Empress
Ultra Embedded Database
V10.20

Product Profile
2009

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.

Empress RDBMS, Empress Embedded, Empress Ultra Embedded, Empress Memory Embedded, Empress RTDC, Empress Real-Time Data Collector, Empress ERT Toolkit, Empress Embedded Real-Time Toolkit, Empress DB Pipe, I=MC^2, "The Embedded Real-Time Database", Empress Connectivity Server, Empress Replication Server, Empress Distributed Server, Empress Database Server, Empress Stand-alone, Empress SQL Utilities, Empress GUI Builder, Empress-in-One, Empress Interactive SQL, Empress Java SQL, Empress 4GL, Empress Report Writer, Empress MR Routines, Empress Kernel "C" Interface, Empress Embedded SQL, Empress MSCALL, Empress Hypermedia, Empress ODBC, Empress JDBC are trademarks of Empress Software Incorporated.

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

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 sub-paragraph © (1) (ii) of the Rights in Technical Data and Computer Software clause at 52.227-7013.

02/2009

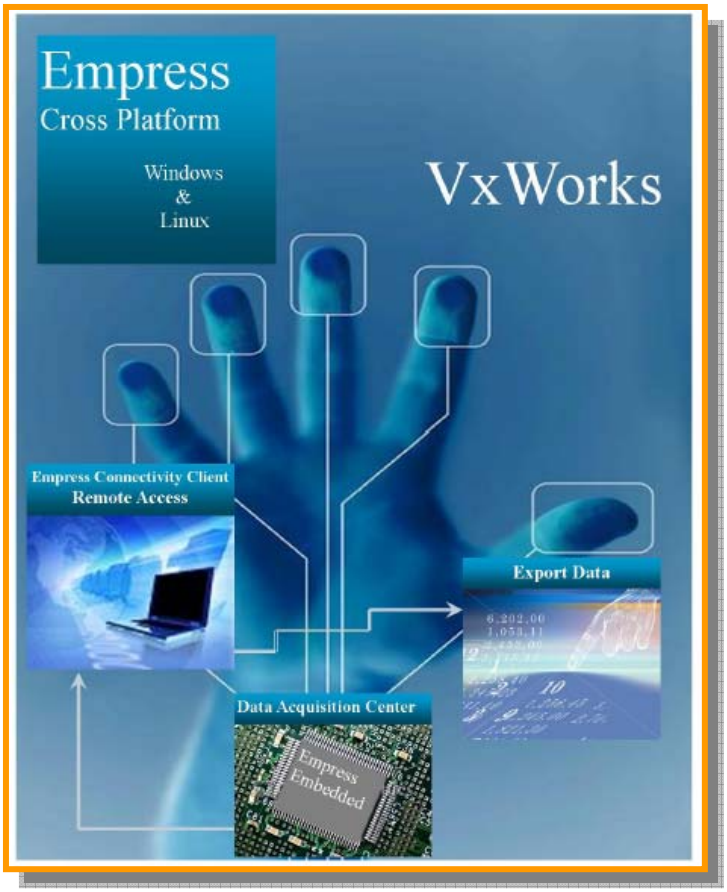
Part #: 011-2009-217

The Empress Product Profile at a glance:

Table of Contents

1 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.



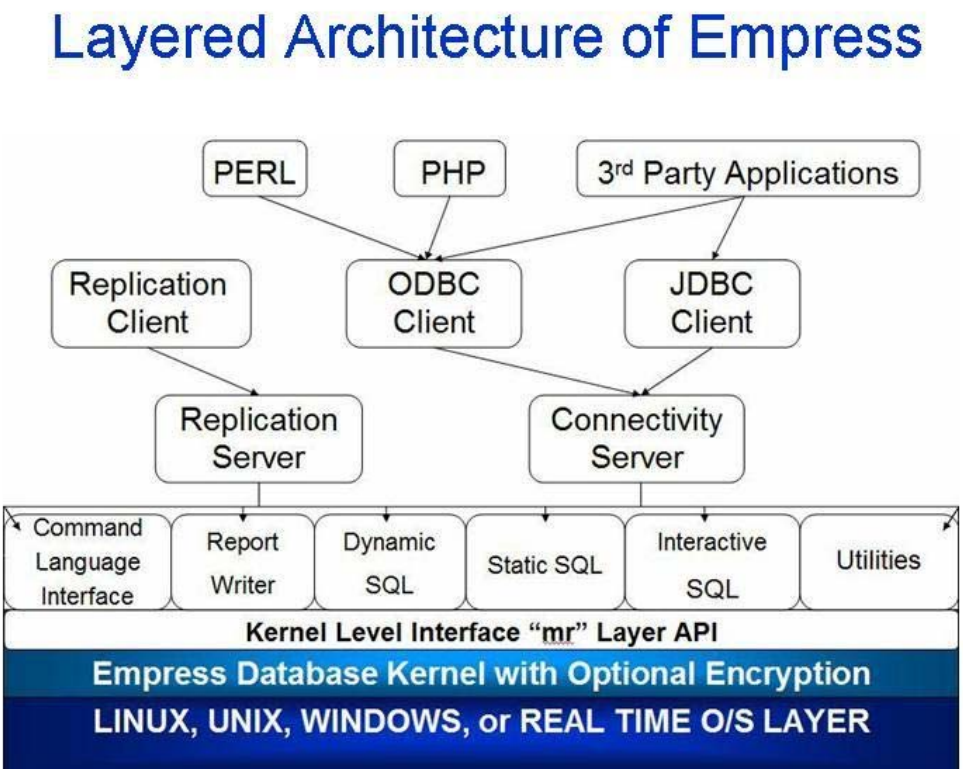
Empress Ultra Embedded Database consists of ***Empress*** Embedded Database, APIs, Connectivity Server, Replication Server and Utilities.

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
PERL
PHP
3rd Party Applications
Replication Client
ODBC Client
JDBC Client
Replication Server
Connectivity Server
Command Language Interface
Report Writer
Dynamic SQL
Static SQL
Interactive SQL
Utilities
Kernel Level Interface "mr" Layer API
Empress Database Kernel with Optional Encryption
LINUX, UNIX, WINDOWS, or REAL TIME O/S LAYER

2 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

Adoptable, 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

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

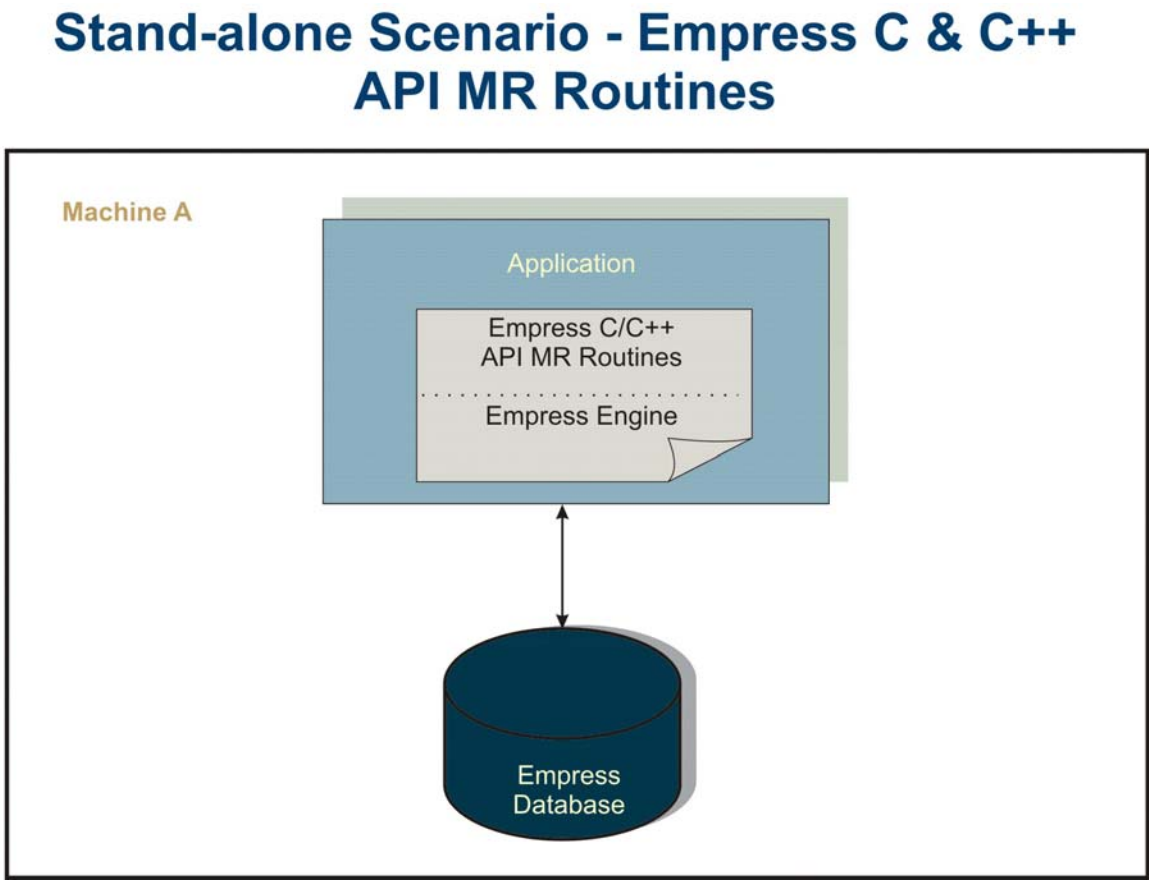


Fig. 4.1 Stand-alone Scenario – Empress C/C++ API MR Routines

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.

