Raima Database Manager Server 8.4 Architecture and Features

By Wayne Warren, CTO – September 2013

Abstract

This White Paper contains an overview of RDM Server 8.4. It will discuss the features and characteristics of the database, its performance-enhancing features and application programming interfaces (APIs), and its administration facilities. It will detail the supported operating systems and network transports. Finally, it will describe additional products and services Raima offers to assist with database development.

The White Paper will be of interest to the embedded developer interested in gaining a deeper understanding of the RDM Server embedded database.

Prerequisites: A basic understanding of application software development.

CONTENTS

1. PRODUCT OVERVIEW

The Raima Database Manager (RDM) Server is a high performance client/server database designed for applications with demanding server based performance requirements for industrial, commercial and general line-of-business applications such as e-business, web applications and Internet infrastructure. RDM Server is scalable and provides a rich set of architectural choices and APIs, including ANSI SQL, ODBC C-API, low-level C-API, and support for custom APIs. Unlike typical relational client/server database products, RDM Server supports both relational and pointer-based network model databases in any combination, as well as application specific processing on either side of the client/server equation. The choices of multiple operating platforms, APIs, processing localities (client or server), and database models can be combined to satisfy the functional or performance requirements of virtually any application.

RDM Server has always offered a strong foundation for application development with unique tools for performance enhancement and database customization. RDM Server 8.4 continues this legacy by adding critical requirements for many users and developers of modern database applications. RDM Server 8.4 is compliant with the latest industry database standards such as SQL, ADO.Net, PHP, ODBC and updated support for JDBC v4.0. In addition, RDM Server adds active-passive replication for applications requiring fault tolerance and high availability. Coupled with Symmetric Multi-Processing (SMP) support and the powerful Application Server Technology the latest RDM Server is the most flexible and powerful version to date.

A quick list of some of RDM Server's capabilities is given below. Detailed descriptions are provided in the sections that follow.

- User login and security
- Multiple network protocol support: TCP/IP, named pipes, shared memory.
- Heterogeneous client support (i.e., data returned from server in client computer's native format).
- Multiple API support: Raima's native navigational (Core-level) API, ODBC, JDBC, and ADO.NET.
- Server extension modules: application-specific, client callable C/C++ based server procedures.
- "Full" SQL—support for many of the ANSI/ISO SQL standards including integrity checks, declared foreign and primary keys, extended join syntax and triggers. Non-standard features include the **CREATE JOIN** statement which maps foreign and primary key related tables to network model sets that result in optimal query join performance.
- SQL C-based, server-resident user-defined procedures and functions.
- Hot online backup API allows the application to control the backup operation and incrementally free database files from hot backup mode.
- Database replication to other RDM Servers
- Ability to enable full data transmission and storage encryption.
- High performance transaction processing support with row-level locking.
- Scalable, multi-threaded server architecture.
- Full administration API that allows all database administration to be performed by the application. No database administrator is needed and the end user does not even need to know that there is a Raima database present.
- Ability to directly link RDM Server with the application in order to form an application server. This eliminates all of the overhead associated with inter-process communication between the application program and the database server.
- Ability to dynamically alter a database definition. You can new tables and indexes or change existing tables to add new columns.

2. WHAT'S NEW IN RDM SERVER 8.4?

2.1 Performance Customization with Multiple Indexing Methods

Hash-indexing in tables with large volumes often provides faster access to data than b-tree indexing and gives developers the versatility to enhance the performance of their applications. The different indexing methods can be used together within an application to optimize database access performance.

2.2 Easily Create ADO.NET Applications

The ADO.NET Data Provider enables developers to easily create .NET applications with the RDM Server database. The Data Provider for RDM Server derives from the classes in the System.Data.Common namespace instead of directly implementing the IDb* interfaces in System.Data, as was the case in version 8.3. This is easier-to-use and follows the current standard for ADO.NET Data Providers.

3. COMBINED DATABASE MODEL

RDM Server enables developers to combine the relational database model with the pointer-based network database model, using RDM Server' **CREATE JOIN** extension to the ANSI SQL Data Definition Language (DDL).

CREATE JOIN implements a pre-defined join between multiple tables, eliminating the need for an external index table to define the relationship. With **CREATE JOIN** the relationship between two tables are defined and maintained through direct access pointers. These pointers create one-to-many "sets" between tables. Sets are physically implemented with linked lists of pointers maintained in the system data area of each stored row.

RDM Server Database Server allows you to use conventional relational foreign key/primary key joins, **CREATE JOIN**, or a combination of both. The ability to combine network and relational database technologies provides database application developers with the following advantages:

- Tables and rows are accessed directly
- Join processing performance is optimal
- Indexes are used only when needed
- Database size can be reduced as fewer indexes are needed
- More complex database designs are supported
- Referential integrity checking is more efficient

