
- Automotive
- Security and Defense
- Data Acquisition and Instrumentation

*Empress’* power has made it the Embedded Database of choice in many diverse fields, including embedded and real-time database applications, especially those using binary data. The following are some of the areas where *Empress* has proven its strength:

- Administration
- Advertising
- Aerospace
- Automotive
- Call Center
- Customer Relationship Management
- Distribution Center
- Document Server
- Geographical Information Systems
- Geological Research
- Manufacturing Process Control
- Medical Equipment
- Meteorological Research (Weather Forecasting)
- Military and Defense
- Power Plant Production and Energy Distribution
- Public Utilities
- Publishing
- Satellite and RADAR Systems
- Telemetry
- Text Search Engine
- Transportation
- Travel Industry
- WEB Database Publishing
Most computer-based solutions are made of several components. The diagram below of Solutions vs. Market Segments shows these components. The “Database Engine” component, being wide, spans many different market segments. **Empress** Embedded Database is used as a database engine in many solutions designed for different market segments.

**Empress** is the “Embedded Real-Time Database.” But, Embedded Real-Time is a much wider category and is not just a market segment. A Manufacturing Process Control solution requires an embedded Database Engine with deterministic response capabilities just as much as an IT application that must deal with system response deterioration after data has grown beyond tested limits.

Below is a selected list of market segments and applications that have used **Empress** Embedded Database together with key reasons why **Empress** Embedded Database was chosen.
5. Where is Empress Being Used?

**Administration**

Example of Application:  
- Attendance System

Reasons for choosing *Empress* Embedded Database:  
- Fast Performance  
- Embeddable  
- Maintenance-free

**Advertising**

Example of Application:  
- Dynamic Media Management

Reasons for choosing *Empress* Embedded Database:  
- Multi-platform Availability  
- Low Acquisition Cost  
- Maintenance-free

**Aerospace**

Example of Application:  
- Flight Simulator

Reasons for choosing *Empress* Embedded Database:  
- High-speed Data Ingest  
- Kernel Level C API  
- Single Memory Space Deployment

**Automotive**

Example of Application:  
- Auto Parts Catalog

Reasons for choosing *Empress* Embedded Database:  
- Fast Performance  
- Maintenance-free  
- Low Cost

**Banking**

Example of Application:  
- Bank Teller System

Reasons for choosing *Empress* Embedded Database:  
- Fast Performance  
- Maintenance-free  
- Low Cost

**Call Center**

Example of Application:  
- Voice Recording System

Reasons for choosing *Empress* Embedded Database:  
- Fast Performance  
- Kernel Level C API  
- Maintenance-free
Customer Relationship Management

Example of Application:
• Point of Sales (POS) System

Reasons for choosing *Empress* Embedded Database:
• Low Cost
• Multi-platform Availability
• Robust ODBC Interface

Data Collector

Example of Application:
• Steel Mill Sensor Data Collection

Reasons for choosing *Empress* Embedded Database:
• Fast Performance
• Multi-Platform Support
• Time Saving for Development

Distribution Center

Example of Application:
• Sales Order Tracking and Inventory Management

Reasons for choosing *Empress* Embedded Database:
• Scalable
• Very Reliable
• Good Development Environment

Document Server

Example of Application:
• Enterprise Form Distribution Center

Reasons for choosing *Empress* Embedded Database:
• Superb Multi-media Handling
• Embeddable
• Excellent Throughput and Performance

Geographical Information Systems

Example of Application:
• Public Utilities Management

Reasons for choosing *Empress* Embedded Database:
• Superb Multi-media Handling
• Robust SQL Interface
• Very Reliable

Geological Research

Example of Application:
• Seismic Activities Logging System

Reasons for choosing *Empress* Embedded Database:
• Very Reliable
• Embeddable SQL Interface
• Small Footprint
5. Where is Empress Being Used?

**Manufacturing Process Control**

Example of Application:
- Postal Sorting System

Reasons for choosing *Empress* Embedded Database:
- High-speed Data Ingest
- Kernel Level C API
- Maintenance-free

**Medical Equipment**

Example of Application:
- Digital Radiography

Reasons for choosing *Empress* Embedded Database:
- Kernel Level C API
- Fast Performance
- Embeddable

**Meteorological Research**

Example of Application:
- Historical Weather Data Collection

Reasons for choosing *Empress* Embedded Database:
- Superior Multi-media Handling
- Maintenance-free
- Low Acquisition Cost

**Military and Defense**

Example of Application:
- Advanced Battlefield Computer System

Reasons for choosing *Empress* Embedded Database:
- Maintenance-free Operation
- Scalable
- Multi-platform Availability

**Power Plant Production and Energy Distribution**

Example of Application:
- Hydro-Electric Plant Control System

Reasons for choosing *Empress* Embedded Database:
- Real-time Response
- Small Footprint
- Robust
5. Where is Empress Being Used?

**Public Utilities**

Example of Application:
- Public Utilities Management System

Reasons for choosing *Empress* Embedded Database:
- Kernel Level C API
- Maintenance-free
- Low Cost

**Publishing**

Example of Application:
- Yellow Pages Management System

Reasons for choosing *Empress* Embedded Database:
- Superb Multi-media Handling
- Kernel Level C API
- Multi-platform Availability

**Satellite and RADAR Systems**

Example of Application:
- Earth Observation System

Reasons for choosing *Empress* Embedded Database:
- High-speed Data Ingest
- Kernel Level C API
- Maintenance-free

**Telemetry**

Example of Application:
- Route Table Management of IP Routers

Reasons for choosing *Empress* Embedded Database:
- Low Cost
- Kernel Level C API
- Maintenance-free
- Hierarchical Join
- Cascade Delete
- Transaction Span in Dealing with Database and Non-Database Operations
- *Empress* and Application can Run in the Same Address Space

**Text Search Engine**

Example of Application:
- WEB Portal

Reasons for choosing *Empress* Embedded Database:
- Fast Performance
- Kernel Level C API
- Maintenance-free
5. Where is Empress Being Used?

**Transportation**

Example of Application:
- Railway Tracks Maintenance System

Reasons for choosing *Empress* Embedded Database:
- Deterministic
- Embeddable
- Maintenance-free

**Travel Industry**

Example of Application:
- Airport Information System

Reasons for choosing *Empress* Embedded Database:
- Kernel Level C API
- Very Reliable
- Low Cost

**WEB Database Publishing**

Example of Application:
- WEB Hosting Management

Reasons for choosing *Empress* Embedded Database:
- Scalable
- Wide Range of Supported API’s
- Very Reliable
6 What are Empress’ Major Features and Benefits?
There are two modes of using *Empress* Ultra Embedded Database and each mode has its own major features and benefits.

Mode one is Application Development. This is where an application developer or an application development team designs, creates and tests application built using *Empress* Ultra Embedded Database.

Mode two is Production Deployment. This is where the application developed in mode one is deployed for day to day use on one or many computers.
Application Development

Mode 1

The first mode of Empress Ultra Embedded Database utilization is Application Development.

Application developers find that software which is flexible and technologically advanced allows for fast production of advanced applications.

Empress Ultra Embedded Database is well suited for fast development of advanced applications.

Below are some of the major features and benefits in Application Development Mode.

Embeddable

Empress Ultra Embedded Database and utilities can be linked with a client application, so that the application and Empress run in a single address space. This allows applications using Empress Embedded Database to be deployed as a single unified program that is potentially robust and efficient.

Deterministic Response

Empress Ultra Embedded Database supports deterministic response through alternative data structures and time-out functions. Alternative data structures allow database response time to be relatively constant. Time-out functions on database calls return control to a calling application once a specified time limit has been exceeded. An application developer can set time-out limits on inserts, deletes, updates and selects.

Fast Performance

Empress Ultra Embedded Database offers a kernel level "C" API called MR routines. Applications written in MR routines do not need to be parsed at runtime. Additional speed functionality in the MR routines include fine control over locking, memory management, and selection based on record numbers. In-memory and on-disk data storage in a single embedded database system allows developers to optimize applications for speed and persistence.
6. What are Empress’ Major Features and Benefits

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexible Development</td>
<td>Developers can use familiar development tools to build applications embedded with the Empress database engine. There is no need to learn a new development environment or new programming language. Empress Ultra Embedded Database supports a number of standard API’s so most popular development tools can be easily connected to Empress Ultra Embedded Database. Applications can be developed in a variety of programming languages including C, C++, JAVA, Visual Basic and Web Technologies.</td>
</tr>
<tr>
<td>Friendly Storage Requirements</td>
<td>Empress databases can be located on most storage devices as long as the operating system supports the storage device. Storage devices may be SCSI RAID, IDE, RAM, CD-RW, DVD-ROM or any new technology the O/S will support. In addition, Empress database tables may be split across many different storage devices. So database tables may be in RAM, on hard disk and on CD-ROM.</td>
</tr>
<tr>
<td>Small Footprint</td>
<td>Empress Ultra Embedded Database is highly modular. The size of the Empress footprint depends on the host operating system and the number of Empress options chosen, ranging from 600KB to 3MB.</td>
</tr>
<tr>
<td>Wide Range of Platforms</td>
<td>Empress Ultra Embedded Database is available on many hardware platforms and operating systems. In addition, Empress Embedded Database may be easily ported to a user requested hardware and operating system for a standard fee. Application developers, who embed Empress Ultra Embedded Database, have a large choice for both development and target systems.</td>
</tr>
<tr>
<td>Cost Effective Partnership Programs</td>
<td>Application developers have easy access to all Empress products. Empress Software offers a cost effective way for application developers to use all Empress products including product support, updates, consulting and discounts on runtime modules.</td>
</tr>
</tbody>
</table>
### Advanced Product Distribution

Empress Software distribution media contains all available Empress products for a specific hardware and operating system combination. An installation key is used to “unlock” the selected Empress products. Additional licenses are installed via new keys sent by email or fax.

### Innovative Solutions

Application developers can easily use Empress advanced features to quickly develop innovative applications ahead of their competitors.
6. What are Empress’ Major Features and Benefits

Production Deployment

Mode 2

The second mode of Empress Ultra Embedded Database utilization is Production Deployment.

Production Mode begins when an application is deployed on a target system.

The benefits of using Empress in Production Mode include reliability, high performance and maintenance-free operation.

Below are some of the major Empress features and benefits in Production Mode.

Significant Cost Savings

Once deployed in Production Mode, Empress has significant cost savings:

1. Empress has an excellent price/performance ratio.
2. Empress system requires little or no maintenance.
3. Empress support and upgrade fees are very cost effective.
4. Empress Embedded Database hardware requirements are low.

High Availability

Empress Ultra Embedded Database supports several high availability methodologies including database integrity utility, recovery log, support for RAID systems and Empress Replication Server.

Deployment Flexibility

Empress Ultra Embedded Database is available for many combinations of hardware and operating systems. Empress client and server software can be installed across a wide range of UNIX, Linux, Free BSD Windows and Real-Time systems.

Scalability

Empress Ultra Embedded Database can be deployed on devices, entry-level workstations as well as on high-end 64-bit multi-processor servers.
<table>
<thead>
<tr>
<th><strong>In-Depth Database Expertise</strong></th>
<th>Empress Software front line Technical Support staff is knowledgeable database experts who give high quality support.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Easy Access to User Data</strong></td>
<td>Users can access data stored in <em>Empress</em> database through standard interfaces, multiple API’s, kernel level interface application programs as well as many popular third party software products.</td>
</tr>
<tr>
<td><strong>High Performance</strong></td>
<td>Users of <em>Empress</em> Ultra Embedded Database will benefit from one of the fastest embedded real-time database engines in the industry.</td>
</tr>
</tbody>
</table>
6. What are Empress’ Major Features and Benefits
What Does Empress Run on?
7. What Does Empress Run on?

Supported Platforms

A list of major Linux, UNIX, Free BSD, Windows and Real-Time platforms that Empress runs on is shown in this chapter.
**Supported Platforms**

*Empress* Embedded Database is available in both client and server modes for all major Linux, UNIX, Free BSD, Windows and Real-Time platforms including:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Environment</th>
<th>Servers</th>
<th>Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX (IBM)</td>
<td>PPC</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Apple Mac O/SX</td>
<td>PPC, x86, x86-64</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Bluecat Linux (real-time)</td>
<td>x86</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>BSD</td>
<td>x86, x86-64</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Cray Unicos</td>
<td>Cray, x86-64</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Fedora Core</td>
<td>x86, x86-64</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IBM</td>
<td>AIX</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IRIX</td>
<td>SGI, x86-64 (AMD)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Linux (most)</td>
<td>x86, x86-64, most 2.6 kernels</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Lynx O/S (real-time)</td>
<td>PPC, x86</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MontaVista Linux (real-time)</td>
<td>ARM, MIPS, PPC, SH-4, x86</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Motorola PPC - BSD, Linux</td>
<td>Apple Mac OS-X, Linux PPC</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Motorola PPC Real-Time</td>
<td>Lynx O/S, QNX, VxWorks</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>QNX 4 &amp; 6</td>
<td>PPC, x86</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Red Hat 7, 8, 9</td>
<td>x86</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Red Hat EL 3, 4, 5</td>
<td>x86, x86-64</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUN Solaris 8, 9, 10</td>
<td>Sparc, x86-64</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUSE-LE 9, 10</td>
<td>x86, x86-64</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TimeSys Linux</td>
<td>PPC, x86</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Tru64 UNIX</td>
<td>DEC Alpha</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Ubuntu 6, 7, 8</td>
<td>x86</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VxWorks 5, 6</td>
<td>ARM, MIPS, PPC, SH-4, x86</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Wind River Linux</td>
<td>ARM, x86</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Windows CE</td>
<td>ARM, x86</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Windows Mobile 5, 6</td>
<td>ARM</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Windows Server 2003, 2008</td>
<td>x86, x86-64</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Windows 2000, XP, Vista</td>
<td>x86, x86-64</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
7. What Does Empress Run on?
8 Empress Maximum Sizes
Operational Parameters
Maximum number of Databases, Tables, Attributes, Size of Databases, Tables, Records and Attributes and other operational parameters are listed in this chapter.
Operational Parameters

The following table lists various *Empress* operational parameters. For practical purposes, sizes are limited by the availability of virtual memory on your system and allowable sizes of files and file systems on your disk, as well as the number of files and directories allowed on your system.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Databases</td>
<td>No <em>Empress</em> imposed limit.</td>
</tr>
<tr>
<td>Size of Database</td>
<td>A single database may occupy an entire file system. A single database may also span several file systems.</td>
</tr>
<tr>
<td>Size of Table (in bytes)</td>
<td>On most machines, this is the size of a 64-bit integer or about $2^{64}$ bytes.</td>
</tr>
<tr>
<td>Size of Record (in bytes)</td>
<td>On most machines, this is the size of a 64-bit integer or about $2^{64}$ bytes.</td>
</tr>
<tr>
<td>Size of Attribute (in bytes)</td>
<td>On most machines, this is the size of a 64-bit integer or about $2^{64}$ bytes.</td>
</tr>
<tr>
<td>Number of Tables/Database</td>
<td>32,767</td>
</tr>
<tr>
<td>Number of Attributes/Table</td>
<td>32,767</td>
</tr>
<tr>
<td>Number of Keys/Sort</td>
<td>No <em>Empress</em> imposed limit.</td>
</tr>
<tr>
<td>Number of Indices/Table</td>
<td>No <em>Empress</em> imposed limit.</td>
</tr>
<tr>
<td>Number of Attributes/Index</td>
<td>No <em>Empress</em> imposed limit.</td>
</tr>
<tr>
<td>Number of Levels-of-Nesting/Subquery</td>
<td>100</td>
</tr>
</tbody>
</table>

Product Profile
8. Empress Maximum Sizes
9 What are the Empress Products?
9. What are the Empress Products?