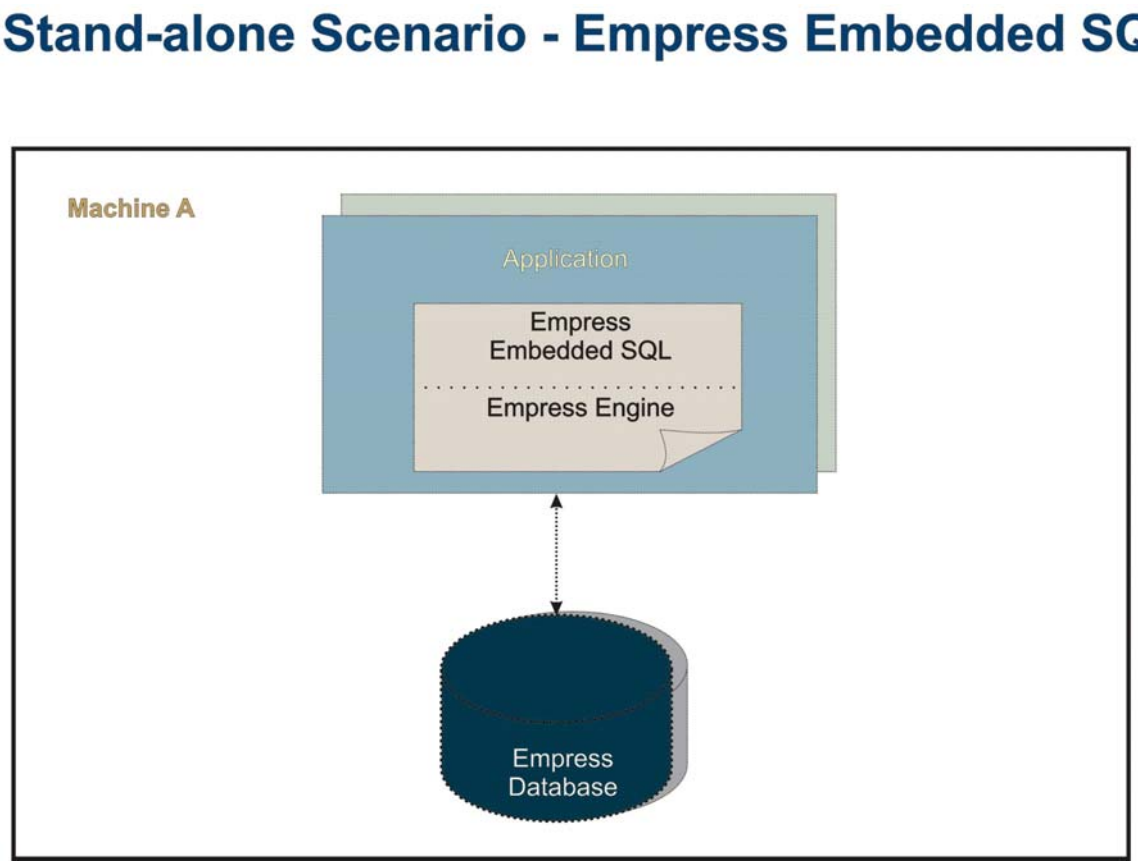


Fig. 4.2 Stand-alone Scenario – Empress C/C++ API Embedded SQL

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.

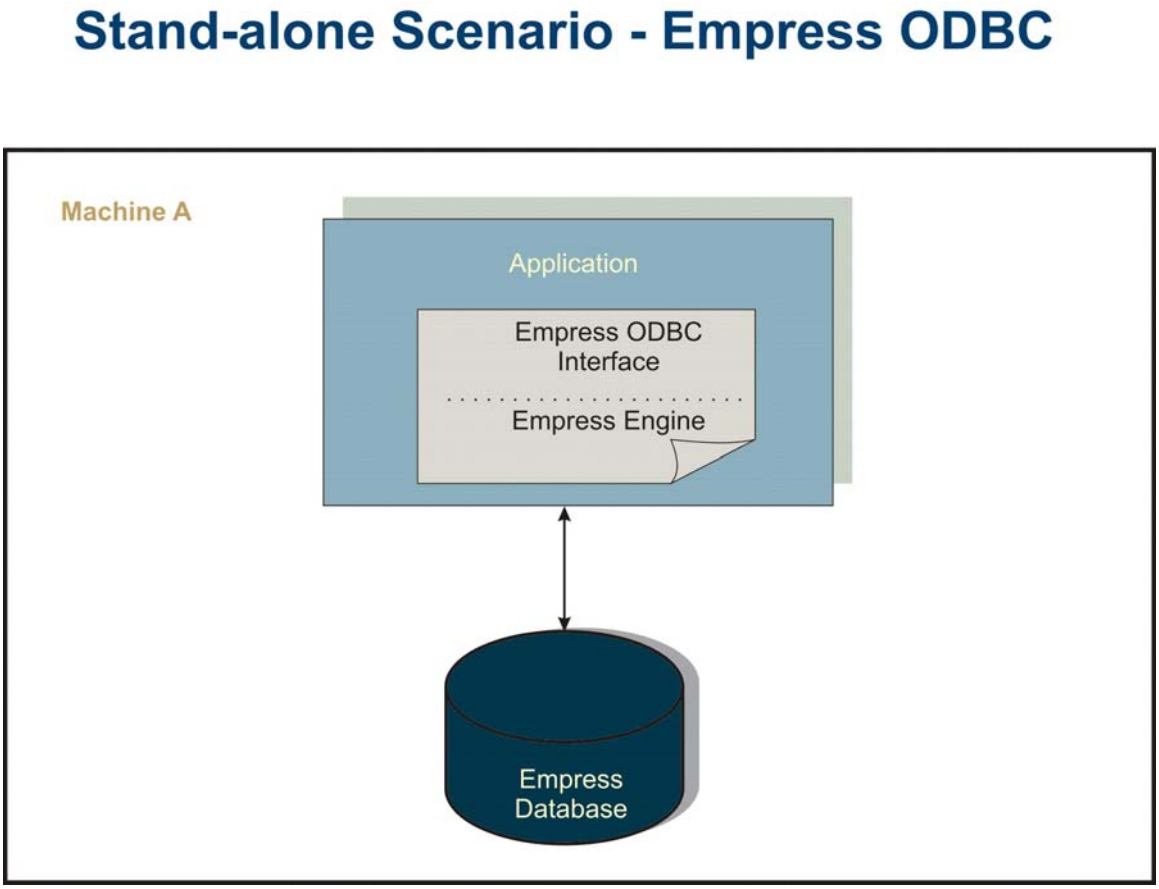


Fig. 4.3 Stand-alone Scenario – Empress ODBC Local Access Interface

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.

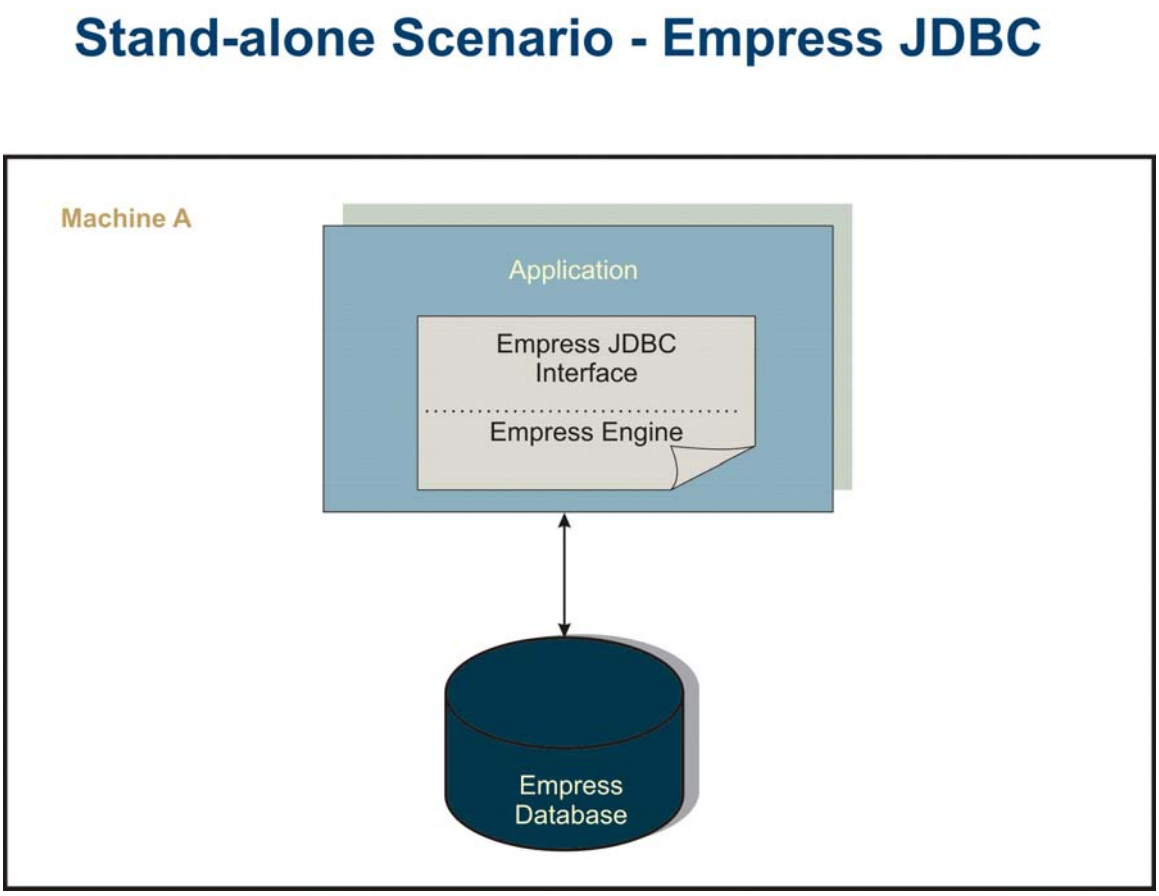


Fig. 4.4 Stand-alone Scenario – Empress JDBC Local Access Interface

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

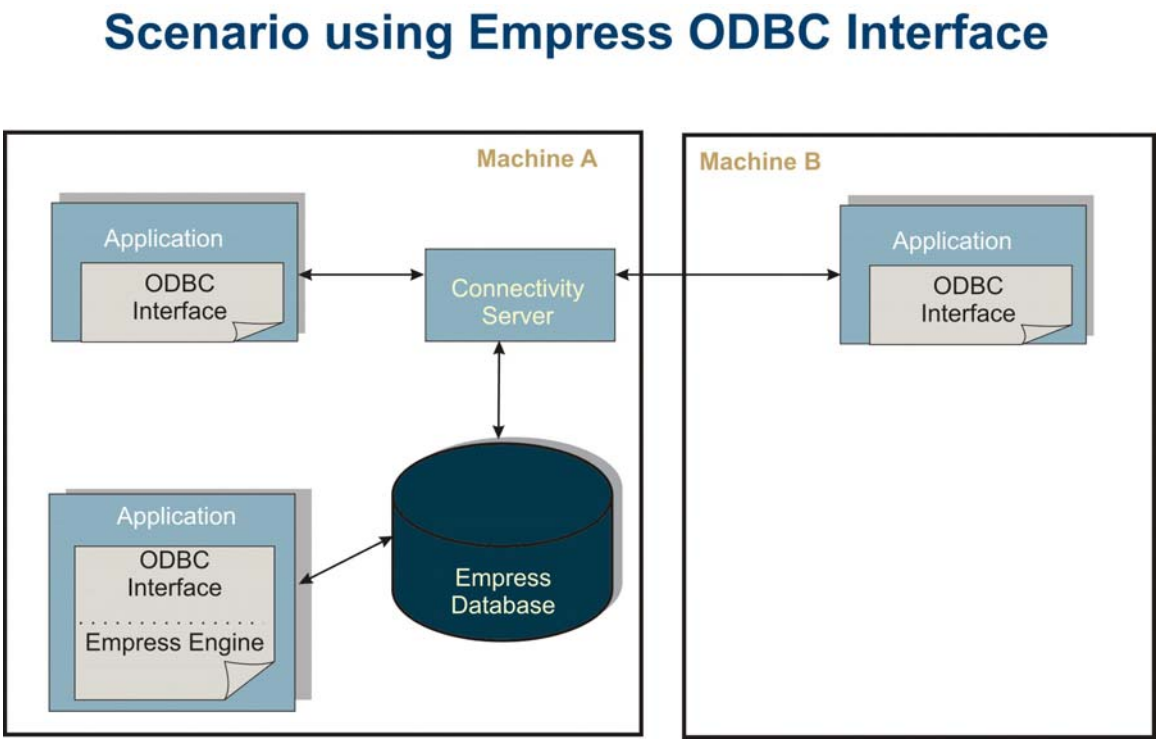


Fig. 4.5 Scenario Using Empress ODBC Interface

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.

