White Paper

RDM Server 4.0

Where the application and the database server become one...

July 16, 2002

Haakon VII's gate 5B, N-0161 Oslo, Norway
Tel: +47 24 13 47 00 Fax: +47 24 13 47 01
hello@birdstep.com www.birdstep.com

Preface

This White Paper contains an overview of the RDM Server 4.0; 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 Birdstep 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.

Table of Contents

Product Overview 4
DBMS Technology 4
What's New in RDM Server 4.0 5
RDM Server Combined Database Model 7
RDM Server Architecture (Client/Server) 9
RDM Server C API 12
RDM Server SQL API 14
Zero Database Administration 18
Extensibility 20
Additional Programming Interfaces 21
Operating Systems and Network Support 22
Product Configuration 23
Additional Birdstep Services 24

Product Overview

The Birdstep RDM Server Database 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 GAPI, 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 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. The new Birdstep RDM Server 4.0 continues this legacy by adding critical requirements for many users and developers of modern database applications. RDM Server 4.0 is compliant with the latest industry database standards such as SQL and ODBC. In addition, RDM Server 4.0 adds unprecedented flexibility with the implementation of Dynamic DDL. 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.

DBMS Technology

The relational model's ease of use and flexibility has earned it popularity among many database developers and IS professionals. Relational databases are designed to load data, even bulk data, into the database quickly and efficiently. That's why SQL RDBMS products are popular for on-line transaction processing (OLTP) applications. RDM Server Database Server fully implements the relational database model.

However, the relational model was not designed for optimal performance in extracting data from the database, and its performance in this area can be too slow for many applications. Achieving acceptable performance using a RDBMS, especially in Internet/intranet applications, often requires more powerful hardware, specially optimized ODBC drivers, and elaborate SQL engineering. Because data access performance is a critical consideration in many applications, developers would ideally have access to a database model that is built to optimize data access performance, as well. Pointer-based Network model database structures provide direct and faster data access for demanding applications, such as embedded systems and Internet/intranet solutions. The Network model—which pre-dates the relational database model—is still used widely in powerful host back-end systems that require maximum performance.

The relational and network database models both have strengths and weaknesses. The best and fastest database applications will take advantage of the strengths that each model provides.

In addition to relational and network model database technology, developers sometimes also hear about hierarchical

and object-oriented databases. In fact, these types of databases are similar to network model databases in many ways, particularly in their access methods, which are based on navigating through related objects (record types) using direct pointers. The hierarchical model is a subset of the network model, while the object-oriented model's use of pointer-based navigation has led some to call it the reincarnation of the network database model.

While the terminology used to describe each model may be different, some generalization can be used to describe the basic components of a database. For example, related data (e.g. information on customers) is stored in "tables" or "files." Rows and records refer to individual instances of data within tables that may each include one or more "columns" or "fields." To join related information that is stored in separate tables, databases typically include either direct or keyed access methods, or both, as is the case with RDM Server. Direct access, which is the hallmark of the network database model, implies that the database provides a physical pointer to access, navigate, manipulate, store, and retrieve related records at the fastest possible speed. In the relational model, data is accessed, manipulated, stored, and retrieved using redundant keys in separate b-tree index files. Complex data designs generally require more tables (data files). Adding more tables in a relational database requires additional indexes in order to relate and access data.

What's New in RDM Server 4.0

Dynamic DDL

During initial development efforts there are often numerous changes to a database's structure in an effort to determine what an optimum design might be. With the inclusion of Dynamic Database Definition Language (DDL) this entire process can be done with one "Create Table" SQL command that would take seconds to execute.

DDL in a simplistic definition is the ability to modify the structure of a database while the database is actively in use. Dynamic DDL allows for activities such as adding a field to a record, creating a new index, or dropping an unused table but doing so automatically, without the need for external utilities or user initiated processes. The ability to quickly alter the composition of a database adds tremendous flexibility to a developer during the product development and upgrade cycle.

RDM Server Dynamic DDL Implementation

Dynamic DDL creation will be made available through both the Core API (for dynamic Core DDL creation) and through SQL DDL statement processing via the RDM Server SQL engine. However, the initial dynamic ddl implementation will support a predetermined set of dynamic features, but provide the foundation for extending the feature set in subsequent versions.