9.1 The Empress Software Products

*Empress* Embedded Database

The core of Empress Software Products is *Empress* Embedded Database Engine. All other products are layered or depend on the *Empress* Embedded Database Engine.

The *Empress* Embedded Database Engine can support a wide range of simultaneous users working with the database as well as user and system processes (together called “power units”). The number of power units required will need to be specified during product installation.

The number of power units can be increased simply by updating the installation key.

The *Empress* Embedded Database Engine can work in client/server or local mode or both at the same time.

*Empress* APIs

*Empress* has API’s for C, C++, Java, SQL, ODBC, and JDBC. *Empress* Embedded Database can work in stand-alone mode with C/C++ API, Embedded SQL, ODBC Local Interface, JDBC Interface and Interactive SQL.

*Empress* Connectivity Server

*Empress* Connectivity Server supports ODBC and JDBC using client/server architecture. The number of simultaneous connections to the *Empress* Connectivity Server will need to be specified during product installation.

*Empress* Replication Server

*Empress* Replication Server allows users to distribute data from one or more targets. There is no limit to the number of replicates.

*Empress* Utilities

The *Empress* Utilities can be categorized as follows: *Empress* System Variables, *Empress* Database Administrative Variables, Database Administration, Database Integrity Checking and Maintenance, Database Warm Restart, Database Backup and Recovery, Database Version Upgrade, Data Import and Export, *Empress* Monitors and Statistics, Shared Memory and Batch SQL.
9. What are the Empress Products?

**Empress Memory Embedded**

Empress Memory Embedded is small footprint (200K), object-oriented design, API for in-memory data creation, search and modification. It allows developers to build applications that require extreme high-speed access to application data. It also provides transparent index management and concurrent access management for application data in memory. Empress Memory Embedded can be used in both C and C++ programming environments.

**Empress Java SQL**

Empress Java SQL is a graphical Java SQL client program configured as a Stand-alone JDBC program. This allows users to view the structure of a local Empress database using JDBC, browse data in Empress tables and issue SQL commands on Empress tables.

**Empress Report Writer**

Empress Report Writer is a highly flexible tool for the making custom formatted reports. It can be used with Empress databases or as a stand-alone report writer for reading data from files. Empress Report Writer allows you to design a wide range of reports, with headers, footers, and pagination. Data can be read from database tables via Empress SQL SELECT statements or taken from operating system files.

**Empress Encryption**

The Empress database encryption is based on performing the encryption/decryption process either:

1. By using a hardware device/appliance; or
2. As a pure software solution using a Security Library that contains an encryption algorithm.

Users have the ability to create indexes on the encrypted columns of any Empress data type that can be normally indexed. Empress has designed encryption in its indexes in such a way to make indexes usable for all kinds of searches, not only for equality searches. Hence, there is no difference in the usability of indexes for searches on encrypted or non-encrypted columns.

For encrypting and decrypting data, symmetric cipher keys are used. Various public domain ciphers, including AES are standard with Empress and are user-selectable. Empress also provides the means to embed user-defined ciphers. The recommended cryptographic algorithm is Advanced Encryption Standard (AES).
9. **What are the Empress Products?**

The main benefits for implementing the encryption solution using Empress are:

- Secure all database data. A solution for protecting user data in a database including protection of all logs and backup files.
- Eliminate the potential for data loss that could be read by another party and all of the serious ramifications that accompany it, including bad press, loss of customers, government intervention.
- An efficient security solution. Insignificant performance overhead when performing the encryption/decryption process as a pure software solution and a small performance overhead when performing the encryption/decryption process by using a hardware device/appliance. Furthermore, Empress has implemented additional provisions to keep the size of the database increase minimal.
- No need for application code changes. This solution does not impact already written applications that use the data in an non-encrypted Empress database.
- No need for adding external provisions in the database to accommodate encryption such as stored procedures, triggers, views, etc. The Empress solution is painless for users who choose to convert their non-secure database solution to a secure one.
9.2 Support and Upgrades

Support by phone, fax and e-mail

The *Empress* Technical Services Department strives to provide the best technical support possible. We keep up-to-date information on the equipment and operating environments of our users so that we can give prompt, accurate and focused service. Users with maintenance agreements receive a support package containing checklists to aid communication with the Technical Services Department. All questions and requests from customers are assigned a reference number to allow tracking by both customers and *Empress*.

You can contact the Technical Services Department by email, phone or fax as needed.

*Empress* Technical Service’s representatives are available 9:00 am to 5:00 pm Eastern time, Monday through Friday.

Upgrades and Updates

A maintenance contract of one year's duration is typically purchased with the initial *Empress* license. Thereafter, maintenance is available for a yearly fee. The maintenance contract includes the following:

- bug fixes and work-arounds
- automatic product updates
- low-cost license transfers
9. What are the Empress Products?

9.3 Technical Services

The following services are also available:

- Installation
- Consulting
- Source-level customization
- Training

Installation
Empress Software engineers can install the Empress products and run verification and integrity tests either on-site or via remote access.

Consulting
Empress Software engineers can assist developers in designing, coding and performance tuning applications either on-site or via remote access.

Source Level Customization
Empress Software engineers can make specific user requested modifications at the source code level.

Training
Empress Software Inc. offers a variety of courses for users and developers. Course offerings range from the introductory to advanced user and developer levels. All taught by experienced professionals.

Training is available at the customer's site on a training-plus-expenses basis, or at the offices of Empress Software Inc.

The following is a selected list of courses available:
### Empress Product Overview
This one-day course introduces students to all the layers of the *Empress* Relational Database Management System. The function and trade-offs of developing applications using the various interfaces are discussed. Examples of the various database interfaces are presented.

### Introduction to Empress SQL
This one-day course provides an introduction to database management concepts, the relational database model, and *Empress*’ implementation of SQL.

### Advanced Empress SQL
This two-day course provides a detailed study of the *Empress* SQL and knowledge of the underlying structure of the *Empress* database.

### C Language Interface: MR Routines
This two-day course provides knowledge of how the high performance MR Routines can be used in a C program to access an *Empress* database. This is the kernel level access to the database, the most powerful and the fastest.

### C Language Interface: Empress Static and Dynamic SQL
This two-day course provides knowledge of how Dynamic and Static SQL commands can be embedded in a C program in order to access an *Empress* database.

### Empress Connectivity, ODBC Interface
This two-day course provides knowledge of how to use the ODBC-API in either stand-alone or client/server mode.

### Java Programming using Empress JDBC
This two-day course provides knowledge of how to use the JDBC-API in stand-alone or client/server mode.

### Database Design & Tuning
This two-day course presents the various performance and efficiency issues that the database designer and applications developer are faced with and how to improve performance.

### Database Administration
This three-day course is intended for database administrators. It provides information on the internals of an *Empress* database, and explains the use of administrative utilities.

### Empress Replication
This one-day course provides knowledge on how the *Empress* Replication Servers and Clients work.

### Empress Persistent Stored Modules (PSM)
This two-day course provides knowledge on setting up triggers and stored procedures using “C” or “SQL”. PSM may be used for specialized data input or output, compression, encryption, data manipulation or to provide other functionality.
9. What are the Empress Products?
What is in the Empress Ultra Embedded Database Package?
10. What is in the Empress Ultra Embedded Package?

10.1 The Empress Ultra Embedded Database Package

An *Empress* Ultra Embedded Database package is made up as follows:

1) An *Empress* CD for the specific hardware and operating system requested containing development or cross platform development toolkit:

2) A letter that contains an installation key for the *Empress* CD above.
10.2 The Empress CD

An *Empress* CD may contain:

**Standard:**

- *Empress* Embedded Database Engine
- C/C++ API, Embedded SQL, Interactive SQL
- *Empress* Utilities
- *Empress* Report Writer
- Manual set in HTML format readable on Linux, UNIX, FreeBSD, Windows and Real-Time with an appropriate Browser

**Optional Products:**

- Local ODBC, JDBC Interface
- Connectivity Server
- Selected ODBC clients
- Replication Server
- Real-Time Data Collector (selected platforms)
- Embedded Real-Time Toolkit (selected platforms)
- Java SQL
- *Empress* Encryption
10. What is in the Empress Ultra Embedded Package?

10.3 The Empress Installation Key

The installation key consists of 4 lines each made up of 3 groups of 4 characters (0-9, a, b, c, d, e, f) separated by dashes and looks like:

3ea5-3b1a-3538
5f5d-1054-6a3d
ac58-4177-5896
8526-7b0f-1ea4

The *Empress* installation process will ask for the key to be typed as part of the installation process.

The installation key controls the number of power units and what *Empress* options are activated. More power units or options can be added by simply updating to a new installation key.
Empress’ Features by Module
11. Empress' Features by Modules

11.1 Empress Embedded Database Engine

- Product Functionality
  For embedded applications that require a fast, reliable database management system with small footprint, Empress is unsurpassed in features, performance and power.

- Supported Data Types
  Empress provides a wide variety of data types, letting you choose the best type for your application.
Product Functionality

For data intensive applications that require a fast, reliable database management system, *Empress* is unsurpassed in the following functionalities:

- Small Footprint enhances use of Embedded Database for compact applications.
- Kernel and SQL Interface Routines allow multilevel control in the *Empress* layered architecture for optimization and rapid prototyping.
- Layered Architecture Accessible at 4 Levels lets developer trade-off between low-level system optimization and rapid prototyping.
- On-disk and in-memory operations.
- Fast Bulk Data Handling allows performance approaching flat file access speeds for binary objects.
- Bulk Chunks allow you to slice up large binary objects into smaller segments thus optimizing system performance.
- Multiple Attributes and File Indexes optimize performance.
- Persistent Stored Modules cut down development time by providing the power to create reusable functions for data manipulation.
- Triggers and Stored Procedures are stored and executed directly on the server and enhance the efficiency of database applications by eliminating repetitive programming and making automation easier.
- No Database Pre-Partitioning is required, thereby making optimal use of the native O/S file system.
- Referential Constraints and Range Checks ensure data integrity.
- MicroSecond Time Stamps allow you to store and retrieve the occurrence of real-time system events down to a millionth of a second.
- Time-out of API calls can be set to predefined time periods which ensure control remains with the calling application.
- Timeseries Index is an efficient alternative indexing mechanism for ongoing data that is sequentially ingested.
- Cascading UPDATE and DELETE prevents update anomalies.
On selected platforms, additional options include:

- JDBC Interface allows Java programmers to access the database using standard JDBC calls.
- C++ Host Language Interface calls MR Routines directly from C++ programs.
- 64 BIT Operating System Enabled for storing and manipulating much larger data using the latest and fastest operating system technology.
- National Language Support allows different language character sets to be used at both the program and HTML file page level.
- Unicode UTF-8 support.
## Supported Data Types

*Empress* supports ANSI and ODBC data types with Empress extensions. These data types are:

<table>
<thead>
<tr>
<th>Data Type Category</th>
<th>Empress Data Type Names</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>CHARACTER</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>CHAR</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>SQL_CHAR</td>
<td>ODBC</td>
</tr>
<tr>
<td></td>
<td>NATIONAL CHARACTER</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>NATIONAL CHAR</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>NCHAR</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>ECHARACTER</td>
<td>Empress Extension</td>
</tr>
<tr>
<td></td>
<td>ECHAR</td>
<td>Empress Extension</td>
</tr>
<tr>
<td></td>
<td>NLSCHARACTER</td>
<td>Empress Extension</td>
</tr>
<tr>
<td></td>
<td>NLSCHAR</td>
<td>Empress Extension</td>
</tr>
<tr>
<td></td>
<td>CHARACTER VARYING</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>CHAR VARYING</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>VARCHAR</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>SQL_VARCHAR</td>
<td>ODBC</td>
</tr>
<tr>
<td></td>
<td>NATIONAL CHARACTER VARYING</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>NATIONAL CHAR VARYING</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>NCHAR VARYING</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>VARCHAR VARYING</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>TEXT</td>
<td>Empress Extension</td>
</tr>
<tr>
<td></td>
<td>NLSTEXT</td>
<td>Empress Extension</td>
</tr>
<tr>
<td></td>
<td>CHARACTER LARGE OBJECT</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>CHAR LARGE OBJECT</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>CLOB</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>NATIONAL CHARACTER LARGE OBJECT</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>NATIONAL CHAR LARGE OBJECT</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>NCHAR LARGE OBJECT</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td>NCLOB</td>
<td>ANSI</td>
</tr>
</tbody>
</table>
### 11. Empress' Features by Modules

<table>
<thead>
<tr>
<th>Integer</th>
<th>TINYINT</th>
<th>SQL_TINYINT</th>
<th>INTEGER8</th>
<th>ANSI</th>
<th>ODBC</th>
<th>Empress Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SMALLINT</td>
<td>SQL_SMALLINT</td>
<td>INTEGER16</td>
<td>ANSI</td>
<td>ODBC</td>
<td>Empress Extension</td>
</tr>
<tr>
<td></td>
<td>INTEGER</td>
<td>INT</td>
<td>SQL_INTEGER</td>
<td>ANSI</td>
<td>ANSI</td>
<td>Empress Extension</td>
</tr>
<tr>
<td></td>
<td>BIGINT</td>
<td>SQL_BIGINT</td>
<td>INTEGER64</td>
<td>ANSI</td>
<td>ODBC</td>
<td>Empress Extension</td>
</tr>
<tr>
<td>Sequence</td>
<td>SEQUENCE32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Empress Extension</td>
</tr>
<tr>
<td></td>
<td>SEQUENCE64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Empress Extension</td>
</tr>
<tr>
<td>Boolean</td>
<td>BOOLEAN</td>
<td></td>
<td></td>
<td>ANSI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Float</td>
<td>REAL</td>
<td>SQL_REAL</td>
<td>FLOAT32</td>
<td></td>
<td></td>
<td>Empress Extension</td>
</tr>
<tr>
<td></td>
<td>DOUBLE PRECISION</td>
<td>SQL_DOUBLE</td>
<td>FLOAT64</td>
<td></td>
<td></td>
<td>Empress Extension</td>
</tr>
<tr>
<td></td>
<td>EFLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Empress Extension</td>
</tr>
<tr>
<td></td>
<td>FLOAT</td>
<td>SQL_FLOAT</td>
<td></td>
<td>ANSI</td>
<td>ODBC</td>
<td></td>
</tr>
<tr>
<td>Decimal</td>
<td>DECIMAL</td>
<td>DEC</td>
<td>NUMERIC</td>
<td>ANSI</td>
<td>ANSI</td>
<td>ODBC</td>
</tr>
<tr>
<td></td>
<td>SQL_DECIMAL</td>
<td>SQL_NUMERIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DOLLAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Empress Extension</td>
</tr>
<tr>
<td>Date</td>
<td>EDATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Empress Extension</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EPOCH_TIME</td>
<td>Empress Extension</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ETIME</td>
<td>Empress Extension</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MICROTIMESTAMP</td>
<td>Empress Extension</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>SQL_TYPE_DATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ANSI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>SQL_TYPE_TIME</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME WITHOUT TIME ZONE</td>
<td>ANSI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>SQL_TYPE_TIMESTAMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP WITHOUT TIME ZONE</td>
<td>ANSI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary (Byte Stream)</td>
<td>BULK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Empress Extension</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BINARY LARGE OBJECT</td>
<td>ANSI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td>ANSI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Missing data values are stored internally in a special form, and you may enter and recall these values by using the NULL keyword.