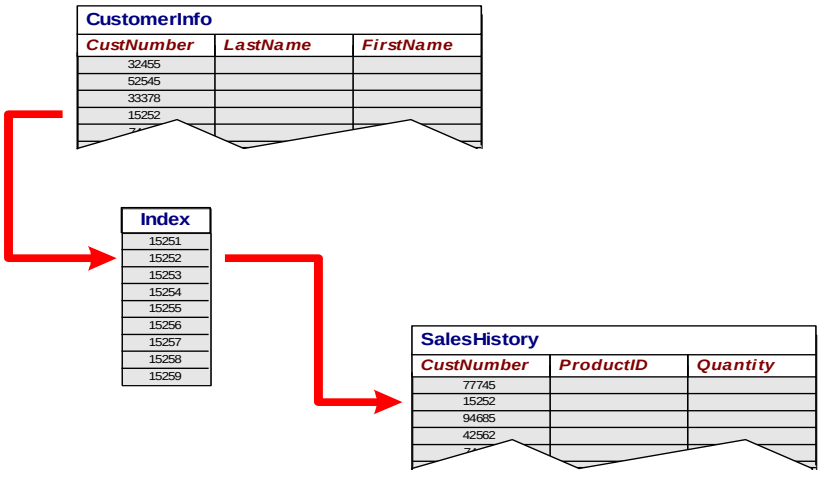


Figure 1: Relational Database Model

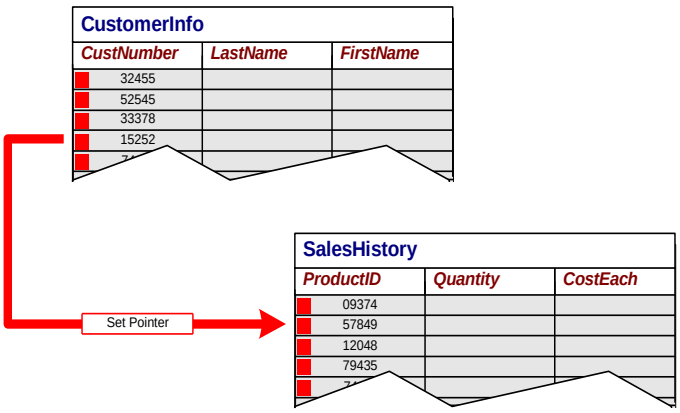


*Figure 2: Relational model using RDM Server' **CREATE JOIN** (note absence of index and redundant CustNumber column)*

The **CREATE JOIN** statement guarantees that only a single logical disk access is needed to retrieve the related row in the referenced table. This means that the performance of referential integrity checking and **SELECT** statement join processing will be optimal.

CREATE JOIN also guarantees optimal performance in locating all of the rows of the tables with a particular foreign key value. This ensures that retrieval can occur from either the many-to-one or the one-to-many direction (thus supporting both inner and outer join processing). These pre-defined joins can be ordered so that the referencing (foreign key) tables' rows for each foreign key value are sorted by the specified columns.

4. CLIENT/ SERVER ARCHITECTURE

The RDM Server SQL system resides on a server as a tightly integrated layer over the RDM Server runtime system. All of the RDM Server SQL database operations are implemented using the standard RDM Server low-level C-API function calls.

An RDM Server SQL-specific client library containing all of the RDM Server SQL C-API function calls is linked with the client application program. These functions interface to the RDM Server Remote Procedure Call (RPC) subsystem. The RDM Server SQL system is re-entrant and uses multiple threads.

RDM Server includes several modules that are distributed between a client and the server. The Multi-protocol Network Communications Processor (MNCP) handles all communications between the client and server. Each of the components on both the client and server are briefly discussed below.

4.1 Server Components

- **Database Management System**—Implements the database runtime system. A true multi-threaded runtime built using asynchronous input/output and operating system threads providing good transaction throughput under heavy loads. The caching scheme and the transaction manager are designed for transaction-oriented processing.
- **Multi-transport Network Communications Processor (MNCP)**—Handles communication between the server and clients accessing the server. The server MNCP maintains queues where client requests are held, and where responses are returned to the client. MNCP launches a number of listening threads depending on the number of configured transports.
- **SQL API—**Includes functionality to store and retrieve data in response to client's SQL requests. Consists of the SQL language parser, query optimizer, code generator, and SQL statement execution interpreter.

- **Server Extensions (Custom APIs)**—Implements high-level server functions for client support and server administration. User-defined modules specific to the client application(s) can be built as Server Extensions.
- **RPC Module—**(or RPC Server), in coordination with the MNCP, makes sure that client requests are processed efficiently. It includes the packet interpreter, scheduler, and command dispatch functions. It also contains a Data Portability Layer (DPL) that implements marshaling/de-marshaling of data in heterogeneous environments.