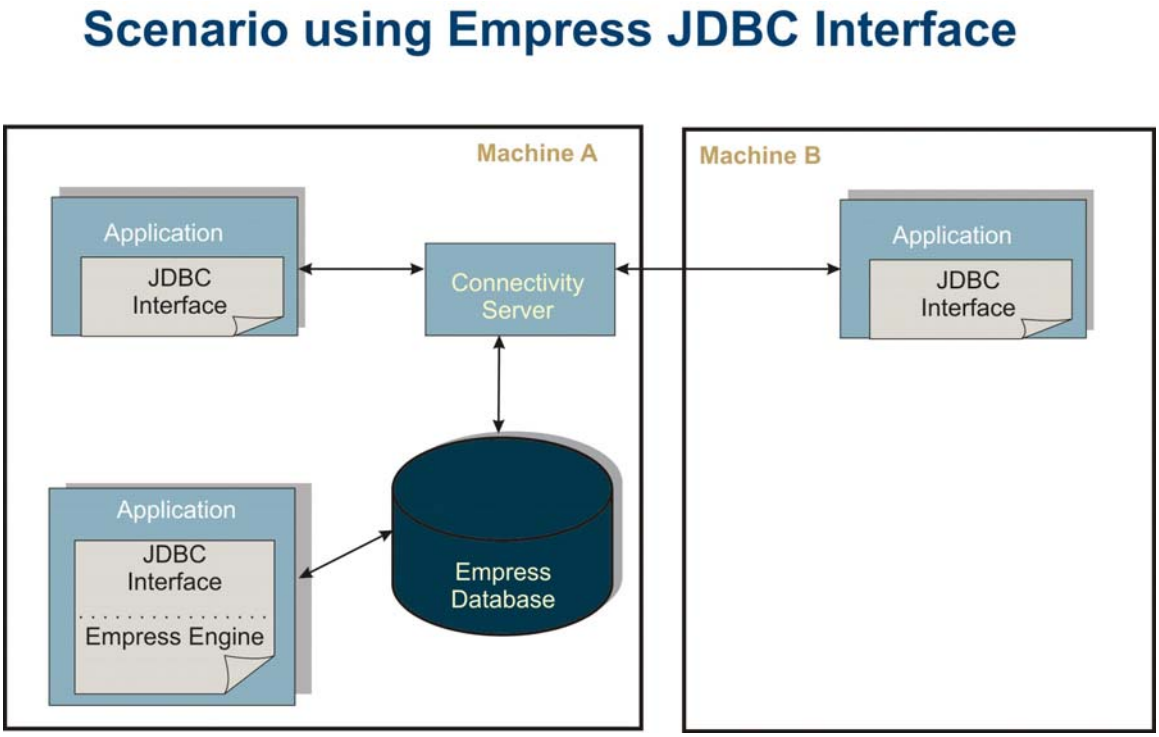


Fig. 4.6 Scenario Using Empress JDBC Interface

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.

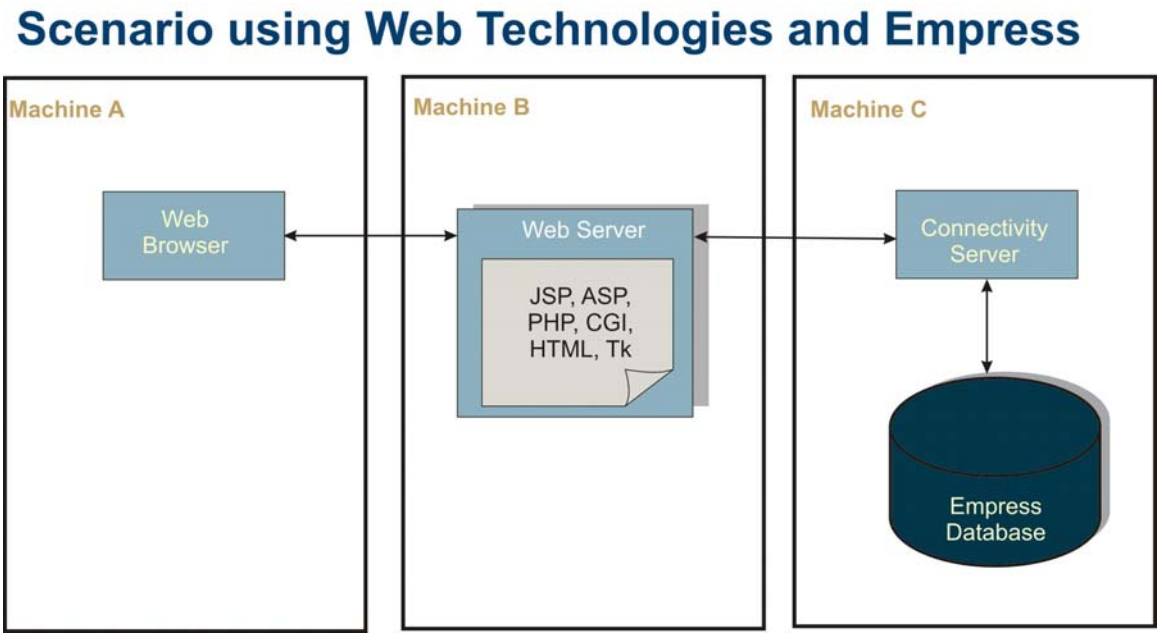


Fig. 4.7 Scenario Using Web Technologies and Empress Interfaces

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 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