The following pages describe **Empress** data types extension.
**Empress** data types extension:

**Character Data**

With **Empress**, you may store character data in fixed-length fields using one of three ways:

- Store only printable characters and strip leading and trailing blanks from data you enter by using **ECHAR** or **ECHARACTER** type 1
- Store only printable characters and preserve leading and trailing blanks from data you enter by using **ECHAR** or **ECHARACTER** type 2
- Store any ASCII character except **NULL** by using type 3

**Text Data**

The **TEXT** data type will store character data varying from several characters to many pages, using only as much storage as is needed for the data. This is an extremely powerful and flexible way of storing a wide variety of text passages, including descriptions, abstracts, comments, and word processor output.

**National Language Support (NLS) Character Data**

The **NLSCHARACTER** data type is essentially the same as the **CHARACTER** with the exception of characters being stored in 8-bit.

**National Language Support (NLS) Text Data**

The **NLSTEXT** data type is essentially the same as **TEXT** with the exception of characters being stored in 8-bit.

**Multimedia Data**

**Empress** stores binary information as **BULK** data. This data type allows you to store data values of any length or format, such as those used to represent voice or graphical data. **Empress** dynamically allocates space when storing bulk data.
11. Empress' Features by Modules

Date Data

The DATE data type stores dates. There are nine date data formats:

Table 2-3: Empress DATE Data Type Formats

<table>
<thead>
<tr>
<th>Type</th>
<th>Format</th>
<th>Year Range</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>yyyymmddd</td>
<td>0000-9999</td>
<td>20020627</td>
</tr>
<tr>
<td>1</td>
<td>dd aaaaaaaaa yyyy</td>
<td>0000-9999</td>
<td>27 June 2002</td>
</tr>
<tr>
<td>2</td>
<td>aaaaaaaaaa dd, yyyy</td>
<td>0000-9999</td>
<td>June 27, 2002</td>
</tr>
<tr>
<td>3</td>
<td>mm/dd/yy</td>
<td>nn00-nn99 *</td>
<td>06/27/02</td>
</tr>
<tr>
<td>4</td>
<td>dd/mm/yy</td>
<td>nn00-nn99 *</td>
<td>27/06/02</td>
</tr>
<tr>
<td>5</td>
<td>dd aaaa yy</td>
<td>nn00-nn99 *</td>
<td>27 Jun 02</td>
</tr>
<tr>
<td>6</td>
<td>aaaa dd, yy</td>
<td>nn00-nn99 *</td>
<td>Jun 27, 02</td>
</tr>
<tr>
<td>7</td>
<td>mm/dd/yyyy</td>
<td>0000-9999</td>
<td>06/27/2002</td>
</tr>
<tr>
<td>8</td>
<td>dd/mm/yyyy</td>
<td>0000-9999</td>
<td>27/06/2002</td>
</tr>
</tbody>
</table>

*nn is defined by the Empress system variable MSDATELIMIT, the value of which is assigned by the user to indicate the preferred century for entering dates.

Time Data

Time data is stored as ETIME. To store time, you may use one of nine formats:

<table>
<thead>
<tr>
<th>Type</th>
<th>Output Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20030828145615</td>
</tr>
<tr>
<td>1</td>
<td>28 August 2003 14:56:15</td>
</tr>
<tr>
<td>2</td>
<td>August 28, 2003 14:56:15</td>
</tr>
<tr>
<td>3</td>
<td>08/28/03 14:56:15</td>
</tr>
<tr>
<td>4</td>
<td>28/08/03 14:56:15</td>
</tr>
<tr>
<td>5</td>
<td>28 Aug 03 14:56:15</td>
</tr>
<tr>
<td>6</td>
<td>Aug 28, 03 14:56:15</td>
</tr>
<tr>
<td>7</td>
<td>08/28/2003 14:56:15</td>
</tr>
<tr>
<td>8</td>
<td>28/08/2003 14:56:15</td>
</tr>
</tbody>
</table>
MicroSecond Timestamp Data

The MICROTIMESTAMP data type stores TIME plus microseconds information. The nine different MICROTIMESTAMP styles are shown below:

<table>
<thead>
<tr>
<th>Type</th>
<th>Output Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20070828145615234599</td>
</tr>
<tr>
<td>1</td>
<td>28 August 2007 14:56:15.234599</td>
</tr>
<tr>
<td>2</td>
<td>August 28, 2007 14:56:15.234599</td>
</tr>
<tr>
<td>3</td>
<td>08/28/07 14:56:15.234599</td>
</tr>
<tr>
<td>4</td>
<td>28/08/07 14:56:15.234599</td>
</tr>
<tr>
<td>5</td>
<td>28 Aug 07 14:56:15.234599</td>
</tr>
<tr>
<td>6</td>
<td>Aug 28, 07 14:56:15.234599</td>
</tr>
<tr>
<td>7</td>
<td>08/28/2007 14:56:15.234599</td>
</tr>
<tr>
<td>8</td>
<td>28/08/2007 14:56:15.234599</td>
</tr>
</tbody>
</table>

Decimal Data

Decimal numbers are stored as DECIMAL or DEC. You can specify the number of digits and decimal places you need for your data. You may store up to 38 digits.

Dollar Data

Dollar and currency are stored as DOLLAR. When using this function, you may store any amount up to 13 digits and use one of three different formats:

<table>
<thead>
<tr>
<th>Dollar Type</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1,234.00</td>
</tr>
<tr>
<td>2</td>
<td>$****1,234.00</td>
</tr>
<tr>
<td>3</td>
<td>1234.00</td>
</tr>
</tbody>
</table>

You can use an Empress variable to control the currency, cents, and the thousands' separator (i.e., DM1,000.00).
Floating Point Data

- **REAL**
- **EFLOAT**
- **DOUBLE PRECISION**

With FLOAT, you can specify the number of decimal places. Use **DOUBLE PRECISION** for very high precision.

Integer Data

Integer numeric values are stored as:

- **INTEGER8**
- **INTEGER32** or **SMALLINT**
- **INTEGER64**

Using **INTEGER64**, you can store twice the number of digits than with **INTEGER32** and eight times the number of digits than with **INTEGER8**.
11.2 Empress Servers

*Empress* Ultra Embedded Database can run in stand-alone mode using any of the "C" APIs including the *Empress* kernel level "MR" Routines, the Embedded SQL for "C", the ODBC Interface with "C" calls as well as the "C++" Interface, the JDBC for "Java" Interface, the Embedded Real-Time Toolkit and Interactive SQL.

The Connectivity Server supports ODBC and JDBC in a client/server environment with clients and servers for UNIX, Linux, Windows and Real-Time environments.

The Replication Server provides a highly flexible multi client/server database replication functionality for LAN and WAN applications.
Connectivity Server (ODBC, JDBC)

*Empress* Connectivity Server is an *Empress* database server that supports ODBC and JDBC using client/server architecture. The main components of *Empress* Connectivity are *Empress* Connectivity Server and *Empress* Client Drivers. *Empress* Connectivity uses proprietary communication protocol which allows application programs running on client systems to communicate with the database server running on a system where user data source resides. With *Empress* Connectivity, clients and servers can reside on different machines with different operating systems.
Replication Server

**Empress** Replication Server provides powerful features that allow users to distribute data from one source to one or more targets.

**Empress** Replication is a process of creating a set of database tables as replicates (copies) of a single master table and their synchronization to the current state of the master table. Typically, the set of tables lives in separate databases in separate locations.

Some of the more important benefits of using **Empress** Replication Server are:

- Higher data availability
- Reducing network traffic by using a local copy of data
- Offloading data contention between database applications
- Distributing system load across multiple locations
- Easy recovery from system failures
- Automatically providing data copies through a network
- Transferring data between different operating systems and hardware platforms

If needed **Empress** Replication can be set to replicate only subsets of data.
Stand-alone Mode

*Empress* Ultra Embedded Database has the additional capability of running in stand-alone (local) mode without needing a server, using any of the following APIs:

- C & C++ MR Routines
- C & C++ Embedded SQL
- C MSCALL
- ODBC Interface
- JDBC Interface
11.3 Empress APIs

*Empress* Ultra Embedded Database supports a number of different API's in several programming languages to give the developer maximum choice and flexibility for building applications. The "C" programming language has the most APIs including the kernel level API MR Routines, Embedded SQL, MSCALL and ODBC. There are also API's for C++ and JAVA.

Applications developed using these APIs may be run in stand-alone and/or server modes.
C-API MR Routines

*Empress* provides a set of Database Manipulation routines for direct access to database structures using the C programming language. This interface provides faster execution times than any of the query language interfaces, since the parser is not invoked for each call. They also provide much greater control over what is done with the retrieved data, allowing you to perform complex arithmetic functions, generate custom-defined report formats, develop high-level user interfaces, and so on. The MR Routines are grouped as follows:

1. Open Routines
2. Lock Routines
3. Allocate Space Routines
4. Attribute Descriptor Routines
5. Filling Records Routines
6. Insert Routines
7. Delete Routines
8. Update Routines
9. Initialize Retrieval Routines
10. Retrieval Routines
11. Compare Routines
12. Input Conversion Routines
13. Qualification Routines
14. Transaction Routines
15. Expression Routines
16. Error Handling Routines
17. Execute SQL Commands
C-API Embedded SQL

The *Empress* C-API Embedded SQL allows for accessing and processing data stored in *Empress* databases, using SQL commands from within a C program. The following operations on tables in *Empress* databases are supported:

- Adding rows to a table.
- Updating rows in a table.
- Deleting rows from a table.
- Selecting rows from a table.
- Adding, updating and deleting all or part of a context of selected rows.
- Use of the transaction facilities of *Empress* to maintain data integrity.
C-API MSCALL

SQL commands as used in Empress SQL (Interactive SQL) may be embedded in a "C" program with the routine MSCALL. The routine MSCALL takes two arguments, the database name and the SQL command.

\[
\text{mscall (database\_directory, sql\_command);}\
\]

Both the database name and the SQL command are strings. The strings may be built by the application program. The routine returns zero (0) if the statement succeeds and a non-zero value if it fails.
11. Empress' Features by Modules

**ODBC Interface**

ODBC is a standard API that allows Client Applications running on Linux, UNIX, Free BSD, Windows or Real-Time systems to access data from a variety of sources either in client/server or stand-alone mode. Client Applications written using the ODBC API Standard should be able to access any ODBC Data Source providing that a corresponding driver exists to support that Data Source. The Data Source may reside on a remote Server platform connected by via a network or locally on the same computer.

ODBC was introduced as a standard to provide continuity between applications using databases and database engines. The goal was to encapsulate and standardize the link between applications and data, so developers could write programs that function independently of the characteristics specific to any given database platform.
JDBC Interface

JDBC is the standard API for the Java Programming Language from Sun Microsystems that connects Java programs to database management systems. It consists of a set of classes and interfaces written in Java. JDBC is platform independent and when combined with the unique cross-platform capabilities of Java, developers have the ability to write a database application once and execute it anywhere.

Empress JDBC Interface is the Empress implementation of the JDBC API that enables Java programs to execute Empress SQL statements. Users of the Empress JDBC Interface can write applications, create servlets and applets that easily connect to databases, send SQL queries, and quickly process these results.
C++ Interface

The *Empress* C++ Embedded SQL allows you to access and process data stored in *Empress* databases, using SQL commands from a C++ program. The following operations on tables in *Empress* databases are supported:

- Adding rows to a table.
- Updating rows in a table.
- Deleting rows from a table.
- Selecting rows from a table.
- Adding, updating and deleting all or part of a context of selected rows.
- Use of the transaction facilities of *Empress* to maintain data integrity.

*Empress* provides a set of Database Manipulation routines for direct access to database structures using the C++ programming language. This interface provides faster execution times than any of the query language interfaces, since the parser is not invoked for each call. They also provide much greater control over what is done with the retrieved data, allowing you to perform complex arithmetic functions, generate custom-defined report formats, develop high-level user interfaces, and so on. The MR Routines are grouped as follows:

1. Open Routines
2. Lock Routines
3. Allocate Space Routines
4. Attribute Descriptor Routines
5. Filling Records Routines
6. Insert Routines
7. Delete Routines
8. Update Routines
9. Initialize Retrieval Routines
10. Retrieval Routines
11. Compare Routines
12. Input Conversion Routines
13. Qualification Routines
14. Transaction Routines
15. Expression Routines
16. Error Handling Routines
17. Execute SQL Commands
11.4 Empress Utilities

Empress provides a set of utilities for system administration ranging from setting the database and operating system environments to tuning the database for maximum performance. The Empress Utilities can be categorized as follows:

- Empress System Variables
- Empress Database Administrative Variables
- Database Administration
- Database Integrity Checking and Maintenance
- Database Backup and Recovery
- Database Version Upgrade
- Data Import and Export
- Empress Monitors and Statistics
- Shared Memory
- Batch SQL
Empress System Variables

*Empress* provides system variables to give each user control over the *Empress* environment. The system variables are used to set such things as the number of spaces between columns in the output from a *SELECT* statement, the system editor, the attribute separator in a dump file, and so on. These variables all have system defaults. They may be reset using *Empress* SQL, application programs or in the operating system.
Empress Database Administrative Variables

*Empress* has a number of variables that give users control over specific aspects of the *Empress* database environment. When a new *Empress* database is created, all these variables have default values. These default values may be changed by the user and the variables may be used to set such things as:

- name of the database administrator,
- enable audit trail logging,
- privileges granted on database data dictionary tables,
- default locking level for newly created tables,
- activate lock statistics information gathering,
- default access permissions for database tables,
- privileges granted on tables at creation,
- specify recovery logs,
- specify database logs,
- etc.
11. Empress' Features by Modules

Database Administration

*Empress* provides a number of utilities for the database administrator to manage *Empress* databases such as:

- create and remove databases
- restrict database access to a database administrator
- restrict database access to single user mode only
- obtain locking information
- view and reset Lock Manager statistics
- remove defunct locks
- map tables to shared memory
- view shared memory information
- view shared memory statistics
- and many more.
Database Integrity Checking and Maintenance

*Empress* Embedded Database processes normally perform a formal shutdown procedure to initiate proper cleanup procedures. These procedures include clearing outstanding locks, disconnecting outstanding connections to the server, deleting temporary files, deleting temporary tables and resolving transactions.

When an *Empress* Embedded Database process is killed (by a machine crash or by a console KILL command); it may not have a chance to clean up properly. As a result, locks without owners and leftover server connections may remain.

To address these problems, *Empress* includes a database integrity checking and maintenance utility that checks databases and performs cleanup operations.
Empress’ Features by Modules

Database Backup and Recovery

**Empress** provides utilities for both on-line and off-line backup (archive) as well as recovery utilities to recover the database from a backup copy. **Empress** backup utilities may create a backup copy of a database on magnetic tape, disk, CD, DVD or other storage media.

An on-line backup of a database means copying the database while it is being used. An off-line backup means copying the database while it is not being used. The on-line backup facility is important to sites with databases that must be available at all times; these sites cannot afford to take the database off-line for the duration of a backup. The on-line backup procedure is transparent to the users and database integrity is maintained.

In case of a hardware problem, a system failure or a user error that destroys the database, the **Empress** recovery utility can be used to restore data from the last backup copy of the database, and roll forward the data from the current recovery log file. This will restore the database up to the point right before the incident.
Database Version Upgrade

*Empress* provides two utilities for database version upgrades. The first utility converts an old *Empress* version database to a new *Empress* version database. The second utility upgrades a database created by the earlier release of *Empress* to be compatible with the newly installed version of *Empress*. Users are recommended to back up the database prior to executing these *Empress* utilities.