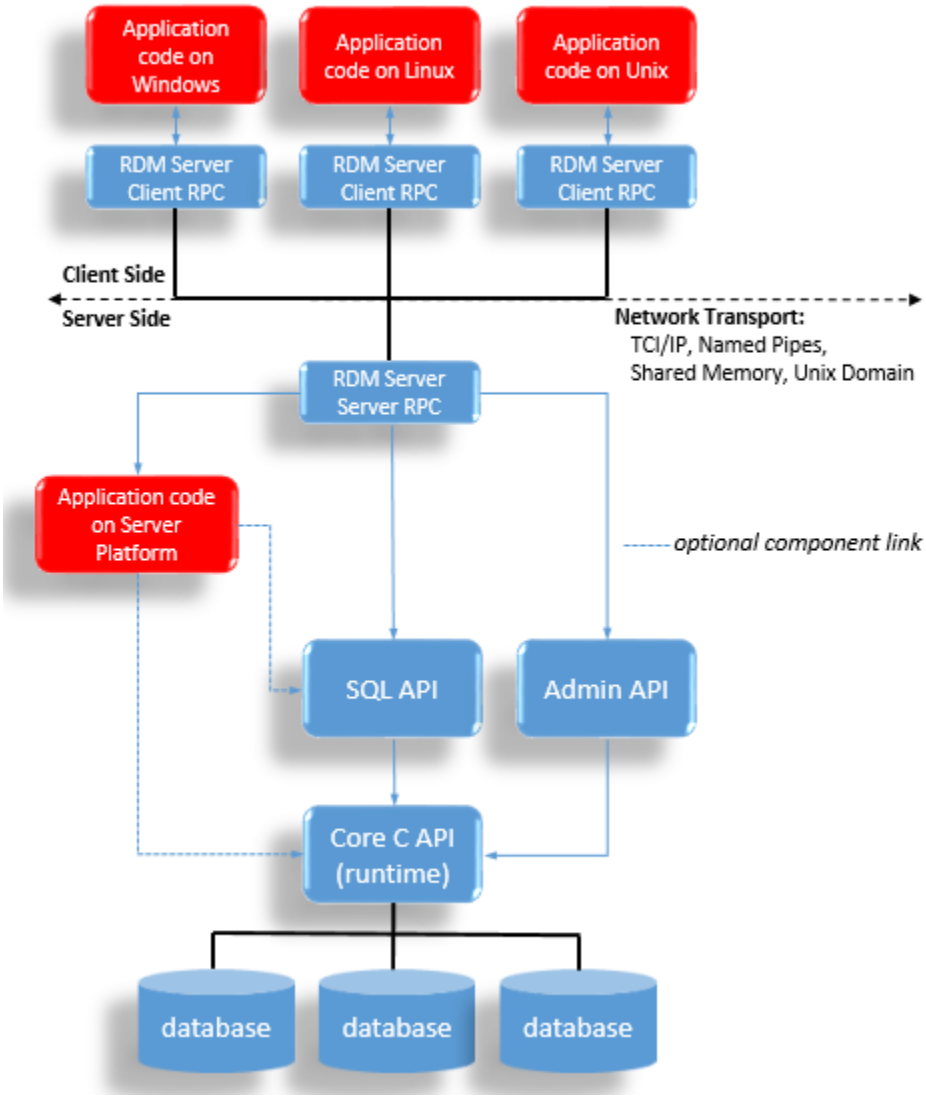


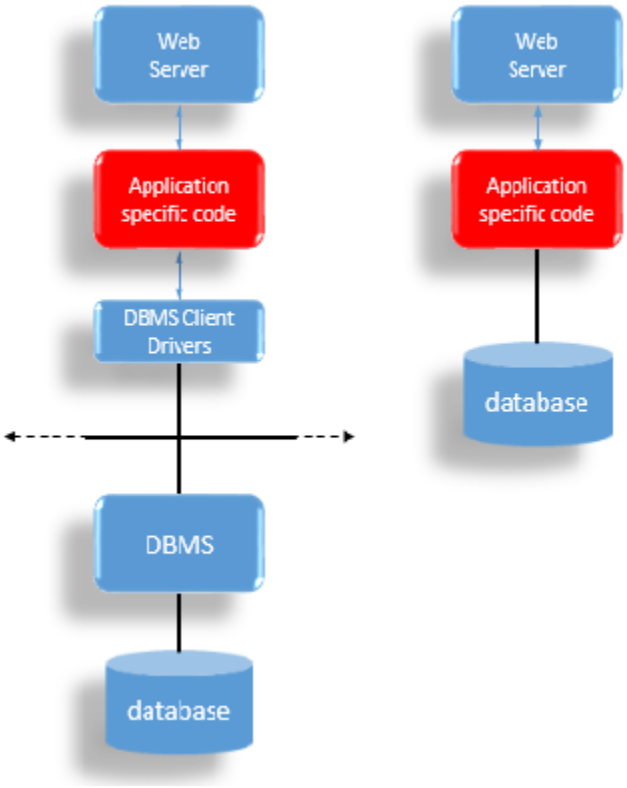
Figure 3: RDM Server Architecture Diagram

4.2 Client Components

- **Client Library—**presents the server APIs to the client application. Client libraries are linked dynamically.
- **Client Multi-transport Network Communications Processor—**contains the client stub of the server mechanism and interfaces to the client network protocol stack. The MNCP allows clients to communicate with servers using multiple transports. For example, the same client could have one open session with a UNIX server via TCP/IP and another with a Windows server using Named Pipes.

4.3 RDM Server Application Server Technology

The RDM Server Application Server Technology provides a powerful alternative for using RDM Server as an embedded database and as an Internet/intranet database system. With this new architecture, developers can directly link applications with the RDM Server database engine, without having to communicate through a communications "layer." In effect, the application becomes the server.



Fidure 4: Comparison of Client/Server vs. RDM Server Application Server

The RDM Server Application Server Technology (RAST) lets the developer select the appropriate remote communication protocol. For a single application, it is possible to use any protocol provided by the operating system or by a third-party supplier. This communication can be controlled by the application, completely outside of RDM Server. In order to benefit from the multi-user/multi-threaded RDM Server environment, an application can launch and manage its own threads, each thread controlling one or more RDM Server login sessions. When a RAST application launches RDM Server, through the s_startup function, it can have the RDM Server start its RPC/MNCP by a subsequent call to the startRPCThreads function. This gives any standard RDM Server client program, such as third-party ADO.Net, PHP, ODBC or JDBC tools, access to the RDM Server database. Applications that require the power and performance of a full multi-threaded/multi-user database engine, will benefit by linking directly to the server process, thus avoiding the performance and memory penalties of accessing a separate process (the RDM Server) through the MNCP.

For example, an Internet web-server application using the RDM Server Application Server Technology has a distinct advantages in comparison to a competing client/server DBMS. A comparison of these two types of architecture is illustrated above. This does not prevent normal RDM Server clients connecting to the Application, now acting as the database server, to operate in the traditional way.

The left side of the above diagram shows that in order to access a typical database management system, the Internet web server must use a server extension to connect to a separate database server, using the usual network protocols

supported by that DBMS. The right side illustrates the architecture possible with the RDM Server multi-threaded database engine, which is installed on the same computer and is directly linked to the Internet web-server, providing more efficient and faster access to the database information. Most web server software development kits, such as Microsoft's Internet Information Server or the Apache Server, provide the ability to develop server extensions that can be integrated with the database via the RDM Server Application Server Technology for dramatically improved scalability and performance.

5. RDM SERVER C API

Every solid building is based on a strong foundation and RDM Server' foundation is its low-level C – API, also referred to as the micro-kernel, record-level or core API. The RDM Server C-API provides developers with greater flexibility in modeling real world problems and provides more "hooks" for controlling the database server with more precision than is possible using the relatively high-level SQL interface. Like the SQL API, the RDM Server C-API includes both a database definition language (DDL) and a rich set of database manipulation functions.

5.1 Non-SQL Database Definition Language (DDL)

An RDM Server database design is specified using its Database Definition Language. There is one DDL specification file for each RDM Server database. In general, there is one database per application, although some applications may require the use of several databases. A DDL specification identifies the database, defines the files that comprise the database, and contains declarations for all record types, data and key fields, and relationships that are to exist in the database. The DDL specification is created using any text editor and is stored in a text file. Input is free form, with comments specified, as in C, between **/* */** pairs. Identifiers are used to name the database, files, records, fields, and sets.