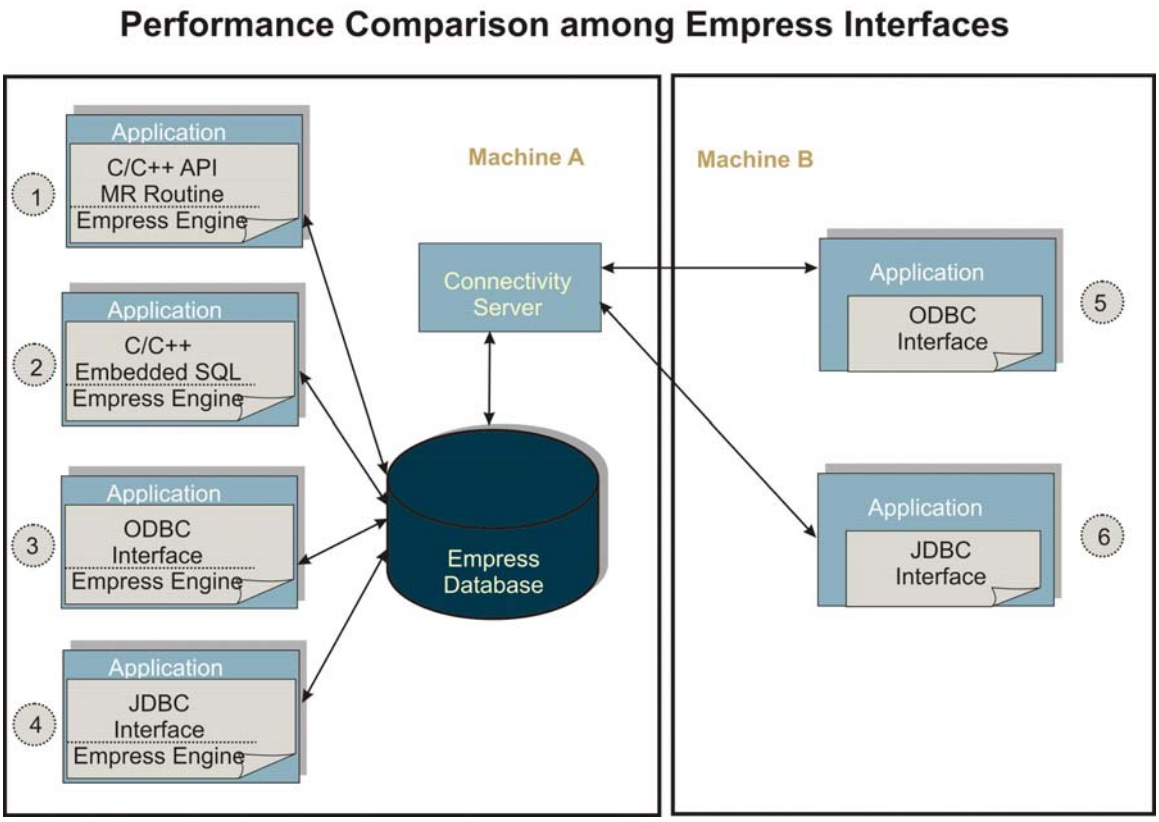


Fig. 4.8 Performance Comparison among Empress Interfaces

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

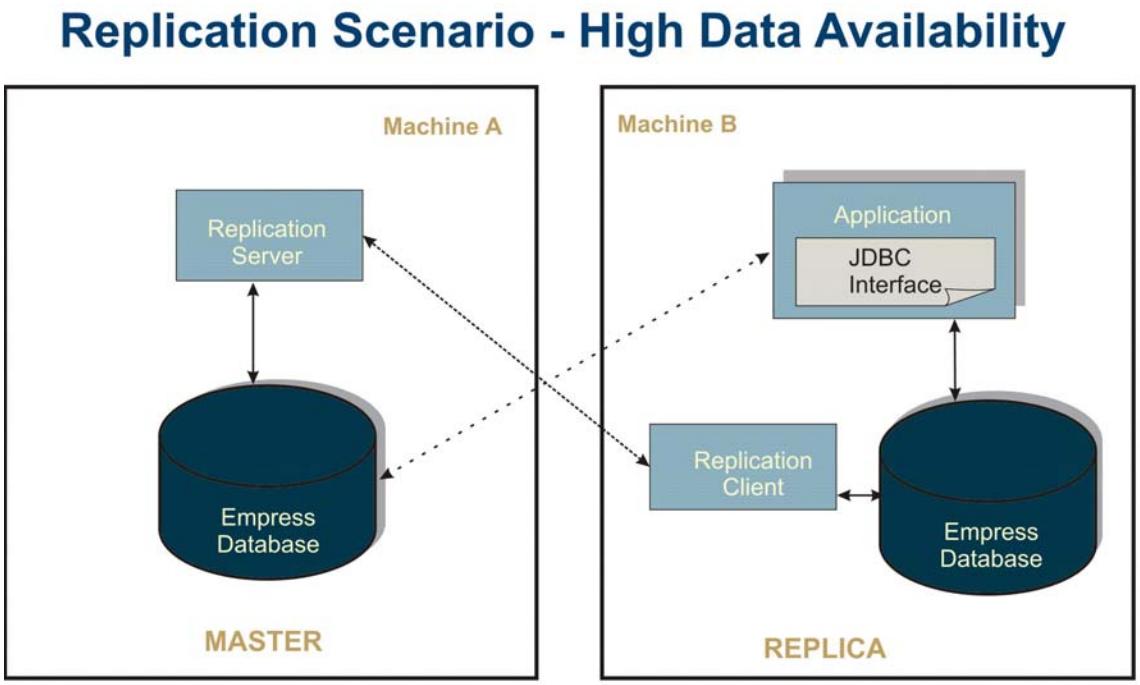


Fig. 4.9 Replication Scenario – High Data Availability

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.

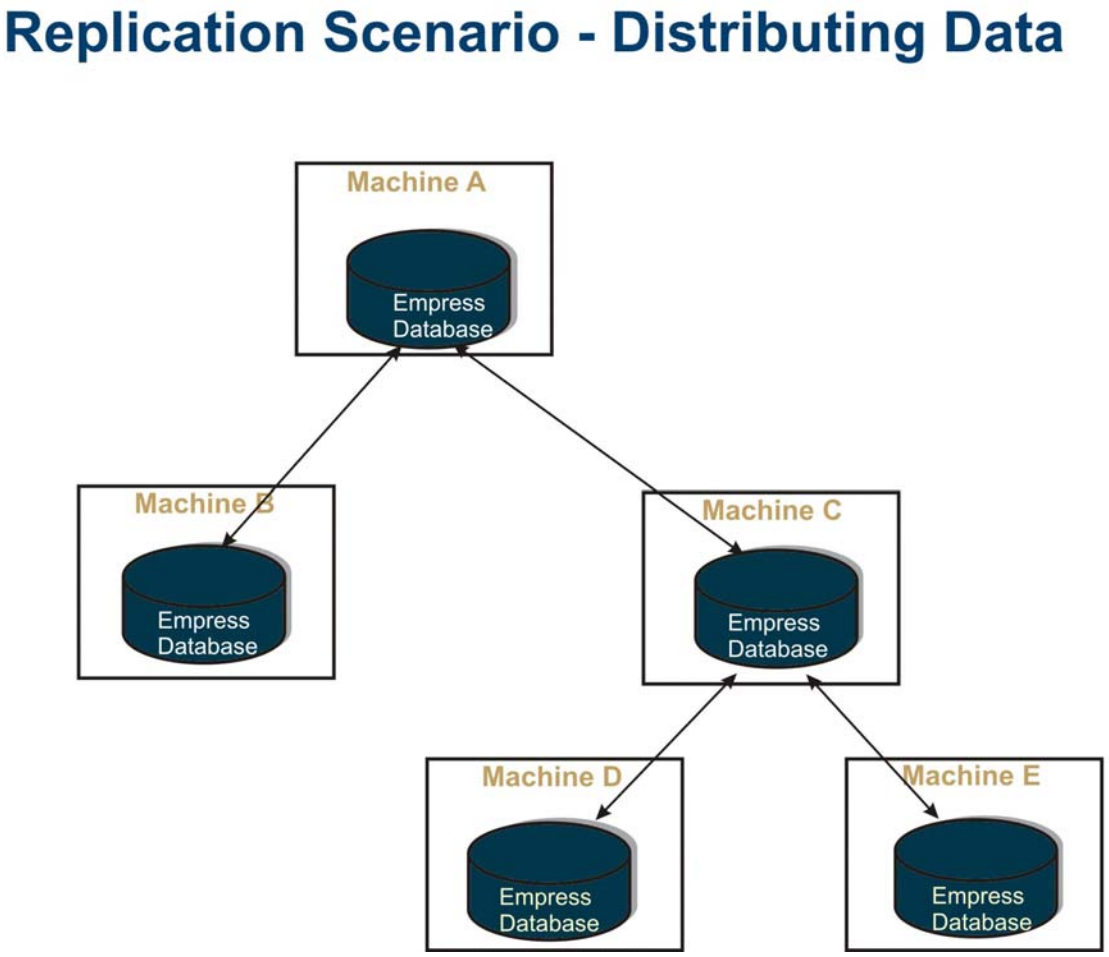


Fig. 4.10 Replication Scenario – Distributing Data

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.

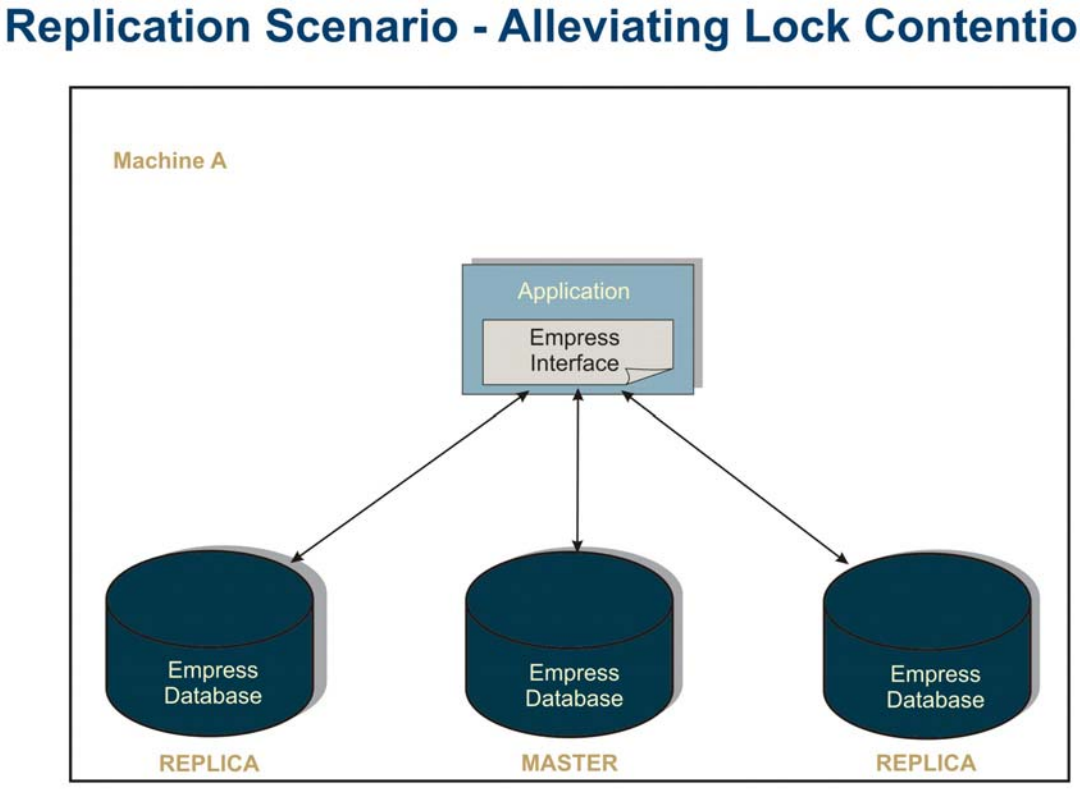


Fig. 4.11 Replication Scenario – Alleviating Lock Contention

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.

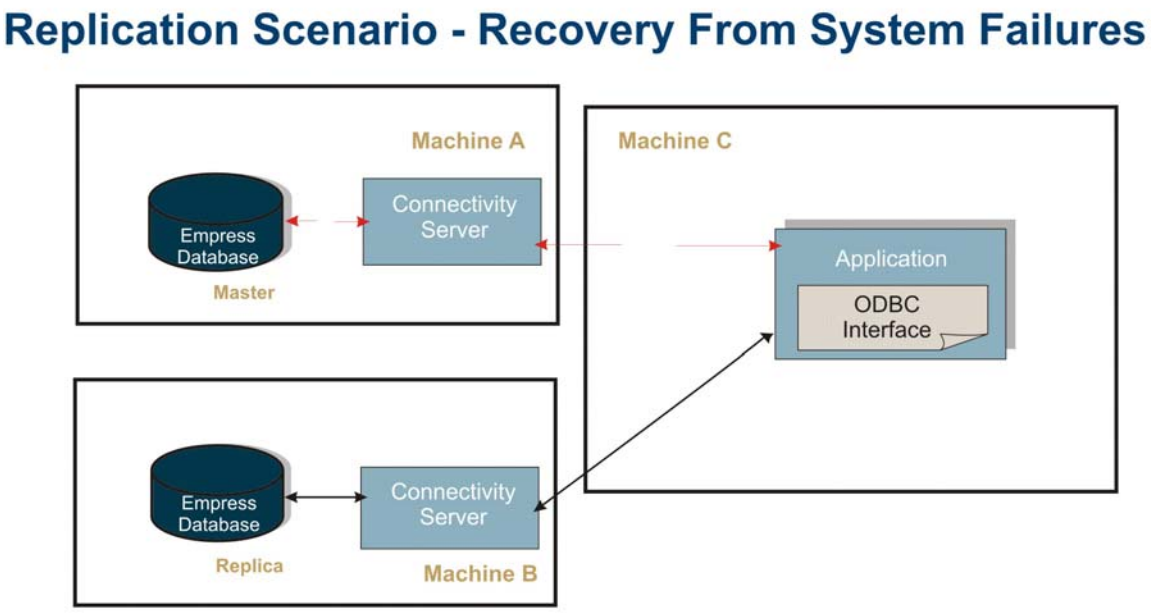


Fig. 4.12 Replication Scenario – Recovery From System Failures

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.