Once the database is upgraded, only the newly installed *Empress* version can access the database.
Data Import and Export

The *Empress* Export utility exports tables from an *Empress* database and the Import utility imports tables to an *Empress* database using data generated by the Export program. These two programs may be used to export and import database tables across operating systems.
Empress Monitors and Statistics

Empress provides a number of utilities to allow users to monitor and gather statistics on Empress databases and Empress servers, such as:

- **Empress** Database Log Analyzer Utility
- **Empress** Server Log Analyzer Utility

Empress Database Log Analyzer Utility is used to analyze and display data from the Empress Database Log File showing aggregate statistical results for the time frame in question such as:

- the number of connections
- connection duration
- number of rejected connections due to the insufficient license resources

Empress Server Log Analyzer Utility is used to analyze and display data from the Empress Server Log File showing the history of operations performed by the Empress Server, and the history of client requests made to an Empress Server.

Empress Server Log Analyzer Utility can report information such as:

- number of servers started up
- number of idle servers
- number of servers in use
- number of connections
- connection durations
- number of rejected connections due to insufficient licenses resources
Shared Memory

*Empress*’ Shared Memory Utility lets users enhance the performance of the *Empress* Embedded Database by using Shared Memory, an operating system feature that allows several processes to share a specified section of main memory. By using Shared Memory to reduce file I/O, database performance can be improved.

- By placing Lock Information and Locks in Shared Memory rather than in files, multiple database processes can access Lock Information and Locks with improved performance.
- Global Buffers allow data from "one or more" files to be kept in Shared Memory for access by one or more processes. This enhances data access to those files.
- Mapped Files allow data from "one or more" files to be kept in Shared Memory and do so in a manner different from Global Buffers. Again, this will enhance data access to those particular files.
Empress' Features by Modules

11. Product Profile

Batch SQL

*Empress* provides a utility to execute a collection of one or more SQL statements as one unit, which may be invoked at a predefined time. The *Empress* Batch SQL utility allows rapid packaging of recurring database management commands. It is general enough to allow console programs, C routines, and other batch programs to be invoked. This encourages rapid design, development and deployment of recurring SQL commands for the database management system.
11. Empress' Features by Modules
12 Where Can I Get Empress?
Empress Software Inc. has offices, distributors and resellers worldwide. For further information, please contact the office nearest you:

**North America**

**In the United States contact:**

- **Empress Software Inc.**  
  11785 Beltsville Drive  
  Beltsville, Maryland  
  USA 20705  
  Toll Free: 1-866-626-8888  
  Tel: 301-220-1919  
  Fax: 301-220-1997  
  E-mail: info@empress.com  
  Web Site: www.empress.com

**In Canada contact:**

- **Empress Software Inc.**  
  3100 Steeles Avenue East  
  Markham, Ontario  
  Canada L3R 8T3  
  Toll Free: 1-866-626-8888  
  Tel: 905-513-8888  
  Fax: 905-513-1668  
  E-mail: info@empress.com  
  Web Site: www.empress.com

**Europe**

**In Great Britain, Northern Ireland, Republic of Ireland:**

- **Care Business Solutions**  
  Surrey, England  
  Tel: 44-1-483-861990  
  Fax: 44-1-483-860064  
  E-mail: den@drellden.co.uk
In France contact:

- **Katalis Sarl**
  Boulogne, France
  Tel: 33-6-0721-2733
  Fax: 33-1-5844-5858
  E-mail: katalis@katalis.fr

In Poland and neighboring countries contact:

- **Empress Consultants (Poland)**
  **Major Information Systems**
  ul. Chmielna 20 lok. 35
  00-020 Warszawa
  Poland
  Tel: 48-601-598-795
  E-mail: A.Borkowski@m-is.pl
  Web Site: [www.m-is.pl](http://www.m-is.pl)

In Russia, the Commonwealth of Independent States contact:

- **SWD Software Ltd.**
  Pr. Gagarina 23
  St. Petersburg
  Russia 196135
  Tel: 7-812-443-0260
  Fax: 7-812-443-0497
  E-mail: info@empress.ru
  Web site: [www.empress.ru](http://www.empress.ru)

In other European countries contact:

- **Empress Software Inc.**
  3100 Steeles Avenue East
  Markham, Ontario
  Canada L3R 8T3
  Toll Free: 1-866-626-8888
  Tel: 905-513-8888
  Fax: 905-513-1668
  E-mail: info@empress.com
  Web Site: [www.empress.com](http://www.empress.com)
12. Where Can I get Empress?

Asia

**In Japan contact:**

- **Empress Data Systems Inc.**  
  Yokohama, Kanagawa-ken, Japan  
  Tel: 81-45-290-6120  
  Fax: 81-45-290-6130  
  E-mail: info@empress-ds.co.jp

- **Miura Corporation**  
  Minato-Ku, Tokyo, Japan  
  Tel: 81-3-5441-7287  
  Fax: 81-3-5441-2687  
  E-mail: jiji@tokyo.kkmiura.co.jp  
  Web site: www.kkmiura.co.jp

- **Planners Land**  
  Naka-ku, Nagoya, Japan  
  Tel: 81-26-258-6236  
  Fax: 81-52-265-2930  
  E-mail: info@planners.co.jp  
  Web site: www.planners.co.jp

- **SORUN Corporation**  
  Minato-Ku, Tokyo, Japan  
  Tel: 81-3-5427-5584  
  Fax: 81-3-5427-5507  
  E-mail: tech-sup@sorun.co.jp  
  Web site: www.sorun.co.jp

**In South Korea contact:**

- **ThinkM Inc.**  
  Seoul, South Korea  
  Tel: 02-2202-5651  
  Fax: 02-2202-5655  
  E-mail: bkback@thinkm.co.kr  
  Web site: www.thinkm.co.kr

**In China contact:**

- **XiKeyouxue Network Technology CO.,Ltd.**  
  Shanghai, P.R. China  
  Tel: 86-21-6317-7857  
  Fax: 86-21-6317-3112  
  E-mail: oushuyun@gokei.cn  
  Web site: www.gokei.cn
In other Asian countries contact:

- **Empress Software Japan**
  Tamagawa 5-29-2 4th Floor
  Chofu, Tokyo, Japan
  Tel: 81-42-488-6985
  Fax: 81-42-488-6985
  E-mail: esi-info@empressjapan.co.jp
  Web Site: www.empressjapan.co.jp

South America

For South American contact:

- **Empress Software Inc.**
  3100 Steeles Avenue East
  Markham, Ontario
  Canada L3R 8T3
  Toll Free: 1-866-626-8888
  Tel: 905-513-8888
  Fax: 905-513-1668
  E-mail: info@empress.com
  Web Site: www.empress.com
12. Where Can I get Empress?
Appendix 1 - A List of MR Routines
This appendix contains summaries and descriptions of the following:

MR Routines
Appendix 1 – A list of MR Routines

The following table lists the MR Routines used to open and close tables:

<table>
<thead>
<tr>
<th>Open Routines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>mropen</td>
</tr>
<tr>
<td>mrtopen</td>
</tr>
<tr>
<td>mropdict</td>
</tr>
<tr>
<td>mrtopdict</td>
</tr>
<tr>
<td>mrclose</td>
</tr>
<tr>
<td>mrcldict</td>
</tr>
<tr>
<td>mrqdb</td>
</tr>
</tbody>
</table>

The following table lists the Routines used to control locking:

<table>
<thead>
<tr>
<th>Lock Routines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>mrlktab</td>
</tr>
<tr>
<td>mrlkrec</td>
</tr>
<tr>
<td>mrultab</td>
</tr>
<tr>
<td>mrulrec</td>
</tr>
</tbody>
</table>
The following table lists the Routines used to allocate and de-allocate memory:

### Allocate Space Routines

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrmkrec</td>
<td>Allocate space to store one record.</td>
</tr>
<tr>
<td>mrfrrec</td>
<td>Free space allocated by mrmkrec.</td>
</tr>
<tr>
<td>mrspv</td>
<td>Allocate space to store an attribute value.</td>
</tr>
<tr>
<td>mrfree</td>
<td>Free space allocated by mrspv.</td>
</tr>
</tbody>
</table>

The following table lists the Routines used to manipulate attribute descriptors:

### Attribute Descriptor Routines

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrngeta</td>
<td>Get a descriptor for a named attribute.</td>
</tr>
<tr>
<td>mrigeta</td>
<td>Get a descriptor for a numbered attribute.</td>
</tr>
<tr>
<td>mrganame</td>
<td>Find an attribute name given its descriptor.</td>
</tr>
</tbody>
</table>

The following table lists the Routines used to assign values to records:

### Filling Records Routines

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrputvs</td>
<td>Assign a value (external format) to an attribute in a record; do not terminate on failure.</td>
</tr>
<tr>
<td>mrputvi</td>
<td>Assign an integer to an attribute in a record; do not terminate on failure.</td>
</tr>
<tr>
<td>mrputi</td>
<td>Assign a value (internal format) to an attribute in a record; do not terminate on failure.</td>
</tr>
<tr>
<td>mrmptvs</td>
<td>Assign a value (external format) to an attribute in a record; terminate on failure.</td>
</tr>
<tr>
<td>mrmptvi</td>
<td>Assign an integer to an attribute in a record; terminate on failure.</td>
</tr>
<tr>
<td>mrsetnv</td>
<td>Assign a null value to an attribute in a record.</td>
</tr>
<tr>
<td>mrsetnr</td>
<td>Assign a null values to all attributes in a record.</td>
</tr>
</tbody>
</table>
Appendix 1. A List of MR Routines

The following table lists the Routines used to insert records into a table:

### Insert Routines

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrad</td>
<td>Insert a record; terminate on failure.</td>
</tr>
<tr>
<td>mradd</td>
<td>Insert a record; do not terminate on failure.</td>
</tr>
<tr>
<td>mraddend</td>
<td>Clean up after inserting records into a table.</td>
</tr>
</tbody>
</table>

The following table lists the Routines used to delete records from a table:

### Delete Routines

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrdel</td>
<td>Delete a record; terminate on failure.</td>
</tr>
<tr>
<td>mrtdel</td>
<td>Delete a record; do not terminate on failure.</td>
</tr>
<tr>
<td>mrtdelend</td>
<td>Clean up after deleting records from a table.</td>
</tr>
</tbody>
</table>

The following table lists the Routines used to update records in a table:

### Update Routines

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrcopyr</td>
<td>Copy attribute values from one record to another.</td>
</tr>
<tr>
<td>mrput</td>
<td>Update a record; terminate on failure.</td>
</tr>
<tr>
<td>mrtput</td>
<td>Update a record; do not terminate on failure.</td>
</tr>
</tbody>
</table>

The following table lists the Routines used to initialize retrieval of values from a table:

### Initialize Retrieval Routines

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrgetbegin</td>
<td>Associate qualifications with records for retrievals; terminate on failure.</td>
</tr>
<tr>
<td>mrtgtbegin</td>
<td>Associate qualifications with records for retrievals; do not terminate on failure.</td>
</tr>
<tr>
<td>mrsrtbegin</td>
<td>Associate qualifications with records for sorted retrievals; terminate on failure.</td>
</tr>
<tr>
<td>mrttsrbegin</td>
<td>Associate qualifications with records for sorted retrievals; do not terminate on failure.</td>
</tr>
</tbody>
</table>
The following table lists the Routines used to retrieve values from a table:

### Retrieval Routines

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrget</td>
<td>Retrieve a record from a table.</td>
</tr>
<tr>
<td>mrtget</td>
<td>Retrieve a record, report if record locked.</td>
</tr>
<tr>
<td>mrreget</td>
<td>Try again for the next record.</td>
</tr>
<tr>
<td>mrprev</td>
<td>Retrieve the previous record.</td>
</tr>
<tr>
<td>mrtprev</td>
<td>Retrieve the previous record, report if locked.</td>
</tr>
<tr>
<td>mrreprev</td>
<td>Try again for the previous record.</td>
</tr>
<tr>
<td>mrgetrec</td>
<td>Get a record referred to by a previously-found pointer.</td>
</tr>
<tr>
<td>mrgetend</td>
<td>Clean up after retrieving records from a table.</td>
</tr>
<tr>
<td>mrcontrol</td>
<td>Specify how many bytes of a particular bulk attribute value will actually be retrieved from a table.</td>
</tr>
<tr>
<td>mrcopyv</td>
<td>Assign the value (external format) of an attribute to an allocated space.</td>
</tr>
<tr>
<td>mrcopyi</td>
<td>Assign the value (internal format) of an attribute to a variable.</td>
</tr>
<tr>
<td>mrgetvs</td>
<td>Return an attribute value (external format).</td>
</tr>
<tr>
<td>mrgetvi</td>
<td>Return an attribute value (as an integer).</td>
</tr>
<tr>
<td>mrgeti</td>
<td>Return an attribute value (internal format).</td>
</tr>
<tr>
<td>mrgetptr</td>
<td>Find a pointer to a given record for later use.</td>
</tr>
<tr>
<td>mrgfunc</td>
<td>Find \texttt{count}, \texttt{max}, \texttt{min}, \texttt{sum}, or \texttt{avg} of an attribute; terminate on failure.</td>
</tr>
<tr>
<td>mrtgfunc</td>
<td>Find \texttt{count}, \texttt{max}, \texttt{min}, \texttt{sum}, or \texttt{avg} of an attribute; do not terminate on failure.</td>
</tr>
</tbody>
</table>

The following table lists the Routines used to perform comparisons between attribute values:

### Compare Routines

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrcompare</td>
<td>Find whether a variable is equal to, less than, or greater than an attribute value.</td>
</tr>
<tr>
<td>mrnullr</td>
<td>Find whether an entire record is null.</td>
</tr>
<tr>
<td>mrnullv</td>
<td>Find whether an attribute value is null.</td>
</tr>
</tbody>
</table>
Appendix 1. A List of MR Routines

The following table lists the Routines used to convert values between different data formats:

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrcvt</td>
<td>Convert a value from external format to file format.</td>
</tr>
<tr>
<td></td>
<td>(Shared buffer with other MR Routines.)</td>
</tr>
<tr>
<td>mrcvtin</td>
<td>Convert a value from internal format to file format.</td>
</tr>
<tr>
<td></td>
<td>(Shared buffer with other MR Routines.)</td>
</tr>
<tr>
<td>mrcvtv</td>
<td>Convert a value from external format to file format.</td>
</tr>
<tr>
<td></td>
<td>(Shared buffer with other mrcvt1.)</td>
</tr>
<tr>
<td>mrcvtv2</td>
<td>Convert a value from external format to file format.</td>
</tr>
<tr>
<td></td>
<td>(Shared buffer with other mrcvt12.)</td>
</tr>
<tr>
<td>mrcvti</td>
<td>Convert a value from internal format to file format.</td>
</tr>
<tr>
<td></td>
<td>(Shared buffer with other mrcvtv.)</td>
</tr>
<tr>
<td>mrcvti2</td>
<td>Convert a value from internal format to file format.</td>
</tr>
<tr>
<td></td>
<td>(Shared buffer with other mrcvtv2.)</td>
</tr>
</tbody>
</table>