A RDM Server database can be defined using RDM Server' proprietary database definition language. Though the DDL is proprietary, it is patterned after the C programming language and will be very intuitive to programmers familiar with C/C++. Below is an example of a database definition utilizing the C DDL. A RDM Server C DDL is processed using the **ddlproc** utility, which can be run from any client machine. This utility compiles the source file, generates the database files, and registers the database within the RDM Server system catalog.

Example of a RDM Server C (Non-SQL) DDL:

```
database tims {
   data file "tims.d01" contains key_word, intersect;
   data file "tims.d02" contains author, borrower, info, text;
   key  file "tims.k01" contains id_code;
   key  file "tims.k02" contains name, friend, word;

   record author {
      unique key char name[32];     /* author's name: "last, first" */
   }                                /* or editor's name */
   record info {
      unique key char id_code[16];  /* dewey dec. or own coding tech. */
      char info_title[80];          /* title of book, article, mag. */
      char publisher[32];           /* name of publisher - prob. coded */
      char pub_date[12];            /* date of publication
```

```
                                    (e.g. most recent copyright) */
   short info_type;              /* 0 = book, 1 = magazine, 2 = article */
}
record borrower {
   key char friend[32];          /* name of borrower */
   long date_borrowed;           /* dates are stored initially as */
   long date_returned;           /* numeric YYMMDD (e.g. 870226) */
}
record text {
   char line[80];                /* line of abstract text */
}
record key_word {
   unique key char word[32];     /* subject key words or classification */
}
record intersect {
   short int_type;               /* copy of info_type to save I/O */
}                                /* when looking only for, say, books */

set has_published {
   order ascending;
   owner author;
   member info by info_title;
}
set article_list {
   order last;
   owner info;
   member info;
}
set loaned_books {
   order last;
   owner info;
   member borrower;
}
set abstract {
```

```
      order last;
      owner info;
      member text;
   }
   set key_to_info {
      order last;
      owner key_word;
      member intersect;
   }
   set info_to_key {
      order last;
      owner info;
      member intersect;
   }
)
```

5.2 Non-SQL Database Manipulation

The Core Level or low-level C–API Database (a derivative of Raima's original database product, RDM, or Raima Database Manager) also includes a set of C functions that provides the lowest and most efficient access and control over a database. This interface consists of a C function library, with functions for navigate through the database, reading and writing to records, adding and deleting records, and performing other tasks on a record by record basis. This library is the foundation on which other interfaces are built—for example, the RDM Server SQL API was written using the native API interface.

The following functions/capabilities add even greater power and flexibility to the RDM Server C-API:

- **Custom Compare Function—**A RDM Server distinctive feature **d_fldcmp**, which allows an application to compare two values using comparison logic specific to a particular database field.
- **Custom Data Types—**the RDM Server C–API provides support for custom 'C' data types including structures and arrays. This is a feature not found in competing databases and is one of many reasons that RDM Server is better choice for modeling real life problems.

Example of C–API functions in C application source code:

```
char vma_desc(25);              /*variable to contain vehicle make      */
struct fleet f;                         /*variable to hold a fleet record*/

  …   /*fleet record entered by user    */

/* validate correct vehicle make code  */
if (d_keyfind(VMA_CODE, f.vma, hDb) ==S_NOTFOUND)
  entry_error("invalid vma code");

else{
  /*enter fleet record */

  d_fillnew(FLEET, &f, hDb);

}

…
```

6. RDM SERVER SQL

RDM Server SQL is an implementation of the ANSI SQL-89 Level 2 and most of the ANSI SQL-92 standards, providing a rich set of commands to create, access and manipulate SQL databases. The SQL DML is itself implemented as an extension of the RDM Server database server and is an excellent example of how the server can be powerfully extended to meet developers' requirements.

6.1 SQL Database Definition Language (DDL)

RDM Server's SQL Data Definition Language (DDL) uses standard SQL for defining the database structure. It provides server-based declarative referential integrity, implemented through the system catalog and is universally enforced, in compliance with the ANSI 1989 Level 2 Integrity Enhancement Addendum. A RDM Server SQL DDL tool, **sddlp** is used to compile RDM Server SQL database definitions and store the definition information in the RDM Server SQL system catalog.

Example of "inventory" database SQL DDL text file:

```
create database inventory;

create table product

(

     prod_id           smallint    primary key,

     prod_desc         char(39)    not null,

     price             float,

     cost              float,

     prod_pic          long varbinary,

     description       wvarchar(120)

);

create unique index prod_key on product(prod_id);
```

```
create table outlet
(
     loc_id            char(3)     primary key,
     city              char(17)    not null,
     state             char(2)     not null,
     region            smallint    not null
);
create unique index loc_key on outlet (loc_id);

create table on_hand
(
     loc_id            char(3)     not null references outlet(loc_id),
     prod_id           smallint    not null references product,
     quantity          int         not null
);
create join inventory order last on on_hand(loc_id);
create join distribution order last on on_hand(prod_id);
```

RDM Server supports standard SQL data types including fixed point precision decimals and allows columns to have null values. Other notable features of RDM Server SQL DDL are:

- **Rowid Primary and Foreign Keys—**providing optimal, direct access row and join retrieval.
- **Auto-increment Column Attribute (AutoNumber)—**supports the declaration of integer columns that are automatically assigned a unique number when a new row is inserted.
- **Full automatic referential integrity checking—**The ANSI ’89 table and column constraint features have been fully implemented in RDM Server SQL. All referential integrity checking as defined by the ANSI ’89 foreign and primary key declarations are available in RDM Server. In addition, RDM Server SQL DDL includes an SQL extension statement, **CREATE JOIN**, that is used with foreign and primary key specification to indicate direct access methods between related tables.
- **Binary Large Object (BLOB) Support—**for columns of type **LONG VARCHAR**, **LONG WVARCHAR**, and **LONG VARBINARY**.
- **International Character Set Support—**SQL DDL data type, **WCHARACTER** (or **WCHAR**) are available in RDM Server to support wide-character data or double-byte fields (Unicode for Windows). Double-byte field support in RDM Server allows developers to create international versions of their applications or to localize existing RDM Server applications for specific international markets.
- In addition to the **WCHAR** data type, RDM Server also uses a standard local setting in the configuration file to apply specific collating sequences to both **CHAR** and **WCHAR** data. For backward compatibility, the traditional country table method is also supported.