Dynamic DDL Features Supported in RDM Server 4.0

- Create a ***New*** database schema.
- Create a ***New*** data/key/blob file.
- Create a ***New*** *Record Type (Table)*
    - Add a BLOB field to a ***New*** record type
    - Add a data field to a ***New*** record type
    - Add a structure field to a ***New*** record type

- Create a ***New*** set type – Only in a ***New*** database
  - Add a member record type to set definition – Only in a ***New*** database
- Create a key on any index capable field of any record type
- Drop an existing key.
- Drop an existing database
- Drop a record type - only if the record is not part of ***Any*** set

Core DDL Modifications

The DDL functions implemented are listed in the following table:

TABLE OF CORE DDL CREATION API FUNCTIONS

Name	Description
ddlAbort	Abort database definition discarding all DDL changes
ddlCreateBlob	Add a BLOB field to a record type. (new record only until Phase 2)
ddlCreateField	Add a data field to a record type. (new record only until Phase 2)
ddlAddMember	Add a member record type to set definition. (Only in a new database)
ddlCreateStruct	Begin adding a structure field to a record type. (new record only 2)
ddlAlterDatabase	Alter an existing database definition.
ddlAlterRecord	Alter an existing record type definition.
ddlBegin	Begin schema modification transaction.
ddlCreateDatabase	Create a database schema.
ddlCreateFile	Create a data/key/blob file.
ddlCreateKey	Create a key.
ddlCreateRecord	Create a record type.
ddlCreateSet	Create a set type. (Only in a new database)
ddlDropDatabase	Drop a database.
ddlDropField	Drop a data field from a record type. (Phase 2)
ddlDropKey	Drop a key.
ddlDropRecord	Drop a record type. (only if the record is not part of a set)
ddlEnd	End database definition and commit the DDL modifications.
ddlEndStruct	End a structure field type in a record type. (new records only)
ddlGetField	Get field information.
ddlGetFileById	Get file information via the file id
ddlGetFileByName	Get file information via the file name
ddlGetRecord	Get record information.
ddlSetVirtualStatus	Set/Unset the virtual status of a field
ddlSchema	Compile and process a string containing a DDL specification.
ddlSchemaFile	Compile and process a file containing a complete DDL specification.

ODBC 3.51

With the release of RDM Server 4.0, support for ODBC version 3.51 is available. This feature will give RDM Server developers an impressive number of third-party tools and applications that can be used in conjunction with the powerful RDM Server engine.

RDM Server supports 100% of the Core ODBC API, and applicable Level 1 and Level 2 API. The following table contains a list of the ODBC 3.51 functions supported by Birdstep RDM Server 4.0:

TABLE OF SUPPORTED ODBC 3.51 API

SQLBindCol	**SQLFreeHandle**	**SQLGetInfo**
SQLConnect	**SQLGetDiagField**	**SQLSpecialColumns**
SQLDescribeCol	**SQLGetDiagRec**	**SQLStatistics**
SQLDisconnect	**SQLSetConnectAttr**	**SQLCopyDesc**
SQLExecDirect	**SQLDescribeParam**	**SQLFetchScroll**
SQLExecute	**SQLSetStmtAttr**	**SQLGetConnectAttr**
SQLFetch	**SQLColumns**	**SQLGetDescField**
SQLGetCursorName	**SQLGetData**	**SQLGetDescRec**
SQLNumResultCols	**SQLGetTypeInfo**	**SQLGetEnvAttr**
SQLPrepare	**SQLParamData**	**SQLGetStmtAttr**
SQLRowCount	**SQLPutData**	**SQLSetDescField**
SQLSetCursorName	**SQLTables**	**SQLSetDescRec**
SQLNumParams	**SQLCancel**	**SQLSetEnvAttr**
SQLBindParameter	**SQLColAttribute**	**SQLMoreResults**
SQLAllocHandle	**SQLNativeSql**	**SQLProcedures**
SQLCloseCursor	**SQLDriverConnect**	**SQLBulkOperations**
SQLEndTran	**SQLGetFunctions**	