The following table lists the Routines used to compare attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrqcon</td>
<td>Compare an attribute value with a constant; terminate on failure.</td>
</tr>
<tr>
<td>mrtqcon</td>
<td>Compare an attribute value with a constant; do not terminate on failure.</td>
</tr>
<tr>
<td>mrqrng</td>
<td>Compare an attribute value with a given range; terminate on failure.</td>
</tr>
<tr>
<td>mrtqrng</td>
<td>Compare an attribute value with a given range; do not terminate on failure.</td>
</tr>
<tr>
<td>mrqatr</td>
<td>Compare two attribute values; terminate on failure.</td>
</tr>
<tr>
<td>mrtqatr</td>
<td>Compare two attribute values; do not terminate on failure.</td>
</tr>
<tr>
<td>mrqmch</td>
<td>Match an attribute value with a constant; terminate on failure.</td>
</tr>
<tr>
<td>mrtqmch</td>
<td>Match an attribute value with a constant; do not terminate on failure.</td>
</tr>
<tr>
<td>mrqnul</td>
<td>Compare an attribute value with null; terminate on failure.</td>
</tr>
<tr>
<td>mrtqnul</td>
<td>Compare an attribute value with null; do not terminate on failure.</td>
</tr>
</tbody>
</table>
### Appendix 1. A List of MR Routines

<table>
<thead>
<tr>
<th>Routine</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrqseq</td>
<td>See if an attribute is equal to a string; terminate on failure.</td>
</tr>
<tr>
<td>mrqieq</td>
<td>See if an attribute is equal to an integer; terminate on failure.</td>
</tr>
<tr>
<td>mrqand</td>
<td>Perform an AND on any two of the above comparisons.</td>
</tr>
<tr>
<td>mrqor</td>
<td>Perform an OR on any two of the above comparisons.</td>
</tr>
<tr>
<td>mrqnot</td>
<td>Perform a NOT on any two of the above comparisons.</td>
</tr>
<tr>
<td>mrqlst</td>
<td>Qualify records in the given list of previously found pointers to records.</td>
</tr>
</tbody>
</table>

The following table lists the Routines used to control transactions:

### Transaction Routines

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrtrstart</td>
<td>Start a transaction.</td>
</tr>
<tr>
<td>mtrcancel</td>
<td>Cancel a transaction.</td>
</tr>
<tr>
<td>mtrcommit</td>
<td>Commit a transaction.</td>
</tr>
<tr>
<td>mtrsave</td>
<td>Set save point in transaction.</td>
</tr>
<tr>
<td>mtrrollback</td>
<td>Roll back to save point.</td>
</tr>
</tbody>
</table>

The following table lists the Routines used to build expressions:

### Expression Routines

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrebegin</td>
<td>Start building expression.</td>
</tr>
<tr>
<td>mreend</td>
<td>Finish building expression.</td>
</tr>
<tr>
<td>mreabort</td>
<td>Abort building expression.</td>
</tr>
<tr>
<td>mrqexpr</td>
<td>Put expression in qualification.</td>
</tr>
<tr>
<td>mrerun</td>
<td>Run expression and return result.</td>
</tr>
<tr>
<td>mrefree</td>
<td>Free expression descriptor.</td>
</tr>
<tr>
<td>mrerecattr</td>
<td>Add an attribute operand.</td>
</tr>
<tr>
<td>mrecons</td>
<td>Add a constant operand.</td>
</tr>
<tr>
<td>mreicvar</td>
<td>Associate a control variable with the most recent variable.</td>
</tr>
<tr>
<td>mreivar</td>
<td>Add a variable operand.</td>
</tr>
</tbody>
</table>
Appendix 1. A List of MR Routines

- **mrecvarg**: Add a type conversion operator.
- **mrefunc**: Add a function or operator.
- **mrenull**: Add a null operator.
- **mrgdtpar**: Get a data type descriptor.

The following table lists the Routines used to handle errors:

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrprterr</td>
<td>Print a suitable message if one of the mrt Routines fails; terminate calling program.</td>
</tr>
<tr>
<td>mrerrmsg</td>
<td>Retrieve last error message.</td>
</tr>
</tbody>
</table>

Product Profile 119
Appendix 2 - A List of Empress System Variables
This appendix contains summaries and descriptions of the following:

System Variables
Appendix 2 – A List of Empress System Variables

The following lists are *Empress* system variables:

<table>
<thead>
<tr>
<th>Path Related Variables</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSHELPPATH</td>
<td>Directory for <em>Empress</em> help files.</td>
</tr>
<tr>
<td>MSNLSDB</td>
<td>The National Language Support database.</td>
</tr>
<tr>
<td>MSTERMDB</td>
<td>The terminal definition database for <em>Empress</em> 4GL.</td>
</tr>
<tr>
<td>MSAPFMDB</td>
<td><em>Empress</em> 4GL form database.</td>
</tr>
<tr>
<td>MS4GLPSHEADER</td>
<td><em>Empress</em> 4GL Postscript header file.</td>
</tr>
<tr>
<td>MSKILLLOGDIR</td>
<td>Directory for <em>Empress</em> utility empkill log file.</td>
</tr>
<tr>
<td>MSCRAYQDELLOGDIR</td>
<td>Directory for <em>Empress</em> utility empqdel log file. This is for Cray only.</td>
</tr>
<tr>
<td>MSGUIDB</td>
<td>System directory for <em>Empress</em> GUI.</td>
</tr>
<tr>
<td>MSGUIDECOLORFILE</td>
<td>Directory for <em>Empress</em> GUI color definition.</td>
</tr>
<tr>
<td>MSSHAREDMEMORYDIR</td>
<td>Memory mapped file directory for the platform which uses mmap (memory mapping).</td>
</tr>
</tbody>
</table>
### General Topics Related Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSDATELIMIT</td>
<td>Preferred century for entering two digit dates.</td>
</tr>
<tr>
<td>MSDATEPIC</td>
<td>Date picture for reports and data entry.</td>
</tr>
<tr>
<td>MSDDLTRANS</td>
<td>When set, the Data Definition Language commands will be wrapped with transaction.</td>
</tr>
<tr>
<td>MSDISPLAYKATAKANA</td>
<td>For terminals that cannot support Half-Katakana characters. This is for <em>Empress</em> Japanese version only.</td>
</tr>
<tr>
<td>MSDOLLAR</td>
<td>Format for dollar data types.</td>
</tr>
<tr>
<td>MSEDITOR</td>
<td>Editor used by .zz command and <em>Empress</em> 4GL.</td>
</tr>
<tr>
<td>MSFILESOPEN</td>
<td>Maximum number of files that will be opened by <em>Empress</em>.</td>
</tr>
<tr>
<td>MSFORCEPLOCK</td>
<td>Causes a process to lock itself into memory, assuming that it has the appropriate privileges.</td>
</tr>
<tr>
<td>MSINDEXLISTCUTOFF</td>
<td>Percentage of the total number of records in a table, beyond which the index will no longer be used for retrievals.</td>
</tr>
<tr>
<td>MSINDEXLISTCUTOFFMIN</td>
<td>Minimum number of records that will be read from the indices regardless of the setting of MSINDEXLISTCUTOFF.</td>
</tr>
<tr>
<td>MSINDEXSORTCUTOFF</td>
<td>Percentage of the total number of records in a table, above which the index will be used for sorting the records.</td>
</tr>
<tr>
<td>MSKEEPNONPRINTCHAR</td>
<td>Flag to interpret non-printing characters.</td>
</tr>
<tr>
<td>MSLANG</td>
<td>Specifies the language in which you want <em>Empress</em> messages to be printed.</td>
</tr>
<tr>
<td>MSLINECONT</td>
<td>String used to continue lines.</td>
</tr>
<tr>
<td>MSLKCOMMITADDEND</td>
<td>Reduces the number of lock manager accesses between consecutive insertions.</td>
</tr>
<tr>
<td>MSLOCKPLAN</td>
<td>Provides lock usage information by <em>Empress</em>.</td>
</tr>
<tr>
<td>MSNLSCODESET</td>
<td>The name of the codeset appropriate to the keyboard/monitor you are using.</td>
</tr>
<tr>
<td>MSNULLVALUE</td>
<td>String to replace null values.</td>
</tr>
<tr>
<td>MSPAGER</td>
<td>System program for paging Query</td>
</tr>
</tbody>
</table>
## Appendix 2. A List of Empress System Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSPERMS</td>
<td>Access permissions for <em>Empress</em> output files.</td>
</tr>
<tr>
<td>MSPRINTER</td>
<td>System program used to send output to a printer.</td>
</tr>
<tr>
<td>MSQUERYPLAN</td>
<td>Provides indices information used by <em>Empress</em> when evaluating any search strategy before going to the database for record retrievals.</td>
</tr>
<tr>
<td>MSSHELL</td>
<td>System shell for shell escapes.</td>
</tr>
<tr>
<td>MSSORTBYPASS</td>
<td>Flag to skip locked records during sorting.</td>
</tr>
<tr>
<td>MSSORTSPACE</td>
<td>Byte limit on space used during sorting.</td>
</tr>
<tr>
<td>MSTHRESHOLDPFLSIZE</td>
<td>Maximum size of internal Process Free List before an automatic flush/merge is performed to the Global Free List.</td>
</tr>
<tr>
<td>MSTMPDIR</td>
<td>Directory where temporary files are created.</td>
</tr>
<tr>
<td>MSTMPFFX</td>
<td>Prefix used for temporary files.</td>
</tr>
<tr>
<td>MSVALSEP</td>
<td>String used to separate attribute values in dump files.</td>
</tr>
<tr>
<td>MSVERSIONMESG</td>
<td>Print <em>Empress</em> version on startup.</td>
</tr>
<tr>
<td>MSWRAPMARGIN</td>
<td>Controls line breaks and wrapping.</td>
</tr>
<tr>
<td>MSCASEINSENSITIVE</td>
<td>Forces Case Insensitivity.</td>
</tr>
<tr>
<td>MSNTMPFILECACHE</td>
<td>Defines a number of the caches used for temporary files associated with temporary tables used during retrieval tasks.</td>
</tr>
<tr>
<td>MSQUOTEDIDENTIFIERS</td>
<td>Allows quoted identifiers.</td>
</tr>
<tr>
<td>MSSELEXTFETCH</td>
<td>Forces backward compatibility for the usage of <em>Empress</em> mr routine call mrprev.</td>
</tr>
<tr>
<td>MSSELMAXROWS</td>
<td>Sets a maximum number of rows/records to return in <em>Empress</em> Interactive SQL for any SELECT statement.</td>
</tr>
<tr>
<td>MSSELTIMEOUT</td>
<td>Sets a number of seconds to wait for an SQL SELECT statement to execute before returning to the <em>Empress</em> Interactive SQL interface.</td>
</tr>
<tr>
<td>MSTMPFILECACHESIZE</td>
<td>Defines a size of the cache used for temporary files associated with</td>
</tr>
</tbody>
</table>
Appendix 2. A List of Empress System Variables

temporary tables used during retrieval tasks.

<table>
<thead>
<tr>
<th>Transactions Related Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variable</strong></td>
</tr>
<tr>
<td>MSTRANSCOMMENT</td>
</tr>
<tr>
<td>MSTRANSSYNC</td>
</tr>
<tr>
<td>MSTRANSTABLELOCK</td>
</tr>
<tr>
<td>MTRANSUFNBI</td>
</tr>
<tr>
<td>MTRANSUFNGFL</td>
</tr>
<tr>
<td>MSTRANSWARMPROTECT</td>
</tr>
</tbody>
</table>
### Locking Related Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSCFEXCLRETRY</td>
<td>Number of retries for exclusive access to the coordinator file.</td>
</tr>
<tr>
<td>MSCFEXCLSLEEP</td>
<td>Interval between retries for exclusive lock access to the coordinator file.</td>
</tr>
<tr>
<td>MSEXCLRETRY</td>
<td>Number of retries for exclusive access to lock file.</td>
</tr>
<tr>
<td>MSEXCLSLEEP</td>
<td>Interval between retries for exclusive lock access.</td>
</tr>
<tr>
<td>MSIAEXCLRETRY</td>
<td>Number of exclusive access retries for Interactive Interface.</td>
</tr>
<tr>
<td>MSIAEXCLSLEEP</td>
<td>Exclusive access retry interval for Interactive Interface.</td>
</tr>
<tr>
<td>MSIALOCKRETRY</td>
<td>Number of lock retries for Interactive Interface.</td>
</tr>
<tr>
<td>MSIALOCKSLEEP</td>
<td>Lock retry interval for Interactive Interface.</td>
</tr>
<tr>
<td>MSINDEXRETRY</td>
<td>Number of retries to lock an index.</td>
</tr>
<tr>
<td>MSINDEXSLEEP</td>
<td>Interval between retries to lock an index.</td>
</tr>
<tr>
<td>MSLOCKRETRY</td>
<td>Number of retries to actually make a lock entry.</td>
</tr>
<tr>
<td>MSLOCKSLEEP</td>
<td>Interval between retries to actually make a lock entry.</td>
</tr>
</tbody>
</table>

### Query Related Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MYSQLAUTOPAGE</td>
<td>Flag to send all command output to a paging program.</td>
</tr>
<tr>
<td>MYSQLCMDSAVE</td>
<td>Number of commands saved by history mechanism.</td>
</tr>
<tr>
<td>MYSQLCOUNT</td>
<td>Flag to display number records selected, updated, deleted.</td>
</tr>
<tr>
<td>MYSQLECHO</td>
<td>Controls commands echoing.</td>
</tr>
<tr>
<td>MYSQLONELINE</td>
<td>Indicates that a new line terminates a command.</td>
</tr>
<tr>
<td>MYSQLPROMPT1</td>
<td>Primary Empress prompt.</td>
</tr>
<tr>
<td>MYSQLPROMPT2</td>
<td>Secondary Empress prompt.</td>
</tr>
</tbody>
</table>
Appendix 2. A List of Empress System Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSQLTRANSACTION</td>
<td>Each command is a transaction and may be cancelled.</td>
</tr>
<tr>
<td>MSQLVARCHARS</td>
<td>Characters used in <em>Empress</em> variable notation.</td>
</tr>
</tbody>
</table>

### Select Output Formatting Related Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSQLGCHARWIDTH</td>
<td>Column width for non-parametric character values.</td>
</tr>
<tr>
<td>MSQLGDATEWIDTH</td>
<td>Column width for non-parametric date values.</td>
</tr>
<tr>
<td>MSQLGDECIMALWIDTH</td>
<td>Column width for non-parametric decimal values.</td>
</tr>
<tr>
<td>MSQLGFLOATWIDTH</td>
<td>Column width for non-parametric float values.</td>
</tr>
<tr>
<td>MSQLGINTEGERWIDTH</td>
<td>Column width for non-parametric integer values.</td>
</tr>
<tr>
<td>MSQLSELBOX</td>
<td>Puts a box around the output.</td>
</tr>
<tr>
<td>MSQLSELCOLSEP</td>
<td>Column separator.</td>
</tr>
<tr>
<td>MSQLSELGROUPCOLCROSS</td>
<td>Group separator crossing column.</td>
</tr>
<tr>
<td>MSQLSELGROUPSEP</td>
<td>Group separator.</td>
</tr>
<tr>
<td>MSQLSELHEAD</td>
<td>Flag to print/suppress headers.</td>
</tr>
<tr>
<td>MSQLSELHEADCOLCROSS</td>
<td>Heading crossing column.</td>
</tr>
<tr>
<td>MSQLSELHEADSEP</td>
<td>Separates headers from rows.</td>
</tr>
<tr>
<td>MSQLSELKEEPDUPLICATE</td>
<td>Flag to keep duplicate group values.</td>
</tr>
<tr>
<td>MSQLSELROWCOLCROSS</td>
<td>Row crossing column.</td>
</tr>
<tr>
<td>MSQLSELROWSEP</td>
<td>Row separator.</td>
</tr>
<tr>
<td>MSQLSELTRUNCATE</td>
<td>Flag to truncate long lines.</td>
</tr>
<tr>
<td>MSQLSELWARN</td>
<td>Print warning if function encounters null argument.</td>
</tr>
</tbody>
</table>
## On-Line Backup Related Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSCOORDTIMERFREQ</td>
<td>The time period (in seconds) within which a client must check in with the coordinator.</td>
</tr>
<tr>
<td>MSOLBBACKUPDEVICE</td>
<td>On-line backup device name.</td>
</tr>
<tr>
<td>MSOLBBLOCKSIZE</td>
<td>On-line backup device block size.</td>
</tr>
<tr>
<td>MSOLBRECOVERYLOG</td>
<td>On-line backup recovery log file.</td>
</tr>
</tbody>
</table>