- **Custom Compare Function—**allows columns to have a specialized comparison function that can either be the system supplied nocase comparison (case insensitive) or a user-defined comparison function that can be used, for example, for a national language. Comparison functions are used to define the natural ordering for the character data stored in the column.
- **Very Large Database (VLDB) Support—**very large databases are supported by allowing the size of the database address (which specifies the physical file location of a row) to be specified as up to 10 bytes in length which gives a theoretical limit of 18 quintillion rows per table.
- **Dynamic DDL—**support for **ALTER DATABASE** allowing tables and indexes to be added to or dropped from an existing database and **ALTER TABLE** which allows columns to be added or dropped.
- **Dynamic CREATE TEMPORARY TABLE—**allows creating of temporary tables which can be used to stored intermediate query results which can then be used in subsequent queries.

6.2 SQL Database Manipulation Language (DML)

All of the basic features defined in SQL for querying, inserting, updating, and deleting rows are supported. Some of the most notable features of RDM Server' DML are:

- **Multiple connections to one or more servers**—RDM Server SQL allows any number of databases to be open and active at the same time, and enables a single client application to open several independent, active connections to one or more RDM Server servers.
- **Database security and encryption—**RDM Server SQL supports the ANSI database security capabilities including both **GRANT** and **REVOKE**. In addition, RDM Server supports data encryption at the storage level as well as at the communication level. The extensible RDM Server architecture allows users to supply proprietary encryption modules to implement storage and communication data security.
- **Cost-based Query Optimizer—**RDM Server SQL uses a sophisticated, cost-based query optimizer that uses data distribution statistics produced by the **UPDATE STATISTICS** statement to generate an efficient execution plan for each **SELECT** statement. As not optimizer can guarantee that the best plan has been generated, RDM Server SQL also provides some additional non-standard features that give the user some control over the choices available to the optimizer to ensure that it makes the best access method choices in those rare situations where the default plan is not optimal.
- **Transaction processing—**RDM Server SQL provides full transaction processing capabilities including the ability to do partial rollbacks. RDM Server' transaction processing provides precise transaction control, including the ability to **START**, **COMMIT** or **ROLLBACK** a complete transaction or partial **ROLLBACK** to a previous **SAVEPOINT**.
- **Stored procedures**—RDM Server SQL stored procedures allow SQL statements to be compiled, optimized, and stored in the system catalog. RDM Server' stored procedures provide a significant performance advantage by reducing the bandwidth required for SQL statements sent from the client to the server, and eliminating the time needed to compile the procedure's SQL statements each time they are executed.
- **Triggers**—Unlike stored procedures, which must be called by an application in order to execute, triggers automatically execute in response to a database event. Triggers are commonly used to enforce business rules and are executed based on a change in the value of a specified column(s). Triggers are defined using the standard SQL **CREATE TRIGGER** statement.
- **Read-only (Static) Scrollable Cursors**—implemented in RDM Server 7.0 and function as specified in the ODBC level 2 specification, allowing client side caching (snapshots) of row sets.
- **Searched and positioned updates and deletes—**Positioned updates and deletes are used in conjunction with the RDM Server SQL C function interface.
- **User-defined functions (UDFs)—**UDFs are user programmed, server resident C functions that are called from RDM Server SQL when the UDFs are referenced in an expression in a SQL statement. They can be used to extend the scalar and/or aggregate functions for use by the application.
- **User-defined stored procedures (UDPs)—**UDPs are user programmed, server resident C modules that are called from RDM Server SQL when a **CALL** statement that references the UDP is executed. UDPs can return one or more result sets (identical to those returned by **SELECT** statements) allowing non-SQL managed data to be returned as tables.

6.3 SQL API

Raima's commitment to standards ensures application portability and interoperability. RDM Server implements a SQL C-API that is syntactically compatible with the ODBC functional specification. RDM Server' SQL processor provides more than 66 documented functions to control, access, and manipulate the database. A partial list of RDM Server SQL API functions is provided in the table below:

Table of RDM Server SQL API functions:

BCDAdd	BCDAllocEnv	BCDCompare	BCDDivide
BCDFreeEnv	BCDMultiply	BCDPack	BCDStatus
BCDSubtract	BCDUnpack	SQLAllocConnect	SQLAllocEnv
SQLAllocStmt	SQLBindCol	SQLCancel	SQLColAttributes
SQLColuimns	SQLConnect	SQLConnectWith	SQLDbaToRowId
SQLDBHandle	SQLDescribeCol	SQLDescribeStmt	SQLDisconnect
SQLDriverConnect	SQLError	SQLExecDirect	SQLExecute
SQLExtendedFetch	SQLExtendedTransact	SQLFetch	SQLFreeConnect
SQLFreeEnv	SQLFreeStmt	SQLGetConnectOption	SQLGetCursorName
SQLGetData	SQLGetFunctions	SQLGetInfo	SQLGetStmtOption
SQLGetTypeInfo	SQLMoreResults	SQLNativeSql	SQLNumParams
SQLNumResultCols	SQLParamData	SQLPrepare	SQLProcedures
SQLPutData	SQLRowCount	SQLRowDba	SQLRowId
SQLRowIdToDba	SQLSessionId	SQLSetConnectOption	SQLSetCursorName
SQLSetParam	SQLSetScrollOptions	SQLSetStmtOption	SQLSpecialColumns
SQLStatistics	SQLTables	SQLTransact	SQLTransactTrigger
SQLWhenever	SYSDbaToRowId	SYSDBHandle	SYSDescribeStmt
SYSMemoryTag	SYSRowDba	SYSRowId	SYSRowIdToDba

7. ZERO DATABASE ADMINISTRATION

One of the high value characteristics of RDM Server that makes it unique among database servers is its ability to be embedded within an application, with database configuration and administration completely controlled by the application, so that end-users of RDM Server-based applications are presented with virtually no administrative or maintenance requirements. This considerably lowers the end-user's total cost of ownership (TCO), which can be an important consideration when purchasing software applications.