5 Where is Empress Being Used?

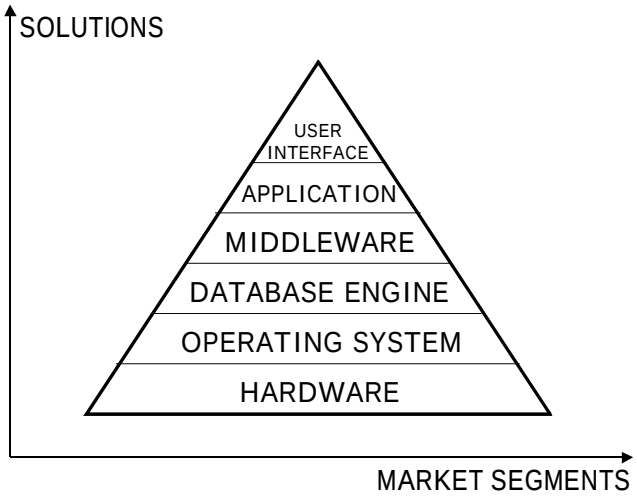
Empress is targeting its embedded database at the following embedded and real-time markets:

- **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.

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

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

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 S:

- 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

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.

Flexible Development

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.

Friendly Storage Requirements

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.

Small Footprint

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.

Wide Range of Platforms

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.

Cost Effective Partnership Programs

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.

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.

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.

In-Depth Database Expertise	Empress Software front line Technical Support staff is knowledgeable database experts who give high quality support.
Easy Access to User Data	Users can access data stored in ***Empress*** database through standard interfaces, multiple API's, kernel level interface application programs as well as many popular third party software products.
High Performance	Users of ***Empress*** Ultra Embedded Database will benefit from one of the fastest embedded real-time database engines in the industry.

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:

Operating System	Environment	Servers	Clients
AIX (IBM)	PPC	Yes	Yes
Apple Mac O/SX	PPC, x86, x86-64	Yes	Yes
Bluecat Linux (real-time)	x86	Yes	Yes
BSD	x86, x86-64	Yes	Yes
Cray Unicos	Cray, x86-64	Yes	Yes
Fedora Core	x86, x86-64	Yes	Yes
HP	HP-UX	Yes	Yes
IBM	AIX	Yes	Yes
IRIX	SGI, x86-64 (AMD)	Yes	Yes
Linux (most)	x86, x86-64, most 2.6 kernels	Yes	Yes
Lynx O/S (real-time)	PPC, x86	Yes	Yes
MontaVista Linux (real-time)	ARM, MIPS, PPC, SH-4, x86	Yes	Yes
Motorola PPC - BSD, Linux	Apple Mac OS-X, Linux PPC	Yes	Yes
Motorola PPC Real-Time	Lynx O/S, QNX, VxWorks	Yes	Yes
QNX 4 & 6	PPC, x86	Yes	Yes
Red Hat 7, 8, 9	x86	Yes	Yes
Red Hat EL 3, 4, 5	x86, x86-64	Yes	Yes
SUN Solaris 8, 9 ,10	Sparc, x86-64	Yes	Yes
SUSE-LE 9, 10	x86, x86-64	Yes	Yes
TimeSys Linux	PPC, x86	Yes	Yes
Tru64 UNIX	DEC Alpha	Yes	Yes
Ubuntu 6, 7, 8	x86	Yes	Yes
VxWorks 5, 6	ARM, MIPS, PPC, SH-4, x86	Yes	Yes
Wind River Linux	ARM, x86	Yes	Yes
Windows CE	ARM, x86	Yes	Yes
Windows Mobile 5,6	ARM	Yes	Yes
Windows Server 2003, 2008	x86, x86-64	Yes	Yes
Windows 2000, XP, Vista	x86, x86-64	Yes	Yes

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.

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

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.

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).

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

9.3 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.

10 What is in the Empress Ultra Embedded Database 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, Free BSD, 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

10.3 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.

11 Empress' Features by Module

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:

Data Type Category	Empress Data Type Names	Comment
Character	CHARACTER	ANSI
	CHAR	ANSI
	SQL_CHAR	ODBC
	NATIONAL CHARACTER	ANSI
	NATIONAL CHAR	ANSI
	NCHAR	ANSI
	ECHARACTER	Empress Extension
	ECHAR	Empress Extension
	NLSCHARACTER	Empress Extension
	NLSCHAR	Empress Extension
	CHARACTER VARYING	ANSI
	CHAR VARYING	ANSI
	VARCHAR	ANSI
	SQL_VARCHAR	ODBC
	NATIONAL CHARACTER VARYING	ANSI
	NATIONAL CHAR VARYING	ANSI
	NCHAR VARYING	ANSI
	VARNCHAR	ANSI
	TEXT	Empress Extension
	NLSTEXT	Empress Extension
	CHARACTER LARGE OBJECT	ANSI
	CHAR LARGE OBJECT	ANSI
	CLOB	ANSI
	NATIONAL CHARACTER LARGE OBJECT	ANSI
	NATIONAL CHAR LARGE OBJECT	ANSI
	NCHAR LARGE OBJECT	ANSI
	NCLOB	ANSI

Integer	TINYINT	ANSI
	SQL_TINYINT	ODBC
	INTEGER8	Empress Extension
	SMALLINT	ANSI
	SQL_SMALLINT	ODBC
	INTEGER16	Empress Extension
	INTEGER	ANSI
	INT	ANSI
	SQL_INTEGER	ODBC
	INTEGER32	Empress Extension
	BIGINT	ANSI
	SQL_BIGINT	ODBC
	INTEGER64	Empress Extension
Sequence	SEQUENCE32	Empress Extension
	SEQUENCE64	Empress Extension
Boolean	BOOLEAN	ANSI
Float	REAL	Empress Extension
	SQL_REAL	ANSI
	FLOAT32	ODBC
	DOUBLE PRECISION	Empress Extension
	SQL_DOUBLE	ANSI
	FLOAT64	ODBC
	EFLOAT	Empress Extension
	FLOAT	ANSI
	SQL_FLOAT	ODBC
Decimal	DECIMAL	ANSI
	DEC	ANSI
	NUMERIC	ANSI
	SQL_DECIMAL	ODBC
	SQL_NUMERIC	ODBC
	DOLLAR	Empress Extension

Date	EDATE	Empress Extension
	EPOCH_TIME	Empress Extension
	ETIME	Empress Extension
	MICROTIMESTAMP	Empress Extension
	DATE SQL_TYPE_DATE	ANSI ODBC
	TIME TIME WITHOUT TIME ZONE SQL_TYPE_TIME	ANSI ANSI ODBC
	TIMESTAMP TIMESTAMP WITHOUT TIME ZONE SQL_TYPE_TIMESTAMP	ANSI ANSI ODBC
Binary (Byte Stream)	BULK	Empress Extension
	BINARY LARGE OBJECT BLOB	ANSI ANSI

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.

Date Data

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

Table 2-3: ***Empress*** DATE Data Type Formats

Type	Format	Year Range	Example
0	yyyymmdd	0000-9999	20020627
1	dd aaaaaaaaa yyyy	0000-9999	27 June 2002
2	aaaaaaaaa dd, yyyy	0000-9999	June 27, 2002
3	mm/dd/yy	nn00-nn99 *	06/27/02
4	dd/mm/yy	nn00-nn99 *	27/06/02
5	dd aaa yy	nn00-nn99 *	27 Jun 02
6	aaa dd, yy	nn00-nn99 *	Jun 27, 02
7	mm/dd/yyyy	0000-9999	06/27/2002
8	dd/mm/yyyy	0000-9999	27/06/2002

* 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:

Type	Output Style
0	20030828145615
1	28 August 2003 14:56:15
2	August 28, 2003 14:56:15
3	08/28/03 14:56:15
4	28/08/03 14:56:15
5	28 Aug 03 14:56:15
6	Aug 28, 03 14:56:15
7	08/28/2003 14:56:15
8	28/08/2003 14:56:15

MicroSecond Timestamp Data

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

Type	Output Style
0	20070828145615234599
1	28 August 2007 14:56:15.234599
2	August 28, 2007 14:56:15.234599
3	08/28/07 14:56:15.234599
4	28/08/07 14:56:15.234599
5	28 Aug 07 14:56:15.234599
6	Aug 28, 07 14:56:15.234599
7	08/28/2007 14:56:15.234599
8	28/08/2007 14:56:15.234599

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:

Dollar Type	Format
1	$1,234.00
2	$****1,234.00
3	1234.00

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

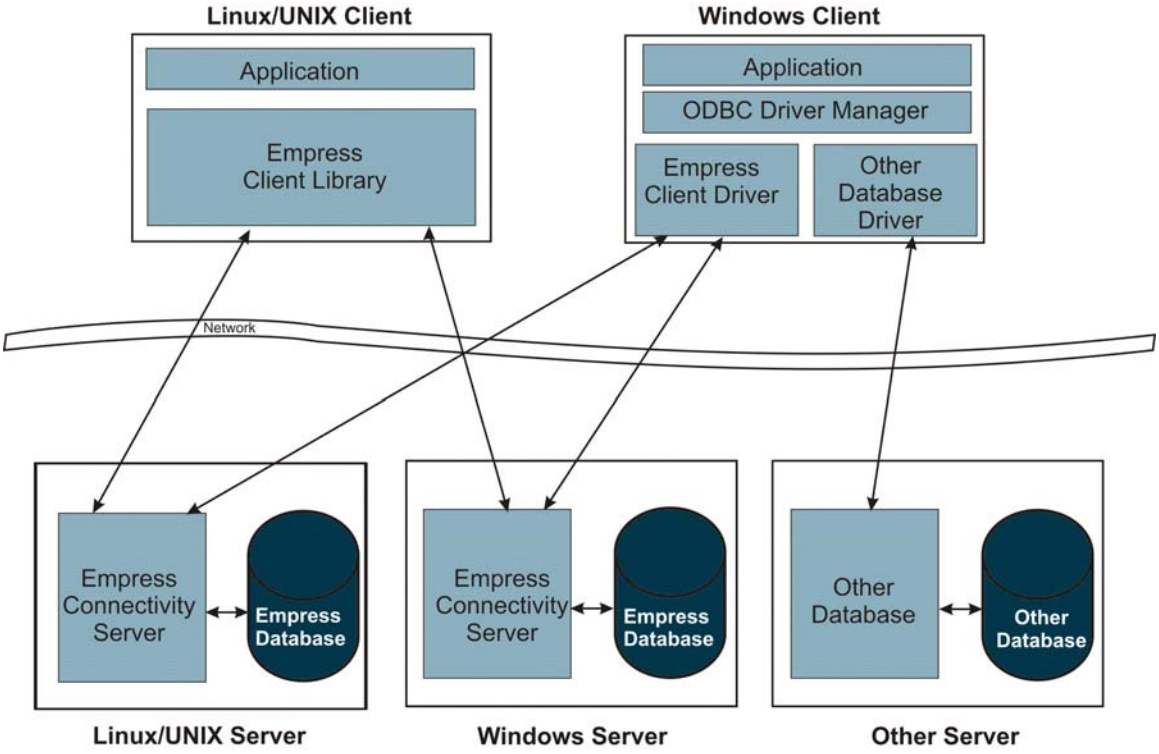
Empress Ultra Embedded Database can run in stand-alone mode using any of the "C" APIs including the ***Empress*** kernel level "MR" Routines, the Embedded SQL for "C", the ODBC Interface with "C" calls as well as the "C++" Interface, the JDBC for "Java" Interface, the Embedded Real-Time Toolkit and Interactive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

```
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.

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.

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.

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.

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.

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

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

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

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: jlji@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: esj-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

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:

Open Routines	
Name	**Purpose**
`mropen`	Open a table for use with the MR Routines, with locking as prescribed in the data dictionary; terminate calling program on failure.
`mrtopen`	Open a table; do not terminate on failure.
`mropdict`	Open all tables of the data dictionary; terminate calling program on failure.
`mrtopdict`	Open all tables of the data dictionary; do not terminate calling program on failure.
`mrclose`	Close a table after use with the MR Routines and unlock it if locked.
`mrcldict`	Close the data dictionary.
`mrqdb`	Check if a directory is a database.

The following table lists the Routines used to control locking:

Lock Routines	
Name	**Purpose**
`mrlktab`	Lock table at table level.
`mrlkrec`	Hold a lock on a record.
`mrultab`	Remove a table level lock.
`mrulrec`	Remove the lock created by `mrlkrec`.

The following table lists the Routines used to allocate and de-allocate memory:

Allocate Space Routines	
Name	**Purpose**
`mrmkrec`	Allocate space to store one record.
`mrfrrec`	Free space allocated by `mrmkrec`.
`mrspv`	Allocate space to store an attribute value.
`mrfree`	Free space allocated by `mrspv`.

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

Attribute Descriptor Routines	
Name	**Purpose**
`mrngeta`	Get a descriptor for a named attribute.
`mrigeta`	Get a descriptor for a numbered attribute.
`mrganame`	Find an attribute name given its descriptor.

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

Filling Records Routines	
Name	**Purpose**
`mrputvs`	Assign a value (external format) to an attribute in a record; do not terminate on failure.
`mrputvi`	Assign an integer to an attribute in a record; do not terminate on failure.
`mrputi`	Assign a value (internal format) to an attribute in a record; do not terminate on failure.
`mrmptvs`	Assign a value (external format) to an attribute in a record; terminate on failure.
`mrmptvi`	Assign an integer to an attribute in a record; terminate on failure.
`mrsetnv`	Assign a null value to an attribute in a record.
`mrsetnr`	Assign a null values to all attributes in a record.

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

Insert Routines	
Name	**Purpose**
`mradd`	Insert a record; terminate on failure.
`mrtadd`	Insert a record; do not terminate on failure.
`mraddend`	Clean up after inserting records into a table.

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

Delete Routines	
Name	**Purpose**
`mrdel`	Delete a record; terminate on failure.
`mrtdel`	Delete a record; do not terminate on failure.
`mrdelend`	Clean up after deleting records from a table.

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

Update Routines	
Name	**Purpose**
`mrcopyr`	Copy attribute values from one record to another.
`mrput`	Update a record; terminate on failure.
`mrtput`	Update a record; do not terminate on failure.

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

Initialize Retrieval Routines	
Name	**Purpose**
`mrgetbegin`	Associate qualifications with records for retrievals; terminate on failure.
`mrtgtbegin`	Associate qualifications with records for retrievals; do not terminate on failure.
`mrsrtbegin`	Associate qualifications with records for sorted retrievals; terminate on failure.
`mrtsrbegin`	Associate qualifications with records for sorted retrievals; do not terminate on failure.

The following table lists the Routines used to retrieve values from a table:

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

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

Compare Routines	
Name	**Purpose**
`mrcompare`	Find whether a variable is equal to, less than, or greater than an attribute value.
`mrnullr`	Find whether an entire record is null.
`mrnullv`	Find whether an attribute value is null.

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

Input Conversion Routines	
Name	**Purpose**
`mrcvt`	Convert a value from external format to file format. (Shared buffer with other MR Routines.)
`mrcvtin`	Convert a value from internal format to file format. (Shared buffer with other MR Routines.)
`mrcvtv`	Convert a value from external format to file format. (Shared buffer with other `mrcvti`.)
`mrcvtv2`	Convert a value from external format to file format. (Shared buffer with other `mrcvti2`.)
`mrcvti`	Convert a value from internal format to file format. (Shared buffer with other `mrcvtv`.)
`mrcvti2`	Convert a value from internal format to file format. (Shared buffer with other `mrcvtv2`.)

The following table lists the Routines used to compare attributes:

Qualification Routines	
Name	**Purpose**
`mrqcon`	Compare an attribute value with a constant; terminate on failure.
`mrtqcon`	Compare an attribute value with a constant; do not terminate on failure.
`mrqrng`	Compare an attribute value with a given range; terminate on failure.
`mrtqrng`	Compare an attribute value with a given range; do not terminate on failure.
`mrqatr`	Compare two attribute values; terminate on failure.
`mrtqatr`	Compare two attribute values; do not terminate on failure.
`mrqmch`	Match an attribute value with a constant; terminate on failure.
`mrtqmch`	Match an attribute value with a constant; do not terminate on failure.
`mrqnul`	Compare an attribute value with null; terminate on failure.
`mrtqnul`	Compare an attribute value with null; do not terminate on failure.

mrqseq	See if an attribute is equal to a string; terminate on failure.
mrqieq	See if an attribute is equal to an integer; terminate on failure.
mrqand	Perform an AND on any two of the above comparisons.
mrqor	Perform an OR on any two of the above comparisons.
mrqnot	Perform a NOT on any two of the above comparisons.
mrqlst	Qualify records in the given list of previously found pointers to records.

The following table lists the Routines used to control transactions:

Transaction Routines

Name	Purpose
mrtrstart	Start a transaction.
mrtrcancel	Cancel a transaction.
mrtrcommit	Commit a transaction.
mrtrsave	Set save point in transaction.
mrtrrollback	Roll back to save point.

The following table lists the Routines used to build expressions:

Expression Routines

Name	Purpose
mrebegin	Start building expression.
mreend	Finish building expression.
mreabort	Abort building expression.
mrqexpr	Put expression in qualification.
mrerun	Run expression and return result.
mrefree	Free expression descriptor.
mrerecattr	Add an attribute operand.
mrecons	Add a constant operand.
mreicvar	Associate a control variable with the most recent variable.
mreivar	Add a variable operand.

`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:

Error Handling Routines	
Name	**Purpose**
`mrprterr`	Print a suitable message if one of the `mrt` Routines fails; terminate calling program.
`mrerrmsg`	Retrieve last error message.

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:

Path Related Variables	
Variable	**Function**
MSHELPPATH	Directory for ***Empress*** help files.
MSNLSDB	The National Language Support database.
MSTERMDB	The terminal definition database for ***Empress*** 4GL.
MSAPFMDB	***Empress*** 4GL form database.
MS4GLPSHEADER	***Empress*** 4GL Postscript header file.
MSKILLLOGDIR	Directory for ***Empress*** utility `empkill` log file.
MSCRAYQDELLOGDIR	Directory for ***Empress*** utility `empqdel` log file. This is for Cray only.
MSGUIDB	System directory for ***Empress*** GUI.
MSGUIDEFCOLORFILE	Directory for ***Empress*** GUI color definition.
MSSHAREDMEMORYDIR	Memory mapped file directory for the platform which uses `mmap` (memory mapping).

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

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

	temporary tables used during retrieval tasks.

Transactions Related Variables

Variable	Function
MSTRANSCOMMENT	Comment string for transactions.
MSTRANSSYNC	Flag to cause immediate write to file.
MSTRANSTABLELOCK	Flag to force table-level locking in transactions.
MSTRANSUFNBI	The size of before image list for the transaction.
MSTRANSUFNGFL	Controls free record access.
MSTRANSWARMPROTECT	Flag to protect transactions from warm restart.

Locking Related Variables	
Variable	**Function**
MSCFEXCLRETRY	Number of retries for exclusive access to the coordinator file.
MSCFEXCLSLEEP	Interval between retries for exclusive lock access to the coordinator file.
MSEXCLRETRY	Number of retries for exclusive access to lock file.
MSEXCLSLEEP	Interval between retries for exclusive lock access.
MSIAEXCLRETRY	Number of exclusive access retries for Interactive Interface.
MSIAEXCLSLEEP	Exclusive access retry interval for Interactive Interface.
MSIALOCKRETRY	Number of lock retries for Interactive Interface.
MSIALOCKSLEEP	Lock retry interval for Interactive Interface.
MSINDEXRETRY	Number of retries to lock an index.
MSINDEXSLEEP	Interval between retries to lock an index.
MSLOCKRETRY	Number of retries to actually make a lock entry.
MSLOCKSLEEP	Interval between retries to actually make a lock entry.

Query Related Variables	
Variable	**Function**
MSQLAUTOPAGE	Flag to send all command output to a paging program.
MSQLCMDSAVE	Number of commands saved by history mechanism.
MSQLCOUNT	Flag to display number records selected, updated, deleted.
MSQLECHO	Controls commands echoing.
MSQLONELINE	Indicates that a new line terminates a command.
MSQLPROMPT1	Primary ***Empress*** prompt.
MSQLPROMPT2	Secondary ***Empress*** prompt.

MSQLTRANSACTION	Each command is a transaction and may be cancelled.
MSQLVARCHARS	Characters used in ***Empress*** variable notation.

Select Output Formatting Related Variables

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

On-Line Backup Related Variables	
Variable	**Function**
MSCOORDTIMERFREQ	The time period (in seconds) within which a client must check in with the coordinator.
MSOLBBACKUPDEVICE	On-line backup device name.
MSOLBBLOCKSIZE	On-line backup device block size.
MSOLBRECOVERYLOG	On-line backup recovery log file.

Shared Memory Related Variables	
Variable	**Function**
MSMKDBSHMEM	Creates database with shared memory.
MSSHMLOCATION	System address where the shared memory segment will exist.
MSSHMPERMS	Access permissions for shared memory partitions.

Dirty Read and Data Streaming Related Variables	
Variable	**Function**
MSBULKSEGMENTSIZE	Memory size for transferring bulk data in small pieces.
MSVALIDATELEVEL	Behaviour of dirty read as a result of checksum.
MSVALIDATERETRY	Number of retries ***Empress*** will make to read an invalid record.
MSVALIDATESLEEP	Interval in seconds between retries to read invalid records.

Database Utility Related Variables	
Variable	**Function**
MSKILLRETRY	Number of times `empkill` attempts to send the termination signal specified by MSKILLSIGNAL.
MSKILLSLEEP	Interval between retries `empkill` attempts to send the termination signal specified by MSKILLSIGNAL.
MSKILLSIGNAL	Signal for `empkill` to terminate.

Server/Internet Related Variables	
Variable	**Function**
MSSERVERHEARTBEAT	Time period for checking Database Server idle time.
MSSERVERNETTYPE	Server access type.
MSSERVERTERSELOG	Compact format for server log file information.
MSINETREPLYTIMEOUT	Timeout value for the client awaiting a high-level response.
MSINETREPLYRETRY	Number of retries for the client awaiting a high-level response.
MSINETMESSAGETIMEOUT	Timeout value for the client's message handler awaiting acknowledgement.
MSINETMESSAGERETRY	Number of retries for the client's message handler awaiting acknowledgement.
MSINETPACKETTIMEOUT	Timeout value for the client's packet handler awaiting acknowledgement.
MISINETPACKETRETRY	Number of retries for the client's packet handler awaiting acknowledgement.
MSGETHOSTTIME	Timeout value for retrieving host information.
MSHOSTNAME	System host name.

Operating System Related Variables	
Variable	**Function**
MSGETPW	Use `getpw` instead of `getwuid`

	system call to obtain the user id. This is for backward compatibility reasons.
MSGCWDSIGCHLDRESET	Resets the signal for some operating system call `getcwd`.
MSCRAYQDELSLEEP	Number of seconds `empqdel` waits before checking whether a request has died. This is for Cray system only.
MSCRAYQDELRETRY	Number of times `empqdel` attempts to send the termination signal. This is for Cray system only.
MSCRAYQDELSIGNAL	Number or symbolic name of the signal for `empqdel`. This is for Cray system only.
MSMIGRATERETRY	Number of retry to access migrated file. This is used on Cray system only.
MSMIGRATESLEEP	Time interval for the retries to access migrated file. This is used on Cray system only.

Report Writer Related Variables

Variable	Function
MSMWNULLOK	Suppress error on printing null.
MSPAGELENGTH	Default page length for reports.
MSPAGEWIDTH	Default page width for reports.

Empress Internal Variables

Variable	Function
MSSAVECWD	Save the current working directory value.
MSBUFFERRECORDFACTOR	Controls the number of records for the read ahead buffering action.
MSMAXPROCS	Maximum number of processes which can access the lock manager in a lock file.
MSFILELOCKNBUCKETS	Number of hash buckets for a lock manager in a lock file.
MSFILELOCKNLOCKS	Maximum number of locks for a lock manager in a lock file.

MSNRECORDCACHE	Cache size in terms of the number of records.
MSLICENCE	License information.

Network Server Related Variables

Variable	Function
MSUSERAUTHCONFIGFILE	Name of user authorization configuration file.
MSNETSERVERCONFIGFILE	Name of network server configuration file.
MSNETTYPECONFIGFILE	Name of network type configuration file.
MSCONFIGFILEPATH	Path for Network Server Related Configuration Files.

Replication Related Variables

Variable	Function
MSREPLOCKRETRY	Number of retries to obtain a lock on a master table during replication synchronization.
MSREPLOCKSLEEP	Interval between retries to obtain a lock on a master table during replication synchronization.
MSREPENABLUPDATE	Internal Variable that enables a Replicate Table to be updateable.
MSREPPURGEINTERVAL	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.

Appendix 3 - Empress White Paper: The Concept of I=MC^2

This appendix contains summaries and descriptions of the following:

Empress White Paper on the Concept of $I=MC^2$

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 **I**nformation **M**anagement **C**omponent 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 a powerful, reliable and cost-effective embeddable knowledge and rule-based relational data management system. It is designed for high-performance, mission-critical, maintenance-free applications on UNIX, Linux, Real-Time and Windows environments. Such applications are typically developed for Aerospace, Military & Defense, Meteorological, Simulator Design, Satellite Imaging, Network Management, Electronic Publishing, Telecommunications and Process Control.

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.

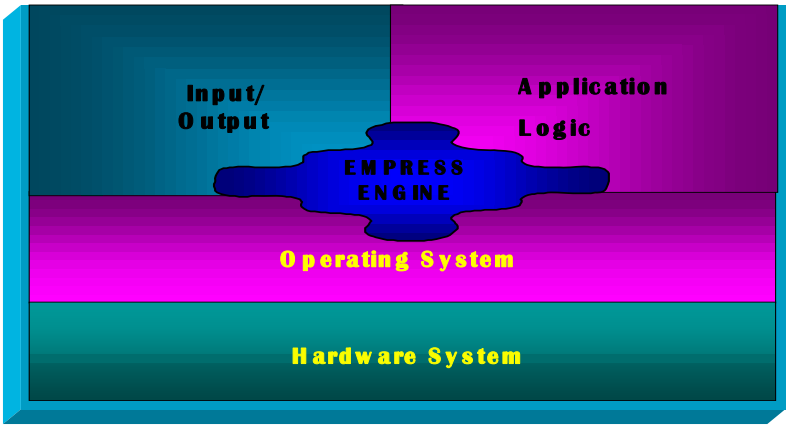
The ***Empress*** Data Management System uses the O/S file system as its native file system while adopting the O/S operational environment as its standard environment. Together with its open architecture, the ***Empress*** Data Management System becomes a functional extension of the O/S file system. ***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)
```

```
save_name              nlschar(32,1)
default_expression     nlstext(20,42,64,1)
range                  nlstext(20,42,64,1)
```