## Shared Memory Related Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSMKDBSHMEM</td>
<td>Creates database with shared memory.</td>
</tr>
<tr>
<td>MSSHMLOCATION</td>
<td>System address where the shared memory segment will exist.</td>
</tr>
<tr>
<td>MSSHMPERMS</td>
<td>Access permissions for shared memory partitions.</td>
</tr>
</tbody>
</table>

## Dirty Read and Data Streaming Related Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSBULKSEGMENTSIZE</td>
<td>Memory size for transferring bulk data in small pieces.</td>
</tr>
<tr>
<td>MSVALIDATELEVEL</td>
<td>Behaviour of dirty read as a result of checksum.</td>
</tr>
<tr>
<td>MSVALIDATERETRY</td>
<td>Number of retries <em>Empress</em> will make to read an invalid record.</td>
</tr>
<tr>
<td>MSVALIDATESLEEP</td>
<td>Interval in seconds between retries to read invalid records.</td>
</tr>
</tbody>
</table>
### Database Utility Related Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSKILLRETRY</td>
<td>Number of times <code>empkill</code> attempts to send the termination signal specified by <code>MSKILLSIGNAL</code>.</td>
</tr>
<tr>
<td>MSKILLSLEEP</td>
<td>Interval between retries <code>empkill</code> attempts to send the termination signal specified by <code>MSKILLSIGNAL</code>.</td>
</tr>
<tr>
<td>MSKILLSIGNAL</td>
<td>Signal for <code>empkill</code> to terminate.</td>
</tr>
</tbody>
</table>

### Server/Internet Related Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSSERVERHEARTBEAT</td>
<td>Time period for checking Database Server idle time.</td>
</tr>
<tr>
<td>MSSERVERNETTYPE</td>
<td>Server access type.</td>
</tr>
<tr>
<td>MSSERVERTERSELOG</td>
<td>Compact format for server log file information.</td>
</tr>
<tr>
<td>MSINETREPLYTIMEOUT</td>
<td>Timeout value for the client awaiting a high-level response.</td>
</tr>
<tr>
<td>MSINETREPLYRETRY</td>
<td>Number of retries for the client awaiting a high-level response.</td>
</tr>
<tr>
<td>MSINETMESSAGETIMEOUT</td>
<td>Timeout value for the client’s message handler awaiting acknowledgement.</td>
</tr>
<tr>
<td>MSINETMESSAGERETRY</td>
<td>Number of retries for the client’s message handler awaiting acknowledgement.</td>
</tr>
<tr>
<td>MSINETPACKETTIMEOUT</td>
<td>Timeout value for the client’s packet handler awaiting acknowledgement.</td>
</tr>
<tr>
<td>MSINETPACKETRETRY</td>
<td>Number of retries for the client’s packet handler awaiting acknowledgement.</td>
</tr>
<tr>
<td>MSGETHOSTTIME</td>
<td>Timeout value for retrieving host information.</td>
</tr>
<tr>
<td>MSHOSTNAME</td>
<td>System host name.</td>
</tr>
</tbody>
</table>

### Operating System Related Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSGETPW</td>
<td>Use <code>getpw</code> instead of <code>getwuid</code></td>
</tr>
</tbody>
</table>
Appendix 2. A List of Empress System Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSGCWDSIGCHLDRESET</td>
<td>Resets the signal for some operating system call getcwd.</td>
</tr>
<tr>
<td>MSCRAYQDELSLEEP</td>
<td>Number of seconds empqdel waits before checking whether a request has died. This is for Cray system only.</td>
</tr>
<tr>
<td>MSCRAYQDELRETRY</td>
<td>Number of times empqdel attempts to send the termination signal. This is for Cray system only.</td>
</tr>
<tr>
<td>MSCRAYQDELSIGNAL</td>
<td>Number or symbolic name of the signal for empqdel. This is for Cray system only.</td>
</tr>
<tr>
<td>MSMIGRATERETRY</td>
<td>Number of retry to access migrated file. This is used on Cray system only.</td>
</tr>
<tr>
<td>MSMIGRATESLEEP</td>
<td>Time interval for the retries to access migrated file. This is used on Cray system only.</td>
</tr>
</tbody>
</table>

Report Writer Related Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSMWNULLOK</td>
<td>Suppress error on printing null.</td>
</tr>
<tr>
<td>MSPAGELENGTH</td>
<td>Default page length for reports.</td>
</tr>
<tr>
<td>MSPAGEWIDTH</td>
<td>Default page width for reports.</td>
</tr>
</tbody>
</table>

Empress Internal Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSSAVECWD</td>
<td>Save the current working directory value.</td>
</tr>
<tr>
<td>MSBUFFERRECORDFACTOR</td>
<td>Controls the number of records for the read ahead buffering action.</td>
</tr>
<tr>
<td>MSMAXPROCS</td>
<td>Maximum number of processes which can access the lock manager in a lock file.</td>
</tr>
<tr>
<td>MSFILELOCKNBUCKETS</td>
<td>Number of hash buckets for a lock manager in a lock file.</td>
</tr>
<tr>
<td>MSFILELOCKNLOCKS</td>
<td>Maximum number of locks for a lock manager in a lock file.</td>
</tr>
</tbody>
</table>
### Appendix 2. A List of Empress System Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSNRECORDCACHE</td>
<td>Cache size in terms of the number of records.</td>
</tr>
<tr>
<td>MSLICENCE</td>
<td>License information.</td>
</tr>
</tbody>
</table>

#### Network Server Related Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSUSERAUTHCONFIGFILE</td>
<td>Name of user authorization configuration file.</td>
</tr>
<tr>
<td>MSNETSERVERCONFIGFILE</td>
<td>Name of network server configuration file.</td>
</tr>
<tr>
<td>MSNETTYPECONFIGFILE</td>
<td>Name of network type configuration file.</td>
</tr>
<tr>
<td>MSCONFIGFILEPATH</td>
<td>Path for Network Server Related Configuration Files.</td>
</tr>
</tbody>
</table>

#### Replication Related Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSREPLOCKRETRY</td>
<td>Number of retries to obtain a lock on a master table during replication synchronization.</td>
</tr>
<tr>
<td>MSREPLOCKSLEEP</td>
<td>Interval between retries to obtain a lock on a master table during replication synchronization.</td>
</tr>
<tr>
<td>MSREPENABLUUPDATE</td>
<td>Internal Variable that enables a Replicate Table to be updateable.</td>
</tr>
<tr>
<td>MSREPPURGEINTERVAL</td>
<td>MSREPPURGEINTERVAL sets the maximum amount of time in hours which will be allowed for a replication table to be synchronized with its dependent replication replicates, before a deleted record is purged automatically.</td>
</tr>
</tbody>
</table>
Appendix 3 - Empress White Paper: The Concept of I=MC²
This appendix contains summaries and descriptions of the following:

Empress White Paper on the Concept of I=MC²
Appendix 3 - White Paper: The Concept of $I=MC^2$

$I = MC^2^{™}$

The Concept

The Empress Data Management System as an Information Management Component

by Njai Wong

A White Paper
Introduction

The Empress Data Management System was designed as the two tiered Information Management Component (IMC) of application software.

$I = M C^2$ represents the concept of a two tiered Information Management Component where:

- $I$ represents Information
- $M$ represents the Empress RDBMS
- $C^2$ represents the Two Tiered Component

The Empress Data Management System works as a two tiered component. The first tier works as an extension of the Operating System file system. The second tier works as an embeddable application database subsystem plug-in. Together, the two tiers create the Force that governs universal data. This Force can process data into information and make it useful.

Empress Data Management System Backgrounder


The Empress Data Management System is powerful and cost effective. The power is shown by its speed, small footprint, adaptability, scalability and extensibility. Its cost effectiveness is shown by its portability, ease of use, ease of integration and ease of maintenance.
The *Empress* Data Management System product differentiators are revealed in its design:

- Designed for embeddable application database subsystem plug-ins
- Designed for handling any object the computer can store
- Designed for handling large volumes of data
- Designed for building knowledge-based databases
- Designed for distributed database and programming processing
- Designed for both disk and in memory database management
- Designed for eXtreme low maintenance

**Tier 1 Component: Operating System Extensions**

One of the basic tasks of the operating system is to interface with storage devices such as disks, allocate storage, keep track of files and directories, provide the input and output of data from disk, and even provide data management functionality.

**Definition**

"Operating System: An integrated collection of routines that service the sequencing and processing of programs by a computer.  
Note: An operating system may provide many services, such as resource allocation, scheduling, input/output control, and data management. Although operating systems are predominantly software, partial or complete hardware implementations may be made in the form of firmware."\(^1\)

In general, an Operating System (O/S) provides users with a large and unified collection of utilities and file handling tool. The *Empress* Data Management System provides an integrated toolkit for information management to complement the standard Operating System facilities and extend the computer resources available to the user.

The *Empress* Data Management System uses the O/S file system as its native file system while adopting the O/S operational environment as its standard environment. Together with its open architecture, the *Empress* Data Management System becomes a functional extension of the O/S file system. *Empress* thus provides a data management system with speed, performance and small footprint together with the power of the O/S.
By adopting the O/S file system as its native file system and the O/S operating environment as its standard environment, the Empress Data Management System interfaces seamlessly with the operating system. For example Empress:

- works with multiple databases on the same disk
- supports Mapped Files, Juke Boxes, RAMDISK, RAID, HIPPI, etc.
- does not impose limits but inherits them from the O/S
- automatically interchanges 32 bit and 64 bit data
- automatically allocates Main Memory, Disk and Processors
- tables can reside on any directory, file system or node
- link with other libraries to make them available to developers
- multiple I/O - output of Empress can be input to another program
- is an integral part of the application
- requires low system resources to power the application
- uses the power of the O/S for speed, performance and small footprint

Tier 2 Component: Application Database Plug-in

A complete application consists of hardware and the following software components: operating system, application logic, application input/output (application data) and some form of data management system. The Empress Data Management System can be configured as an application plug-in allowing seamless integration with the whole application system.
An ideal database management system should:

- be an integral part of an application
- need no more administration than an application data subsystem
- need no more system resources than an application database subsystem

Most commercial database management systems cannot seamlessly integrate with applications, need dedicated database administrators and require heavy system resources.

The Empress Data Management System was designed as an embeddable application database subsystem plug-in. Multiple databases can be created at the application developer's specified locations, on both local and remote machines. With Empress, the application database subsystem is under the control of the application developer and becomes an integral part of a complete application. For a developer, this is what a database management system should be.

Information Management of Any Kind of Data

The Empress Data Management System was designed to handle any information the computer can store. This includes anything from name, address, phone number to JPEG images to compiled C++ programs to MP3 files to Meta data about application programs. This allows developers to create event driven, expert, knowledge and rule based systems. The result is a radically different paradigm for a database system where algorithms and visualization are placed "next" to user data. This new paradigm unites the executable code, user data and its meaning under the umbrella of the term "database".
Application Data

The Empress Data Management System provides a wide variety of data types, enabling users to choose the data type most suited for the type of data being stored and the application. These data types are:

- Character data
- Variable length text data (dynamic space allocation)
- National Language Support for character and text data
- Date data (9 external formats)
- Time data (9 external formats)
- Micro second time data
- Decimal data
- Currency data
- Real numbers
- Floating point numbers
- Double precision floating point numbers
- Long, regular and short integers
- Binary data of any kind
- User interpreted binary data with user defined functions

Meta Data

A good example of using relational tables to store Meta data is the Empress database data dictionary. The Empress database data dictionary consists of several relational tables, which hold information about database tables and attributes. These tables can be accessed like any other tables that hold application data. However, for faster retrieval, all this information is compiled and stored in one binary (bulk) attribute dict_comp in the sys_dictionary table:

Table: sys_dictionary
Attributes:

- dict_creator nlscharacter(31,1,0)
- dict_tabname nlscharacter(32,1,0)
  
  Not Null
- dict_comptime time(1)
- dict_comp bulk(20,0,1024,1)

The same methodology can be used for applications that need to store data for both user readable descriptors and computer readable compiled code.
Information: Application Programming Logic as Data

Application programming logic can be stored both in the *Empress* database tables and in the database schema. Application logic that executes depending on certain data is a candidate for storing in database tables. Application logic that acts or reacts to selected data is a good candidate for storing in the database schema such as triggers and stored procedures. The following sections provide examples of each.

Example 1: Application Programming Logic in Tables

The *Empress* 4GL and GUI are both examples of application development environments that allow developers to create event-driven applications. The following shows how the *Empress* 4GL is structured within the database. All information about 4GL applications is stored in several tables in the database. When the developer creates a form, this form is stored in the database table with 3 attributes:

**Table: sys_4gl_form**

Attributes:

- name: nlschar(32,1) Not Null
- compile: bulk(20,471,512,1) Not Null
- application_name: nlschar(32,1) Not Null

The form is compiled into a user interpreted data structure and stored as binary (bulk) data in the attribute "compile".

Fields are associated with a form. Fields allow data to be input and displayed. The information about a field is also stored in a database table:

**Table: sys_4gl_field**

Attributes:

- name: nlschar(32,1) Not Null
- application_name: nlschar(32,1) Not Null
- window_name: nlschar(32,1) Not Null
- enter_script: nlstext(20,0,64,1)
- exit_script: nlstext(20,70,64,1)
- mode: char(1,1)
- data_type: nlschar(32,1)
- next: nlschar(32,1)
- previous: nlschar(32,1)
- number: integer Not Null
- next_number: char(5,1)
- previous_number: char(5,1)
enter_script and exit_script store 4GL logic (programs) which are executed when the field is entered or exited. In a similar fashion, database tables are defined for other objects to keep track of and provide meaning for:

- Initialization for the enter script clean up for the exit script
- Size of each window
- Definition of each "form" that is associated with each window
- Logic associated with the input device
- Definitions of function keys
- Parameterization of mouse functions
- Definition of Braille devices

All information about the application is compiled into one attribute value, in one record, in one table (sys_4gl_compile) which is then linked (sys_4gl_link) with associated libraries. When the user calls the application, the attribute value in compile that matches the application name in the sys_4gl_link table is executed.