7.1 Administrative API

The RDM Server Admin API is a collection powerful and useful database administration utilities and functions. The Admin API is exposed to developers, allowing them to programmatically configure and run these database utilities and/or include them within their database application.

The RDM Server API offers a rich set of administrative functions to support application level administration of the server, allowing the database system to perform the administrative tasks automatically, without having to launch special administration utilities.

The administrative API consists of a family of approximately 50 documented function calls for such purposes such as configuration, adding new users, moving database files, performing hot backup, and getting/setting parameters.

Table of RDM Server Administration Level Functions:

d_taskinfo	s_configGet	s_configModify	s_dbAdd	s_dbClone
s_dbDel	s_dbGet	s_dbInit	s_dbModify	s_dbUserAdd
s_dbUserDel	s_dbUserGet	s_devAdd	s_devDel	s_devGet
s_devModify	s_emAdd	s_emDel	s_emGet	s_emModify
s_familyAdd	s_familyDel	s_familyMemAdd	s_familyMemDel	s_familyModify
s_getMinFreeSpace	s_iniGet	s_iniGetPrivate	s_iniSet	s_iniSetPrivate
s_login	s_logout	s_ping	s_setMinFreeSpace	s_showDbFiles
s_showDbs	s_showDbUsers	s_showDevs	s_showEms	s_showFamilyMems
s_showFamilys	s_showServers	s_showUserDbs	s_showUsers	s_shutdown
s_statistics	s_userAdd	s_userDel	s_userGet	s_userModify

7.2 RDM Server Administration

- **Administration Tools—**RDM Server also supports ad hoc administrative activities by the end-user via an integrated administration utility for managing one or more server installations. The utility's graphical user interface makes it easy for the system administrator to:
    - Create, modify, delete, and query server objects, including users, databases, Server Extensions, and database devices
    - Start and shut down the server and perform backups
    - Change all server and client configuration parameters and default settings
    - Display server performance statistics
    - Perform file management, including copying files to and from the server, deleting files on the server, and copying or moving files on the server

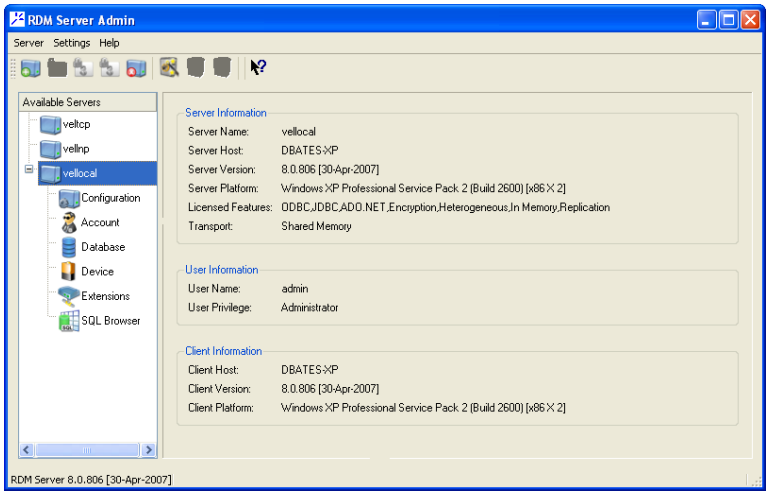


Figure 5: Administration Utility Windows Interface

- **Client Recovery—**In a networked environment, failure can occur if a user disconnects from the network or turns off a PC during a database transaction. RDM Server monitors client activities and performs client recovery when a client process abnormally terminates.
- When a client process terminates unexpectedly the server will detect its absence. The server will then automatically abort any active transactions, release locks, close the database files, and log the client out.
- **Hot On-line Backup—**RDM Server supports hot on-line backup, or backing up the database server while reads, writes, and other database activities continue. While in hot backup mode, the RDM Server database files are kept in a static, transaction consistent state, allowing any backup utility to back up the files. Alternatively, a developer can incorporate backup capabilities into an application, using the new hot backup administration functions. While in hot backup mode, RDM Server stores all changed database pages in a separate "hot" file. When hot backup mode has ended, the pages from the hot file are gradually migrated back to the database files as they are referenced.

7.3 Database Utilities

RDM Server containing the following utilities:

- **DBstat—**Database space consumption statistics and analysis utility
- **DBReplay** - Use of database families for controlling backups and roll-forward recovery have been eliminated in RDM Server and replaced by system-wide hot on-line backup (described above) and the **DBReplay** utility. **DBReplay** is a stand-alone utility that is used to replay the change log files created after a full system backup has finished. **DBReplay** ensures that the database files match the change logs so that the correct change log files are reprocessed in the correct order. User control over replay termination is available on a date/time basis.
- **DBCheck** functionality is implemented in a Server Extension. The Server Extension is invoked by a new API function named **dbcheck**. RDM Server also provides a client program named **DBCheck**, which will call the API function. This utility inspects data and key files for consistency, verifies that all set linkages are correct, and reports any database inconsistencies.
- **Keybuild** is a Server Extension, a new API function (named **keybuild**), and a client program to call the function. This utility is used to re-create key (index) files. Keybuild is useful for implementing DDL changes where non-key fields are changed to key fields and vice-versa.