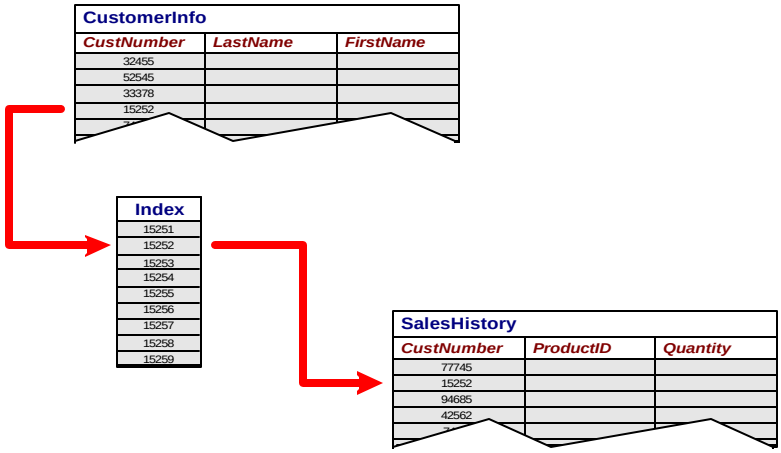
RDM Server Combined Database Model

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 pointers. These pointers create "sets" of record types, and the sets can be based on one-to-many, many-to-many, many-to-one, or recursive relationships between record types. Sets are physically implemented with linked lists of pointers to the row locations of the set members and owners. The pointer consists of four bytes and is contained within each record to support the relationship.

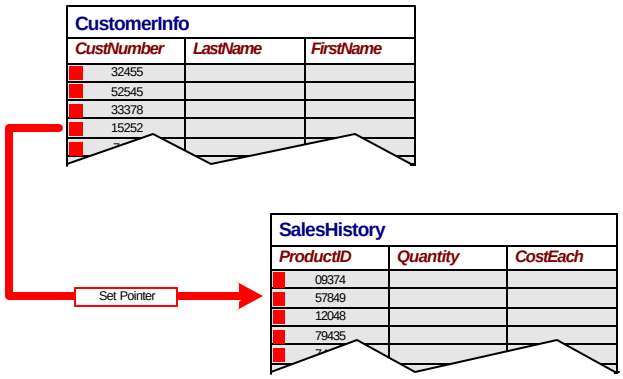
RDM Server Database Server allows you to use conventional relational foreign key/primary key joins, direct **CREATE JOIN**, or a combination of both. The ability to combine network and relational database technologies provides database application developers with the following advantages:

- Tables and rows are accessed directly
- Join processing performance is optimal

- Indexes are used only when needed
- Database size can be minimized
- More complex database designs are supported
- Referential integrity checking is faster



Relational Database Model



*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 member tables' rows are in a specific order, eliminating the need to index foreign keys to obtain faster data access.