Example 2: Application Programming Logic in Schema

A database contains information of hurricane watches and warnings on tropical cyclones over the Atlantic, Caribbean, Gulf of Mexico and the Eastern Pacific over the past 20 years. Researchers work with this data to improve hurricane forecasting techniques. To assist this process, a Java function was written that returns "true" if a rectangle (geographic area under consideration) and a circle (the hurricane) intersect.
Storing this piece of programming logic in the *Empress* database schema turns it into "reusable code" which can be shared by many applications. Below are the table definitions and the interface definition for the JAVA program:

```sql
CREATE TABLE Storm {
    key INTEGER  NOT NULL,
    name CHARACTER (20,1) NOT NULL;
}

CREATE TABLE Storm_data {
    key LONGINTEGER NOT NULL,
    longitude LONG FLOAT NOT NULL,
    latitude LONG FLOAT NOT NULL,
    radius LONG FLOAT NOT NULL,
    category INTEGER NOT NULL,
    time_value TIME (2) NOT NULL;
}
```

For the JAVA program called *box_circle_intersection*, the interface definition is:

```java
GLOBAL_SHARED_FUNC msbool
    box_circle_intersection
    (double longitude,
    double latitude,
    double radius,
    double long1,
    double lat1,
    double long2,
    double lat2)
```

**Input Parameters:**
- longitude: longitude of storm eye
- latitude: latitude of storm eye
- radius: radius of storm
- long1: longitude of Northwest Corner
- lat1: latitude of Northwest Corner
- long2: longitude of Southeast Corner
- lat2: latitude of Southeast Corner

**Exports:**
- returns true if within region

Once this program is stored in the *Empress* database schema, it can be used by all the *Empress* interfaces. These include interactive, dynamic and static SQL, C, C++ and FORTRAN, ODBC, JDBC, Report Writer, etc.
The SQL statement below lists the names of hurricanes that occurred in the given geographical region near Florida between January 1, 1985 and December 31, 1999:

```
SELECT  DISTINCT 
name
FROM  Storm
WHERE  key  IN  (SELECT  DISTINCT  key
FROM  Storm_data
WHERE  time_value  BETWEEN
19850101  AND  19991231
AND  box_circle_intersection
(longitude,  latitude,  
radius,  140.0,  24.5,
75.0,  50.0)

);  
```

The parameter values of `longitude`, `latitude`, and `radius` for the function `box_circle_intersection` are obtained from the `Storm_data` table. This value changes each time a record is obtained from the outer select statement.
Conclusion

The *Empress* Data Management System with its unique two tiered component architecture provides the developer with the "can do" flexibility of a developer created database plug-in without the inconvenience and overhead of traditional databases. The *Empress* Data Management System not only deals with the storage and retrieval of data but also programming logic and user defined functions that turn data into information. The resulting application with the *Empress* Data Management System works seamlessly resulting in a small footprint, efficient and robust information application that requires virtually no maintenance.
References


For a more detailed description of Example 2 dealing with Hurricanes and specified geographic locations as well as more details how the user can extend the Empress Database Management System, please see:


For a more detailed look at the Empress Database Management System Datatypes, the Data Dictionary, storing compiled code, storing other binary data, adding persistent stored modules and user defined functions, please see:


For a more detailed look at the Empress Database Management System Distributed Database capability including Peer to Peer, please see:

Appendix 4 - Empress White Paper: Securing Data at Rest – Data Encryption
Appendix 4 - White Paper: Securing Data at Rest

Securing Data at Rest:
Database Encryption Solution using Empress RDBMS

By: Srdjan Holovac
© Empress Software Inc.

Contents

Introduction

Terminology

Different Concepts of Empress Database Encryption Solution
Implementing Encryption outside Empress RDBMS
Implementing Encryption inside Empress RDBMS

How It Works
Encrypting Binary Large Objects/ Character Large Objects
Encrypting Indexed Data
Type of Encryption
Key Management
Cipher Key Verification
Key Rotation/ Key Change
Unit of Encryption

One Key vs. Multiple Key Considerations

Performance and Resource Consumption Considerations
Future Developments
In Summary
Literature
Introduction

Importance of security

Enforcing data security is a top priority for both governments and businesses worldwide. Recent legislation set new standards for protecting customer information, such as standards for the security of medical records and standards for the financial industry regarding the privacy and security of customers’ personal financial information. How can this confidential data be protected? Organizations may protect data through data security approaches that account for host, application or network security, to mention only a few. New technology advances, including encryption, offer significant security value for this purpose.

Protecting data at rest

This paper focuses on a security solution for protection of data at rest, specifically protection of data that resides in databases. Most databases are deployed and stored in some kind of a persistent storage device such as a disk. Periodically, even in-memory databases have the need to backup data, hence data could end up in a persistent storage device in plaintext. Many embedded devices that contain an embedded database hold sensitive data that must be protected.

Encryption, the process of disguising data in such a way to hide its substance, is a very effective way to achieve security for data at rest. Implementation of a database encryption strategy raises several important factors that must be taken into consideration including: Should the encryption be performed inside the database engine, in the application where the data is generated or in a hardware device? Should encryption keys be kept inside the database or somewhere else where it is more secure? Should the granularity of encrypting data be applied to a database, a table or a column level? What are performance and the size considerations of different approaches?

This paper describes the implementation of the encryption solution using Empress RDBMS, summarizes the benefits of this solution for users, and discusses the issues mentioned above.
Terminology

The definitions of the basic terms are simplified for the usage in this paper. The consulted terminology sources were: [PKCS], [FIPS] and [BRUCE] (see Bibliography).

Encryption: The process of disguising data in such a way to hide its substance.

Cipher (Cryptographic Algorithm): The mathematical function used for encryption and decryption.

Cipher Key (Cryptographic Key, Encryption Key): A parameter used in conjunction with a cipher that determines:
- the transformation of plaintext data into ciphertext data,
- the transformation of ciphertext data into plaintext data.

Plaintext (Cleartext): A block of data that has not been encrypted.

Ciphertext: A block of data that has been encrypted.

Decryption: The process of transforming ciphertext back into plaintext.

Padding: A string, typically added when the plaintext block is short. For example, if the block length is 4 bytes and the cipher requires 16 bytes, then 12 bytes of padding must be added. The padding string may contain zeros, alternating zeros and ones, or some other pattern.
Figure 1. Encryption and Decryption

Figure 1 describes the process of encrypting and decrypting data. There are two general types of key-based algorithms: public-key and symmetric. In the case of most symmetric ciphers, the encryption and decryption keys are the same (as shown in Figure 1). Empress utilizes symmetric ciphers for its encryption solution.

Different Concepts of Empress Database Encryption Solution

The Empress database encryption solution is based on performing the encryption/decryption process either:

A) Outside the RDBMS by using a hardware device/appliance (Figure 2); Empress RDBMS can be configured with DataSecure appliances, dedicated hardware systems provided by Ingrian Networks [INGR]);

or

B) Inside the RDBMS (pure software solution) using a Security Library that contains an encryption algorithm (Figure 3); Empress RDBMS can be configured with libcrypt, a general purpose library that contains various cryptographic algorithms [GNUPG].
Implementing Encryption outside Empress RDBMS

Empress RDBMS incorporates C API calls from the Public Key Cryptography Standard (PKCS #11) that interface with the Security Adapter inside its Database Engine so that the encryption/decryption process is hidden from users [PKCS].

PKCS is a suite of specifications developed by RSA Security in conjunction with industry, academic, and government representatives. PKCS #11 is the specification for the cryptographic token interface standard, defining a technology-independent programming interface for cryptographic applications.

Using the PKCS #11 standardized approach Empress RDBMS is effectively integrated with the Ingrain Security Adapter – NAE PKCS #11 Provider [NAE]. Ingrian NAE Provider initiates encrypt and decrypt operations inside the Ingrain Security Device/Appliance, such as DataSecure. Both cipher key and cipher are contained inside the Security Device. DataSecure offers support for leading, standards-based ciphers: AES, 3DES, RSA and others.

Figure 2 describes the overall concept and placement of the Security Adapter in different Empress application scenarios. Empress standalone applications and Empress utilities, such as empsql or empclean, are directly linked to the Security Adapter. In the client-server scenario, such as the ODBC and JDBC applications in Figure 2, the Security Adapter is linked to the Empress Connectivity Server.

![Figure 2. Concept A: Database Encryption Concept with a Security Device](image-url)
Implementing Encryption inside Empress RDBMS

Figure 3 describes the second concept of adding the encryption capability to the Empress RDBMS. This is a pure software solution that involves a Security Library, which contains a cipher. Empress RDBMS is effectively integrated with libgcrypt, which performs data encryption and decryption. A cipher key is held either in a protected place in the file system, application/process environment or with a user.

Empress standalone applications and Empress utilities, such as empsql or empclean, are directly linked to the Security Library. In the client-server scenario, such as the ODBC and JDBC applications in Figure 3, the Security Library is linked to the Empress Connectivity Server.

Figure 3. Concept B: Database Encryption Concept with a Security Library
How It Works

The following are the basic postulates regarding adding encryption to the Empress RDBMS. They apply to both concepts described above.

The encryption is done on a column level. Users have the capability to define which columns are to be encrypted.

Let's assume a scenario where the database table **customer** has four columns **cust_no**, **name**, **ssn** and **address**, where customer number **cust_no** and social security number **ssn** have to be encrypted. To create such a table in an Empress database one would use the SQL CREATE TABLE command:

```sql
CREATE TABLE customer (
    cust_no LONGINTEGER NOT NULL ENCRYPTED,
    name CHAR(20),
    ssn CHAR(9) ENCRYPTED,
    address TEXT);
```

Since the column **cust_no** requires being searchable, an index has to be created too:

```sql
CREATE UNIQUE INDEX customer_index ON customer(cust_no);
```

Empress RDBMS will encrypt data for the columns **cust_no** and **ssn** that need encryption and decrypt data from those columns when the application needs it.

User applications that access table **customer** need NO changing. The same scenario works for all interfaces that Empress offers and also for Empress utilities.

Furthermore, users are given the ability to toggle between an encrypted and an unencrypted form by altering database schema, changing the column that needs to be or not to be encrypted. For example:

```sql
ALTER TABLE customer CHANGE ssn NOT ENCRYPTED;
```

Or to define the encryption on the column again:

```sql
ALTER TABLE customer CHANGE ssn ENCRYPTED;
```

One ALTER command can be issued in order to define encryption on multiple columns at once.

Users do NOT have to change the data type or the size of the encrypted column.
Unit of Encryption

How much data must be encrypted to provide security? Empress encryption provides flexibility to accommodate for the varying levels of encryption granularity. Users do not have to encrypt the whole database or the whole table.

Encryption is done on a column level in the database table.

Empress groups all encrypted columns in a table record.

Padding is done on the group of encrypted columns instead of padding on every encrypted column.

An Empress database is a directory in a file system. Users can have multiple databases on the same machine and switch databases as needed.

The unit of encryption varies depending on the type of file. Types of files that contain table column data in an Empress database are: main data (.rel) files, overflow data (.dtf) files, index data (.ix) files, transaction logs, recovery logs, backup files and temporary files.

File headers are not encrypted, as they contain no user data.

**Figure 4** illustrates an example scenario that describes encrypted parts and non-encrypted parts of the files related to the table *customer* in the Empress database.
There will be NO data from encrypted columns stored on the disk in plaintext.

Empress allows encryption on any column data type available in Empress RDBMS (e.g. CHAR, TEXT, BULK, INTEGER, LONGINTEGER, REAL, DOUBLE PRECISION, DECIMAL, DATE, TIME, MICROSECONDTIMESTAMP, etc.).
Encrypting Binary Large Objects/ Character Large Objects

Particular data types require more work to encrypt or decrypt data. Those kinds of data types are Empress BULK (for storing Binary Large Objects – BLOBs) and Empress TEXT and NLSTEXT (for storing Character Large Objects). The amount of data stored in the columns of these data types is expected to be large. Empress RDBMS does not impose limits on the size of those objects, eg: forcing users to split data into chunks when dealing with large volumes of binary or text data.

Encrypting Indexed Data

Users have the ability to create indexes on the encrypted columns of any Empress data type that can be normally indexed. Empress has designed encryption in its indexes in such a way to make indexes usable for all kinds of searches, not only for equality searches. Hence, there is no difference in the usability of indexes for searches on encrypted or unencrypted columns.

Type of Encryption

For encrypting and decrypting data, symmetric cipher keys are used.

Various public domain ciphers, including AES are standard with Empress and are user-selectable.

Empress also provides the means to embed user-defined ciphers.

The recommended cryptographic algorithm is Advanced Encryption Standard (AES). AES uses one of the 3 key lengths: 128, 192 and 256. The larger the key length the more computation it requires, and the greater security it provides.

All data in encrypted columns in a database is encrypted with the same cipher.

All the encrypted columns in a database are encrypted with one cipher key

Encryption is specified for a database when it is created. The cipher is selected at the time of creation and the selection is stored in the database. The cipher and the key should not be changed thereafter except by using the special Empress Key Rotation (Key Change) Utility.

Key Management

The Empress RDBMS engine has the basic need to obtain a cipher key for a given database. One of the essential key management questions is: Where should a cipher key reside? Some options are to store a cipher key: in a database, outside of a database in persistent storage (e.g. access-protected files), in memory, or prompt for it each time encryption is needed.

An additional option for exceptionally strong protection is to store a cipher key in the hardware security device (see Concept A). All cipher keys are stored in the Ingrian DataSecure Appliance.
In order to encrypt/decrypt data in the Empress database, Empress RDBMS must obtain **key info** (i.e. user name, password, key name) to pass to the Ingrian DataSecure Appliance in order to authenticate itself and gain access to the requested cipher key.

In the **Concept B**, Empress RDBMS must obtain a cipher key to pass to a Security Library.

The following options address the need for both concepts.

1) The MS (Empress) variable MSCIPHERKEYINFO can be set to <database, key info> pairs (**Concept A**) or to <database, key> pairs (**Concept B**). Hence, key info or key resides in memory (i.e. environment of the Empress Application/Utility/Server).

2) The MS variable MSCIPHERKEYINFOFILE can be set to the name of a file containing either key info (**Concept A**) or cipher key (**Concept B**). The file is called **credentials** file.

3) An additional very secure option is for a user to input key info or a key itself each time Empress application/utility is restarted.

**Cipher Key Verification**

A means of verifying that a cipher key is correct for a database is highly desirable so that meaningless data is not produced with an incorrect cipher key.

**Concept A**

During database creation, key info is encrypted with the cipher key it relates to and stored in the database. Thereafter, a given key info is encrypted with the cipher key it relates to and compared to the copy in the database to verify that it is correct for that database. If the key info is incorrect, a sleep period of several seconds is enforced before reporting the error.

**Concept B**

During a database creation, the cipher key is encrypted with itself and stored in the database. Thereafter, a given cipher key is encrypted with itself and compared to the copy in the database to verify that it is correct for that database. If the key is incorrect, a sleep period of several seconds is enforced before reporting the error.
Key Rotation/ Key Change

Empress provides a utility for off-line key rotation. The utility is callable via a stand-alone interface and may be used for key rotation, either periodic or as needed.

The assumption is made that the database needs to be off-line in order to re-encrypt all of the sensitive data within the database with a new cipher key. The following steps occur:

1. A second key is generated in the Security Device or in the credentials file.
2. A utility reads the encrypted data from the database, decrypts the information using the first key, encrypts with the second key, and writes the new encrypted data back to the database.
3. The key parameters are updated in the database to reflect the usage of the second key.

One Key vs. Multiple Key Consideration

“One Key”

Pros:
- Uses less C API calls when communicating with the Security Device
- Efficient solution. Requires padding only once on a group of encrypted columns

Cons:
- Less flexible solution

“Multiple Key”

Pros:
- Provides flexibility
- Users need the ability to associate a different key for different columns in database tables

Cons:
- Introduces additional complexity
- Requires padding on every encrypted column
- Greater increase of the size of a database table

Performance and Size Considerations

What is an acceptable trade-off between data security and application performance?

The Empress encryption strategy includes multiple design and optimization consideration in order to ease the trade-off between data security and application performance.