8. REPLICATION

High Availability is a broad term that can mean many things to many people. For the most part though, people agree that it should describe the overall accessibility of data to the user. With this in mind, HA can encompass the ability of your system to be “always up” in a 24x7 manner; in the telecom industry, HA is measured in percentage availability (five nines or better). HA can also include how your system reacts to internal component failure (fault tolerance). And finally, it can describe the level of responsiveness of your system. A system that is “always up”, but not very responsive (a query takes hours to complete) cannot truly be described as “Highly Available”.

Why is high availability important?

- In today’s “Always Up” business environment, where customers demand instant response, and expect your services at their convenience, the cost of downtime can conservatively run into the thousand of dollars per minute.
- In addition, access to data can be a company’s lifeline; data is the primary ingredient of business intelligence and many of the activities that surround the business. Consequently, the moment access is lost, the business suffers and expensive resources become inefficient or unavailable.

Raima Technology, Inc recognizes the importance of high availability but a complete HA solution involves much more than just database availability as hardware components need to be controlled as well. The DBMS' role in an HA solution is to maintain multiple, identical copies of the database so that in the event the one becomes unavailable, the HA system can switch over to one of the copies. Thus, RDM Server includes support for multi-slave architectures, allowing for complex replication models, increasing server uptime, data redundancy and data availability.

8. 1 RDM Server Replication

RDM Server provides native support for asynchronous, active-passive replication. The solution is simple and yet effective: one machine can act as the master server, receiving updates to the master database, while one or more machines can take the role of slave servers, providing read-only access to the slave databases.

Example Solutions Using RDM Server's Replication

- **Scalability** – Through replication, you can scale up your application, giving more users access to your data. With RDM Server you can have an unlimited number of slaves providing 'read' access while still devoting the master to receive only updates.
- **Failover** – Master and slave servers can have their roles reassigned in the event that either fails. If a master fails, a slave server can take on the master role; if a slave fails, secondary slaves can pick up the slack.
- **Security -** For reasons of security, it might be desirable to replicate certain databases on specific servers. With RDM Server's replication, it's possible to configure the system, such that specific databases on the master are replicated to targeted slaves. Also, there may arise situations where certain databases can expect a higher number of hits from users. By placing these databases on more powerful machines, the response times can be significantly improved.
- **Mobile Snapshots** – With asynchronous replication, the slave need not always be connected to the master. A snapshot can be taken of the database for those that require only 'read' access. This database can then be taken "on the road", and updated on a need to basis.

8.2 RDSADM Console

With RDM Server administration utility, rdsadm, you can configure all aspects of the slave master behavior remotely. This can be very useful in the event that you wish to reconfigure the master slave setup, start the initialization or simply terminate the entire replication all together.

8.3 Native API

Finally, RDM Server includes a replication control API. A brief description of each of the functions defined in this API is given below.

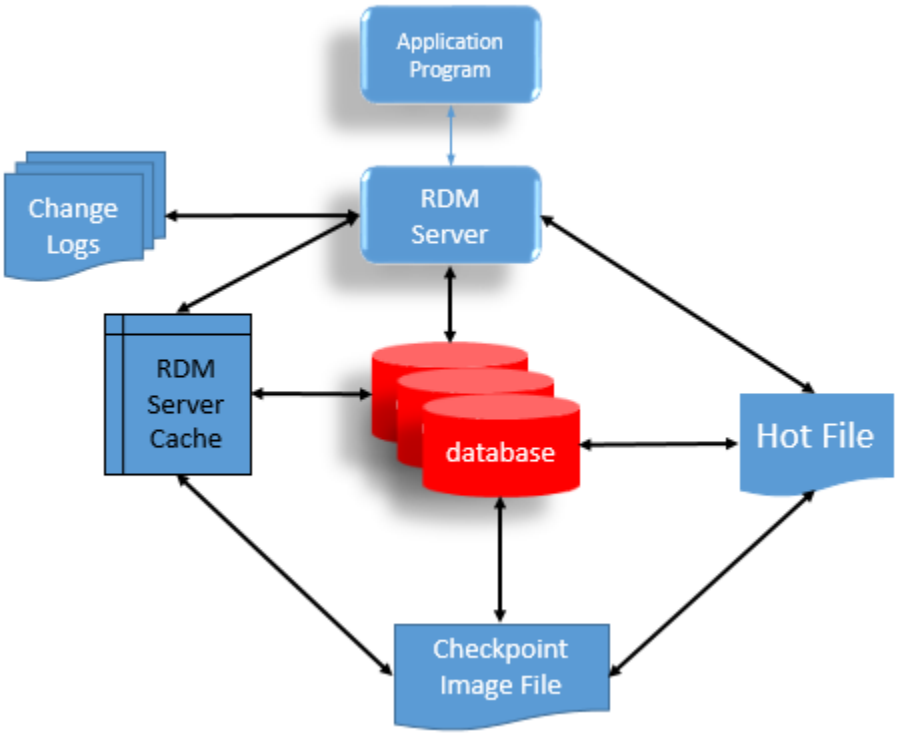
Replication API	Description
s_repBegin	Lets the slave server manually start the replication process
s_repCheckMode	Retrieves the operation mode of the slave server
s_repConnect	Connects the slave server to the specified master server
s_repConnected	Checks to see if the slave server is connected to the master server
s_repDbFiltered	Checks to see whether or not the specified database is one of the filtered (i.e. replicated) databases on the slave server
s_repDisconnect	Disconnects the slave server from the master server
s_repGetStatus	Retrieves the current status of the replication/initialization process on the slave server
s_repKill	Makes a request to kill the ongoing replication/initialization process
s_repRole	Retrieves the role the server plays

9. HIGH AVAILABILITY THROUGH HOT ONLINE BACKUP

High availability can also be implemented through Hot Online Backup. In today's dynamic business environment, where the expectation is that data should be accessible 24x7, Hot Online Backup offers the ability to back up one's database while still providing active read/write accessibility.