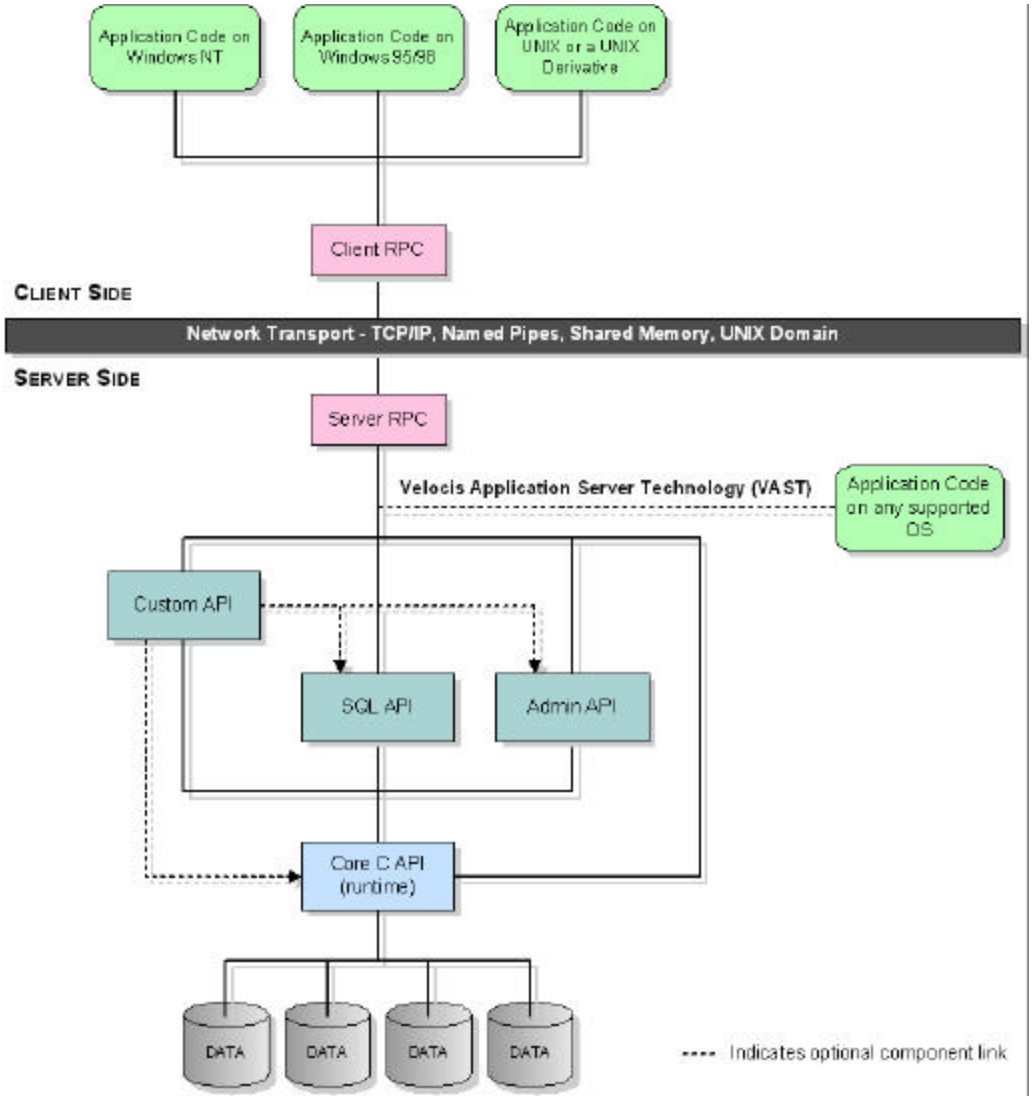
Foreign keys are declared in a table's definition, identifying tables with matching primary keys, thereby forming a logical relationships between tables. The **CREATE JOIN** statement specifies that the relationship formed by the primary key reference clause is to be maintained physically as a direct relationship. Through this statement, a direct relationship is formed,, consisting of the "owner"(that is, the primary key) table and the member (foreign key) tables.

RDM Server Architecture (Client/Server)

The RDM Server SQL system resides on a RDM Server 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. A 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, utilizing multiple thread features.

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.



RDM Server Architecture Diagram

Server Components

- ✓ **Database Management System**—Implements the database runtime. A true multi-threaded runtime built using asynchronous input/output and utilizing operating system threads for maximum throughput under heavy loads. The caching scheme and the transaction manager are optimized for transaction-oriented services.
- ✓ **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. Also includes the SQL language parser and optimizer.

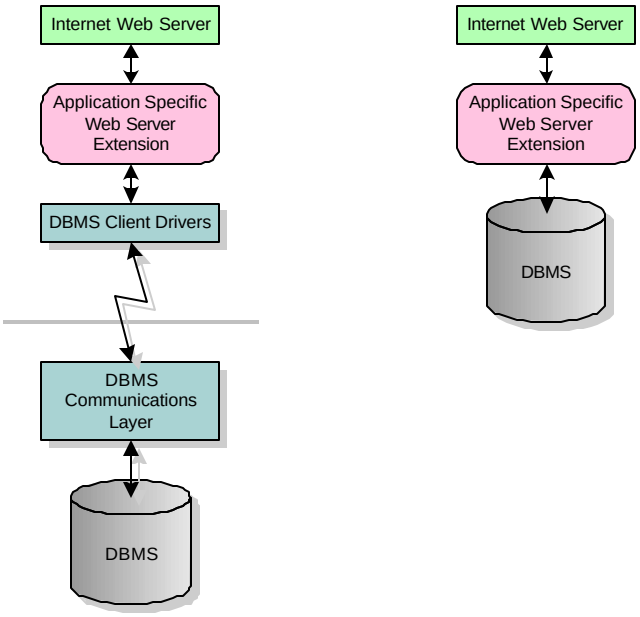
- ✓ **Server Extensions (Custom APIs)**—Implements high-level server functions for client support and server administration. User-defined modules specific to the client application(s) can be built as Server Extensions.
- ✓ **RPC Module—**(or RPC Server), in coordination with the MNCP, makes sure that 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.