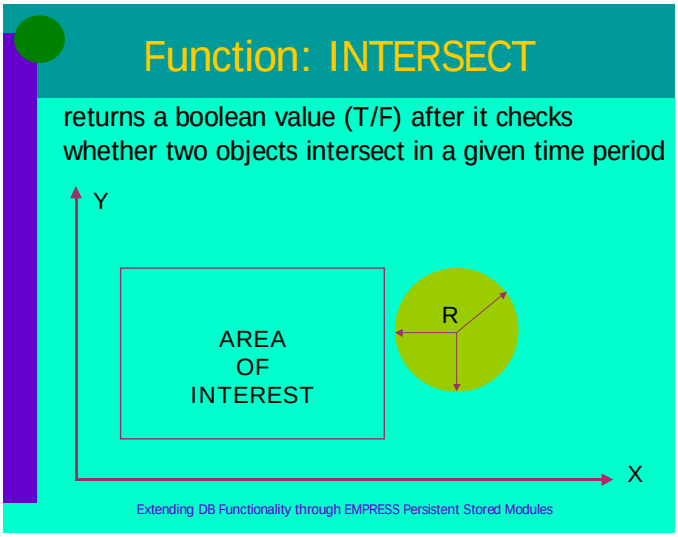
enter_script and *exit_script* store 4GL logic (programs) which are executed when the field is entered or exited. In a similar fashion, database tables are defined for other objects to keep track of and provide meaning for:

- 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:

```
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:

```
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

1] National Communications System Technology & Standards Division, "Telecommunications: A Glossary of Telecommunications Terms - Federal Standard 1037C", General Services Administration Information Technology Service, August 7, 1996.

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:

[2] S. Savchenko, "Extending Database Functionality Through Persistent Stored Modules", Seventh Workshop Proceedings on Meteorological Operational Systems, European Centre for Medium-Range Weather Forecasts, November 1999.

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:

[3] Empress Software Inc, "Empress V8.60 Manual Set", Empress Software Inc, April 2000.

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

[4] N. Wong, "Beyond Client Server", Empress Software Inc. White Paper, 2001.

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

White Paper

Contents

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.

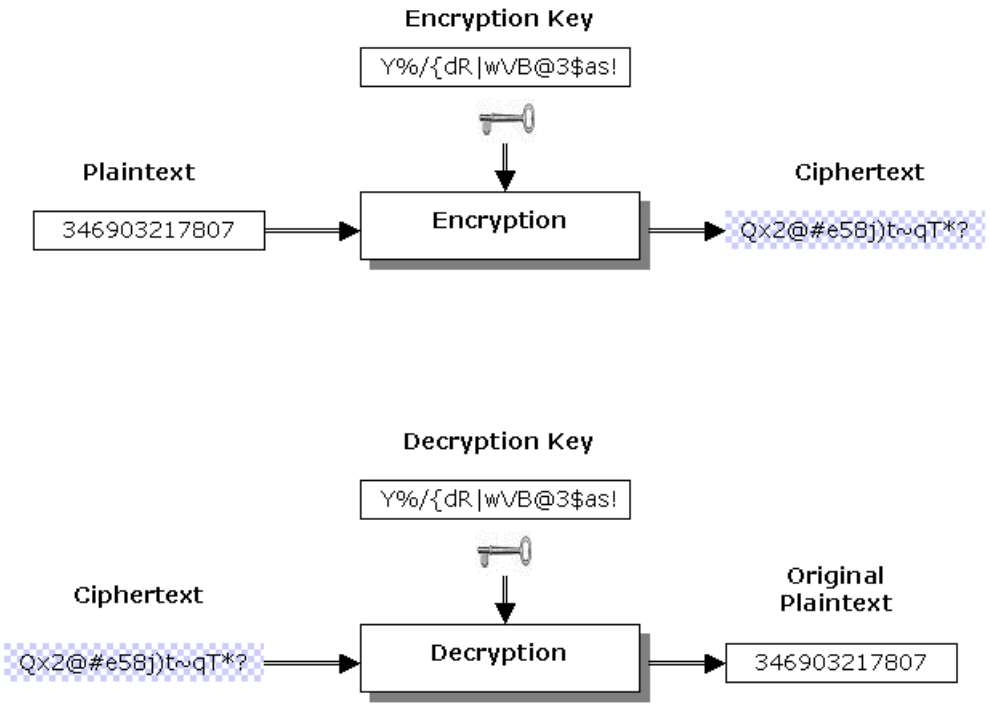


Figure 1. Encryption and Decryption

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 **libgcrypt**, 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 at the ODBC and JDBC applications in **Figure 2**, the Security Adapter is linked to the Empress Connectivity Server.