Empress encryption initiates encrypt/decrypt API calls inside the Empress Engine. This preferable design offers a more efficient solution and requires much less overhead than other design alternatives such as via stored procedures/triggers.
Size Considerations

Furthermore, as an additional optimization, Empress groups columns that are to be encrypted at the record level to optimize the encryption process, thus decreasing the number of times it has to encrypt/decrypt data. See Figure 5 for sample Internal Empress Record Layout Optimization.

![Figure 5. Internal Empress Record Layout Optimization](image)

Empress encryption solution also saves space. Space is an issue with column encryption because, in general, encrypted columns are larger than unencrypted columns.

For example, the cust_no column of the data type LONGINTEGER and 4 bytes in size might need 12 bytes of padding.

The overhead in an encrypted database size compared to the unencrypted database size is largely dependant on the cipher used. Let’s assume that cipher requires padding to the nearest 16 bytes. Hence, a record of 508 bytes would have to be extended to 512 bytes, making the overhead insignificant.
This internal optimization with grouping of encrypted columns is not visible to users. Hence, the original order of table columns is preserved. If for example, a user displays a table definition or retrieves data from a table, the user still get the original column order:

1* display customer all;

*** Table: customer ***

Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>cust_no</td>
<td>longinteger</td>
<td>Not Null Encrypted</td>
</tr>
<tr>
<td>name</td>
<td>character(20,1)</td>
<td></td>
</tr>
<tr>
<td>ssn</td>
<td>character(9,1)</td>
<td>Encrypted</td>
</tr>
<tr>
<td>address</td>
<td>text(20,1024,1024,1)</td>
<td></td>
</tr>
</tbody>
</table>

Creator:  srdjan

Indices:  UNIQUE BTREE customer_index ON (cust_no)

...  

2* select from customer;

  cust_no  name  ssn  address

...

**Future Developments**

**Multiple Key Approach**

The current security strategy is to use a single cipher key for each database. This simplifies key management and allows for performance optimizations when Empress issues encrypt/decrypt calls.

A future solution will be implemented where a key per database column can be specified. This will allow users to tighten their overall security policy when encrypting additional data types.

**On-Line Key Rotation Utility**

The current Empress Key Rotation Utility is an off-line tool where the database has to be quiescent. Empress plans to upgrade this utility with the option to perform an on-line key rotation where the database can remain available to applications. The on-line process will perform a similar re-encryption process as the off-line tool, updating the new encrypted data along with the meta information that indicates which key has been used to encrypt a specific field.
In Summary

In summary, the main benefits for implementing the encryption solution using Empress RDBMS are emphasized:

- Secure all database data. A solution for protecting user data in a database including protection of all logs and backup files.

- Eliminate the potential for lost data that could be read by another party and all of the serious ramifications that accompany it, including bad press, loss of customers, government intervention.

- An efficient security solution. Insignificant performance overhead when performing the encryption/decryption process inside the RDBMS (pure software solution) and a small performance overhead when performing the encryption/decryption process outside the RDBMS by using a hardware device/appliance. Furthermore, Empress has implemented additional provisions to keep the size of the database increase minimal.

- No need for application code changes. This solution does not impact already written applications that use the data in an unencrypted Empress database.

- No need for adding external provisions in the database to accommodate encryption such as stored procedures, triggers, views, etc. The Empress solution painless for users who choose to convert their non-secure database solution to a secure one.

Literature


[PKCS] PKCS #11 v2.11: Cryptographic Token Interface Standard, RSA Laboratories, Revision 1 — November 2001

[INGR] www.ingrian.com


<table>
<thead>
<tr>
<th>Topic</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Empress SQL</td>
<td>63</td>
</tr>
<tr>
<td>aix</td>
<td>53</td>
</tr>
<tr>
<td>AIX</td>
<td>53</td>
</tr>
<tr>
<td>ansi</td>
<td>24</td>
</tr>
<tr>
<td>Application Data</td>
<td>138, 140</td>
</tr>
<tr>
<td>Application Development</td>
<td>44, 45</td>
</tr>
<tr>
<td>Application Programming Logic as Data</td>
<td>141</td>
</tr>
<tr>
<td>Application Programming Logic in Schema</td>
<td>142</td>
</tr>
<tr>
<td>Application Programming Logic in Tables</td>
<td>141</td>
</tr>
<tr>
<td>ascii</td>
<td>76</td>
</tr>
<tr>
<td>Batch SQL</td>
<td>11, 58, 91, 101</td>
</tr>
<tr>
<td>Bluecat</td>
<td>53</td>
</tr>
<tr>
<td>bulk</td>
<td>71, 76</td>
</tr>
<tr>
<td>Bulk Chunks</td>
<td>71</td>
</tr>
<tr>
<td>Bulk Data Handling</td>
<td>123</td>
</tr>
<tr>
<td>C Language Interface</td>
<td>63</td>
</tr>
<tr>
<td>C++ Interface</td>
<td>90</td>
</tr>
<tr>
<td>cache</td>
<td>124, 129</td>
</tr>
<tr>
<td>c-api</td>
<td>85, 86, 87</td>
</tr>
<tr>
<td>C-API Embedded SQL</td>
<td>86</td>
</tr>
<tr>
<td>C-API MR Routines</td>
<td>85</td>
</tr>
<tr>
<td>C-API MSCALL</td>
<td>87</td>
</tr>
<tr>
<td>cgi</td>
<td>29</td>
</tr>
<tr>
<td>char</td>
<td>76</td>
</tr>
<tr>
<td>character</td>
<td>76, 143</td>
</tr>
<tr>
<td>Character Data</td>
<td>76</td>
</tr>
<tr>
<td>client/server</td>
<td>22, 27</td>
</tr>
<tr>
<td>commit</td>
<td>19, 116</td>
</tr>
<tr>
<td>Compaq</td>
<td>53</td>
</tr>
<tr>
<td>Connectivity Server</td>
<td>2, 11, 22, 27, 58, 80, 81</td>
</tr>
<tr>
<td>Consulting</td>
<td>62</td>
</tr>
<tr>
<td>Core Product</td>
<td>58</td>
</tr>
<tr>
<td>Data Import and Export</td>
<td>11, 58, 91, 98</td>
</tr>
<tr>
<td>Data Source</td>
<td>88</td>
</tr>
<tr>
<td>Database Administration</td>
<td>11, 58, 63, 91, 94</td>
</tr>
<tr>
<td>Database Backup</td>
<td>11, 58, 91, 96</td>
</tr>
<tr>
<td>Database Backup and Recovery</td>
<td>11, 58, 91, 96</td>
</tr>
<tr>
<td>Database Design &amp; Tuning</td>
<td>63</td>
</tr>
<tr>
<td>Database Integrity Checking and Maintenance</td>
<td>11, 58, 91, 95</td>
</tr>
<tr>
<td>Database Version Upgrade</td>
<td>11, 58, 91, 97</td>
</tr>
<tr>
<td>date</td>
<td>77, 122, 140</td>
</tr>
<tr>
<td>Date Data</td>
<td>77</td>
</tr>
<tr>
<td>decimal</td>
<td>78</td>
</tr>
<tr>
<td>Decimal Data</td>
<td>78</td>
</tr>
<tr>
<td>delete</td>
<td>85, 90, 113</td>
</tr>
<tr>
<td>Deterministic Response</td>
<td>45</td>
</tr>
<tr>
<td>Distributed Server</td>
<td>2</td>
</tr>
<tr>
<td>Dollar Data</td>
<td>78</td>
</tr>
<tr>
<td>double</td>
<td>79</td>
</tr>
<tr>
<td>dsql</td>
<td>18</td>
</tr>
<tr>
<td>dump</td>
<td>92, 123</td>
</tr>
<tr>
<td>Embedded Real-Time Database</td>
<td>2, 10, 37</td>
</tr>
<tr>
<td>Embedded Real-Time Toolkit</td>
<td>2, 80</td>
</tr>
<tr>
<td>Embedded SQL</td>
<td>2, 24, 29, 30, 80, 83, 84, 90</td>
</tr>
<tr>
<td>empkill</td>
<td>121, 127</td>
</tr>
<tr>
<td>empqdel</td>
<td>121, 128, 129</td>
</tr>
<tr>
<td>Empress 4GL</td>
<td>2, 121, 122, 141</td>
</tr>
<tr>
<td>Empress APIs</td>
<td>84</td>
</tr>
<tr>
<td>Empress CD</td>
<td>66, 67</td>
</tr>
<tr>
<td>Empress Database Administrative Variables</td>
<td>11, 58, 91, 93</td>
</tr>
<tr>
<td>Empress Monitors and Statistics</td>
<td>11, 58, 91, 99</td>
</tr>
<tr>
<td>Empress Package</td>
<td>66</td>
</tr>
<tr>
<td>Empress Report Writer</td>
<td>2, 59</td>
</tr>
<tr>
<td>Empress Servers</td>
<td>80</td>
</tr>
<tr>
<td>Empress Software Products</td>
<td>58</td>
</tr>
<tr>
<td>Empress SQL</td>
<td>2, 59, 63, 87, 89, 92</td>
</tr>
<tr>
<td>Empress System Variables</td>
<td>11, 58, 91, 92, 121</td>
</tr>
<tr>
<td>Empress Utilities</td>
<td>11, 58, 91</td>
</tr>
<tr>
<td>Empress-in-One</td>
<td>2</td>
</tr>
<tr>
<td>ERT Toolkit</td>
<td>2, 59</td>
</tr>
<tr>
<td>esql</td>
<td>18</td>
</tr>
<tr>
<td>export</td>
<td>98</td>
</tr>
<tr>
<td>float</td>
<td>79, 126, 143</td>
</tr>
<tr>
<td>Floating Point Data</td>
<td>79</td>
</tr>
<tr>
<td>Free BSD</td>
<td>48, 52, 53, 67, 88</td>
</tr>
<tr>
<td>Global Buffers</td>
<td>100</td>
</tr>
<tr>
<td>GSA</td>
<td>20</td>
</tr>
<tr>
<td>GUI Builder</td>
<td>2</td>
</tr>
<tr>
<td>Hitachi SH3</td>
<td>53</td>
</tr>
<tr>
<td>HP</td>
<td>53</td>
</tr>
<tr>
<td>HP-UX</td>
<td>53</td>
</tr>
<tr>
<td>l=MC</td>
<td>2, 133, 134, 135</td>
</tr>
<tr>
<td>IBM</td>
<td>53</td>
</tr>
<tr>
<td>import</td>
<td>98</td>
</tr>
<tr>
<td>Information Management</td>
<td>135, 136, 139</td>
</tr>
<tr>
<td>insert</td>
<td>85, 90, 113</td>
</tr>
<tr>
<td>Installation</td>
<td>62, 68</td>
</tr>
<tr>
<td>Installation Key</td>
<td>68</td>
</tr>
<tr>
<td>int97</td>
<td>79, 143</td>
</tr>
<tr>
<td>integer</td>
<td>79</td>
</tr>
<tr>
<td>Integer Data</td>
<td>79</td>
</tr>
<tr>
<td>Intel x86</td>
<td>53</td>
</tr>
<tr>
<td>Interactive SQL</td>
<td>2, 11, 58, 80, 87, 123</td>
</tr>
<tr>
<td>Introduction to Empress SQL</td>
<td>63</td>
</tr>
<tr>
<td>IRIX</td>
<td>53</td>
</tr>
<tr>
<td>isql</td>
<td>18</td>
</tr>
<tr>
<td>java 2, 18, 26, 29, 46, 59, 63, 72, 80, 84, 89, 142, 143</td>
<td></td>
</tr>
<tr>
<td>Java Programming</td>
<td>63, 89</td>
</tr>
<tr>
<td>Java SQL</td>
<td>2, 18, 59</td>
</tr>
</tbody>
</table>

164 Empress Software
jdbc: 2, 11, 17, 18, 26, 28, 29, 30, 58, 59, 63, 72, 80, 81, 83, 89, 143
JDBC Interface .... 11, 28, 29, 30, 58, 72, 83, 89
JDBC Local Access Interface ............ 26, 28, 30
jdbc-api ........................................... 63
jsp .............................................. 29
kernel ... 17, 23, 38, 40, 41, 42, 45, 49, 63, 71
Linux .............................................. 53
Linux PPC ............................................ 53
lock ........................................... 33, 93, 122, 125, 129, 130
longinteger ........................................ 79
MontaVista ........................................ 53
Motorola PPC .................................... 53
MR Routines 18, 23, 24, 29, 30
multilevel ........................................ 71
Multimedia Data .................................. 76
multi-processor .................................. 48
National Language Support 72, 76, 121, 140
nlscharacter ....................................... 76
nlstext ........................................... 76
null 75, 76, 112, 114, 115, 117, 122, 126, 129, 143
odbc : 2, 11, 17, 18, 25, 27, 29, 30, 39, 58, 63, 67, 80, 81, 83, 84, 88, 143
ODBC Interface 27, 29, 30, 39, 63, 80, 83, 88
ODBC Local Access 25, 27, 30
ODBC Local Access Interface 25, 27, 30
odbc-api ......................................... 63
oem ............................................. 20
oems .............................................. 14
Operational Parameters ................................ 56
Persistent Stored Modules 63, 71, 165
php .............................................. 29
pos .............................................. 19
power units .................................... 58, 68
Product Functionality 70, 71
Production Deployment 44, 48
psm .............................................. 63
psms ............................................ 17
QNX 4 & 6 ....................................... 53
query ......................................... 123, 125
queue ........................................... 59
radar .......................................... 36, 41
raid ............................................ 46, 48, 138
ram ............................................. 46
RDBMS Engine 58, 67, 70
real ............................................. 79
Real-Time Data Collector 2, 59
recall ........................................... 75
recovery ...................................... 34, 48, 82, 93, 96, 127
Referential constraints ................................ 19
replication 2, 11, 18, 31, 32, 34, 48, 63, 80, 82, 130
Replication Scenarios ................................ 31
Replication Server 2, 11, 48, 63, 80, 82
Report Writer 59, 129, 143
RTLinux ............................................. 53
SCO ............................................... 53
scsi ........................................... 46
select .......................................... 59, 92, 123, 144
gsi ............................................. 53
SGI ................................................. 53
Shared Memory 11, 58, 91, 100, 127
shortinteger ...................................... 79
Small Footprint 39, 40, 46, 71
Solaris ............................................ 53
Solaris for Intel .................................... 53
Source Level Customization 62
gsl 11, 18, 24, 39, 58, 59, 63, 71, 86, 87, 89, 90, 101, 123, 143, 144
Stand-alone Mode ................................ 83
Stand-alone Scenarios ................................ 23
Static and Dynamic SQL 63
SUN ................................................. 53
SUN O/S .......................................... 53
Support ...................................... 49, 61
Supported Data Types 70, 73
Supported Platforms 52, 53
SUSE ............................................. 53
System Variables 92, 120
Technical Services 61, 62
text ............................................ 76
time ............................................ 77, 78, 143
Time Data ......................................... 77
timestamp ....................................... 18, 59
TimeSys ........................................... 53
Training ......................................... 62
triggers ....................................... 17, 63, 71, 141
Triggers and Stored Procedures 71
Tru64 UNIX ...................................... 53
truncate ........................................ 126
Unicode ........................................ 17, 72
update ......................................... 85, 90, 113
Updates ........................................... 61
Upgrades ....................................... 61
UTF-8 ............................................ 17, 72
VxWorks ........................................... 53
WIN 2000 ......................................... 53
WIN NT ........................................... 53
WIN XP ........................................... 53