Hot online backup does not require any downtime of the system. RDM Server's implementation is fairly simple; anyone with administrator privilege can initiate backup either from RDSADMIN utility or programmatically using the Administration API. When your system is in hot backup mode, you can expect the following conditions to apply:

- Database Files are opened as shareable.
- All updates to the database are written to a "hot" file.
- No updates are made to the database files that will affect the logical consistency of the backup database.
- You can expect to experience some performance degradation.



It is important to understand that RDM Server does not provide a utility to write the database file contents to a backup media device such as a tape drive. Once hot backup mode begins, the actual backup can be performed manually or by using third-party backup software. You can even control the backup process through your own application using RDM Server's hot backup API.

When hot backup completes, the contents of the hot file are gradually migrated to the database on a need-to basis or you can choose, through an API call, to move all the changes from the hot file to the database immediately.

10. EXTENSIBILITY

One of the best ways developers can achieve excellent performance is by designing it into the database and application. RDM Server provides a database server that can be easily customized to meet the high performance demands of data intensive processes and to reduce the complexities of developing client/server or Internet/intranet applications.

- **Server extensions—**RDM Server is an open database server allowing user programmed extensions to be executed on the server under the control of RDM Server. Thus, programmers can customize and extend the server to meet their specific needs.
- Server Extensions execute on the database server with a single RPC call from heterogeneous clients. Server Extensions can be used to perform a variety of database-intensive operations needed by an application. A primary benefit of Server Extensions is the flexibility they provide. Developers can build applications that fit their technical requirements precisely.
- RDM Server includes the Server Extension Toolkit, a class library acting as a framework for quickly writing C++ client/server applications using the RDM Server Database Server Extension facility.
- **SQL User-defined functions (UDFs)—**UDFs are user programmed, server resident C functions that are called from RDM Server SQL when the UDFs are referenced in an expression in a SQL statement. They can be used to extend the scalar and/or aggregate functions for use by the application.
- **SQL User-defined stored procedures (UDPs)—**UDPs are user programmed, server resident C modules that are called from RDM Server SQL when a **CALL** statement that references the UDP is executed. UDPs can return one or more result sets (identical to those returned by **SELECT** statements) allowing non-SQL managed data to be returned as tables.

11. OPERATING SYSTEMS AND NETWORK SUPPORT

RDM Server supports a wide range of operating systems and network protocols. Supported operating systems include:

- Microsoft Windows
- Mac OS X
- Linux
- Solaris
- HPUX

Please visit Raima's web site or contact a sales representative for a complete list of supported operating system platforms.

RDM Server network connectivity services provide access to multiple applications over a heterogeneous network (for example, Windows clients connected to a UNIX server). RDM Server' supported network protocols include Named Pipes, TCP/IP, UNIX Domain Sockets, and Local (Shared Memory).

RDM Server' Multi-transport Network Communication Processor (MNCP) allows the server and client to be configured to use multiple network transports concurrently. Transports supported by RDM Server on a particular platform may be enabled or disabled on the server or client. This feature enables the server to maintain open sessions with clients using different network transports simultaneously.

Conversely, the same client application can open sessions to more than one server using different network transports. For example, a server can communicate with remote clients using the more expensive (in terms of resource requirements) TCP/IP transports, while using lightweight Local Transport to communicate with the client in the same memory space or, better yet, using the RDM Server Application Server Technology direct link, eliminating the need for the second network communication transport altogether.

- **UNIX Domain Sockets—**As a lightweight alternative to TCP/IP on UNIX implementations that support kernel threads, RDM Server offers a transport solution based on UNIX Domain sockets. Instead of using IP addresses and ports, the listening socket creates a temporary file and the connection socket specifies this file. The performance of this transport is higher than that of TCP/IP, but not as high as that of the Local (shared memory) Transport. RDM Server also has an option on UNIX platforms that allows the server to be spawned as a background process.
- **Windows Server Technology—**RDM Server takes advantage of windows extensive hardware support and support for multiprocessor hardware. The supported transports are TCP/IP, Named Pipes, and Local (shared memory).

12. PRODUCT CONFIGURATION AND DOCUMENTATION

RDM Server is available for a variety of system configurations and is priced based on the number of required configurations. RDM Server comes with comprehensive, updated on-line documentation, in HTML format, from our award-winning technical publications team.

- Installation and Administration Guide
- User's Guide
- Reference Guide
- SQL User's Guide
- SQL Reference Manual
- ADO.NET User's Guide
- JDBC Guide

This seven-piece documentation set provides information for both users and programmers, including detailed material on how to install, set up, administer, and develop RDM Server applications. A comprehensive suite of programming examples is provided along with example databases and feature descriptions.

13. ADDITIONAL RAIMA SERVICES

12. 1 Technical Support

Through Raima's Support programs, you can receive unlimited telephone support and consultation, access to 24x7 support site for support ticket management, free product updates, and access to Raima's FTP site.

12.2 Training

Raima provides product training throughout the world. Each course is divided into several sessions, covering a wide range of topics including basic database design principles, manipulating data with SQL, administering a database, and system-level functions. Please consult Raima's Web page at http://www.Raima.com/ or contact your Raima Sales Representative for on our training schedule.

12.3 Professional Development and Consulting Services

Raima Technology also provide technology development and consulting through it's professional services team. It serves an international customer base that includes Fortune 1000 companies and some of the world's leading technology developers. The team provides a wide range of services, and due to its close relationship with the core engineering team is particularly well suited for projects involving customization or application development with Raima's high performance database technology. For more information visit our Web site at http://www.Raima.com/.

Want to know more?

Please call us to discuss your database needs or email us at info@raima.com. You may also visit our website for the latest news, product downloads and documentation:

www.raima.com

Headquarter: 720 Third Avenue Suite 1100, Seattle, WA 98104, USA T: +1 206 748 5300
Europe: Stubbings House, Henley Road, Maidenhead, UK SL6 6QLT: +44 1628 826 800