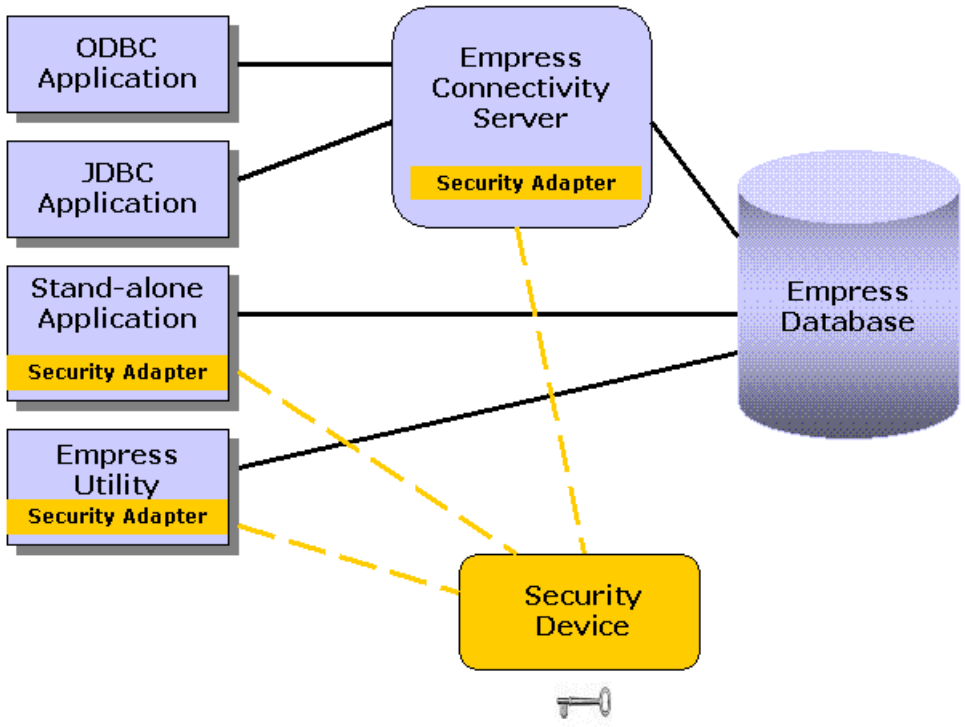


Figure 2. Concept A: Database Encryption Concept with a Security Device

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.

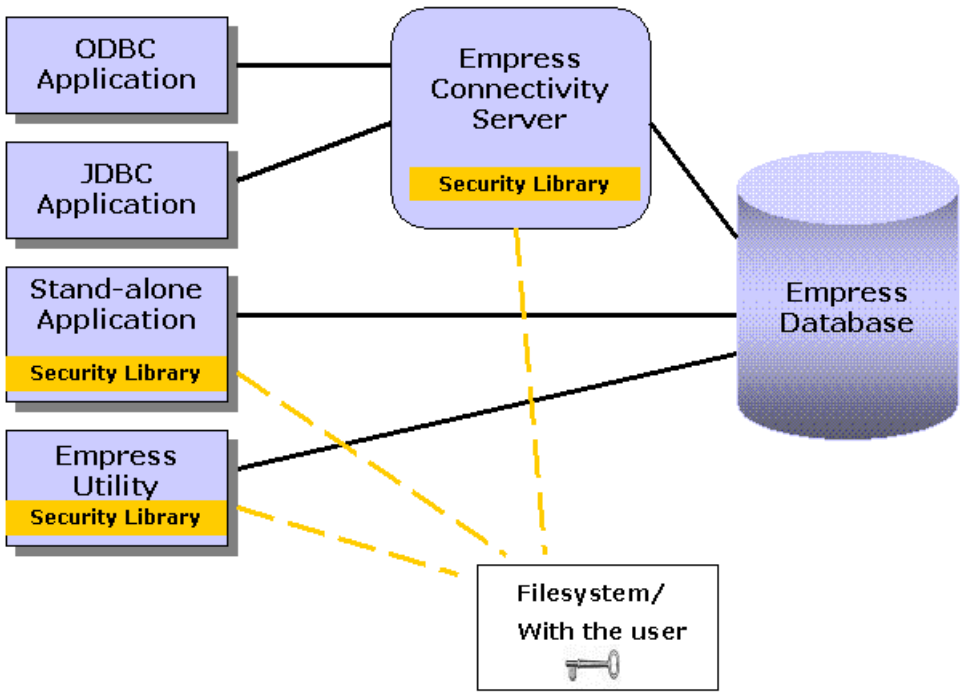


Figure 3. Concept B: Database Encryption Concept with a Security Library

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.

Lets 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:

```
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:

```
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:

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

Or to define the encryption on the column again:

```
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.

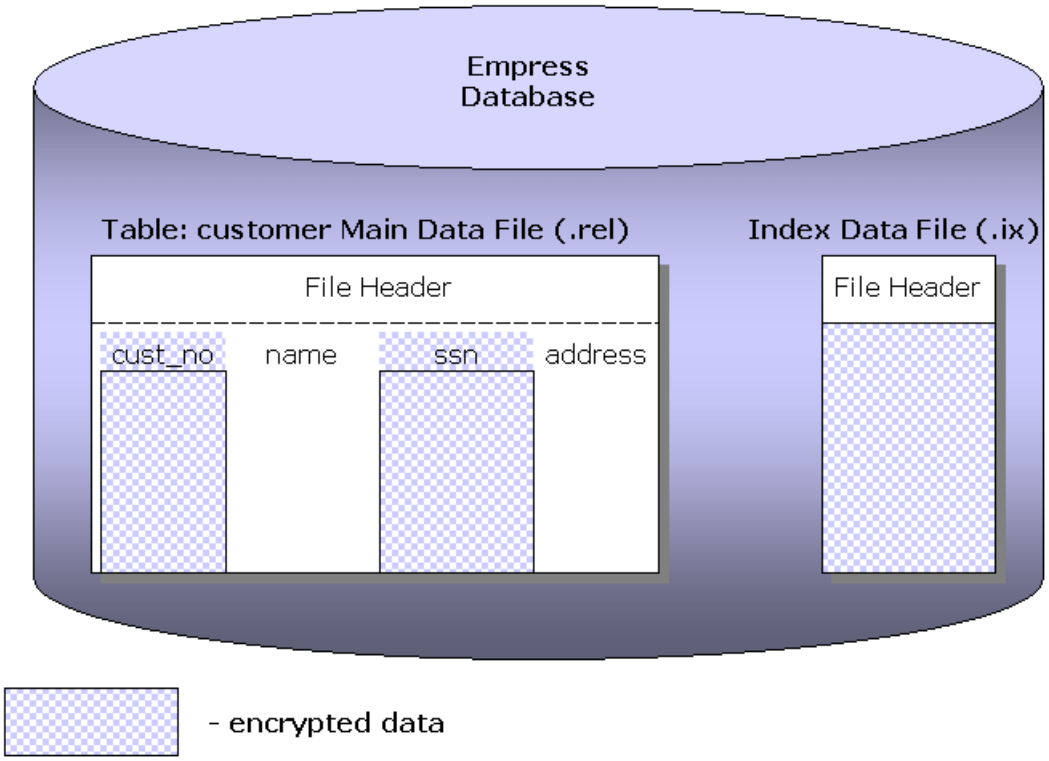


Figure 4. Encrypted and Unencrypted Parts of the Empress Database Table **customer**

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.

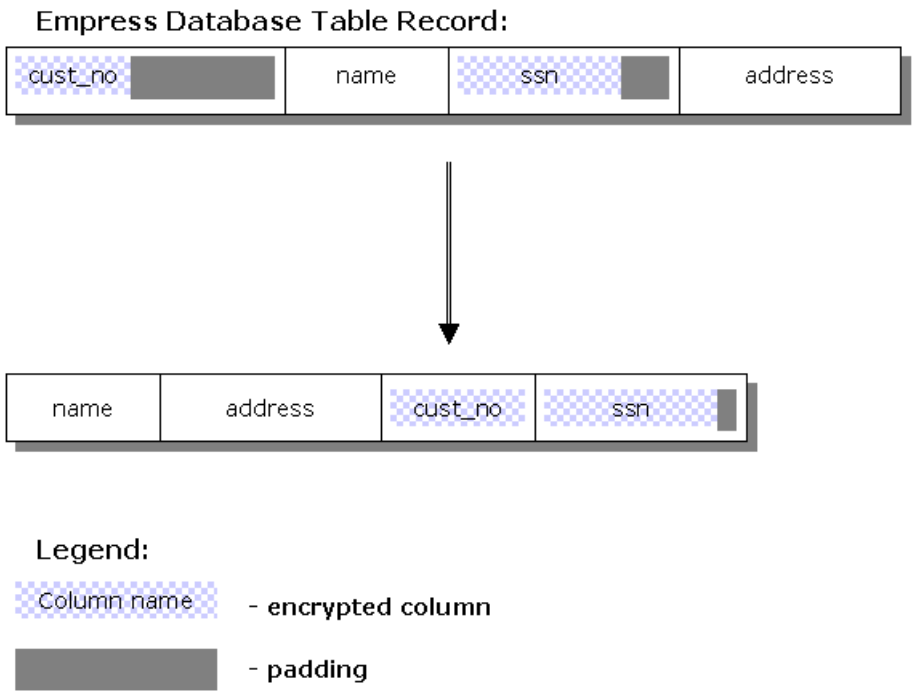


Figure 5. Internal Empress Record Layout Optimization

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:
   cust_no                longinteger Not Null Encrypted
   name                   character(20,1)
   ssn                    character(9,1) Encrypted
   address                text(20,1024,1024,1)

 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

[BRUCE] Bruce Schneier: Applied Cryptography (Second Edition), John Wiley & Sons, 1996 ISBN 0-471-11709-9

[FIPS] FIPS PUB 140-2, Security Requirements for Cryptographic Modules, NIST, 2001.

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

[INGR] www.ingrian.com

[GNUPG] The GNU Privacy Guard, www.gnupg.org

[NAE] NAE Developer Guide for the PKCS #11 Provider, Ingrian Networks, 2005.

Index