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 NT server using Named Pipes.

RDM Server Application Server Technology

The RDM Server Application Server Technology provides a powerful new alternative for using RDM Server as an embedded database and as a 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.



Comparison of Client/Server vs. RDM Server Application Server

The RDM Server Application Server Technology 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 (for example, COM/DCOM on Windows NT, JAVA, etc.). 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 VAST application launches RDM Server (through a call to the new RDM Server

s_startup function), there is an option of having RDM Server also start its RPC/MNCP. This gives any standard RDM Server client program, such as third-party ODBC or OLE DB tools, access to the RDM Server database. Standalone (that is, non client-server) 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 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.

The left side of the 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 4.0 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 Netscape 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.

RDM Server C API

Every solid structure 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 GAPI 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 GAPI includes both a database definition language (DDL) and a rich set of database manipulation functions

Non-SQL Database Definition Language (DDL)

A RDM Server database design is specified in the 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 which 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" */
   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 */
```

Example of a RDM Server C DDL (continued)

```
   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;
   }
)
```

Non-SQL Database Manipulation

The Core Level or low-level C–API Database (the origins of which can be traced back to Birdstep'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 of the database server. 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 of some 150 functions is the foundation on which other interfaces are built—for example, the RDM Server SQL API was written using the record-level interface. A partial list of RDM Server C-API functions is given below.

TABLE OF RDM SERVER 4.0 CORE LEVEL C API FUNCTIONS

d_checkpoint	d_close	d_dbflush	d_errinfo	d_cotype	d_crread
d_crtype	d_csmread	d_crset	d_setmm	d_setmo	d_setmr
d_crwrite	d_csmwrite	d_csowrite	d_delete	d_csmset	d_csoget
d_csoset	d_crget	d_csoread	d_keyread	d_recread	d_connect
d_discon	d_disdel	d_fillnew	d_keydel	d_findlm	d_findnm
d_findpm	d_ismember	d_iclose	d_iopen	d_open	d_rdlockmodes
d_isowner	d_members	d_decode_dba	d_encode_dba	d_keyexist	d_keystore

d_makenew	d_recwrite	d_keylast	d_keynext	d_keyprev	d_pkeyfind
d_keywrstate	d_rdcurr	d_wrcurr	d_cmtype	d_pkeynext	d_pkeyprev
d_findco	d_findfm	d_recfrst	d_reclast	d_recnext	D_recprev
d_recset	d_curkey	d_keyrdstate	d_keyszstate	d_set_dberr	s_login
s_logout	D_csmget	d_setom	d_setoo	d_setor	d_setrm
d_setro	d_keydir	d_keyfind	D_keyfrst		

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

- ✓ **Custom Compare Function—**A RDM Server 4.0 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 G–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{
  d_ccread(VMA_DESC, vma_desc, hDb);                     /*enter fleet record*/
…
```

RDM Server SQL API

RDM Server' SQL API is a full implementation of most of the ANSI SQL 89 and some of 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 RDM Server server can be powerfully extended to meet developers' requirements.

SQL Database Definition Language (DDL)

RDM Server' 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            reference outlet(loc_id),
        prod_id             smallint            not null            reference 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:

- ✓ **Auto-increment Column Attribute (AutoNumber)—**supports the declaration of integer columns that are automatically assigned a unique number when a new row is inserted.
- ✓ **Binary Large Object (BLOB) Support—**RDM Server 4.0 conforms to the ODBC specification for columns of type **LONG VARCHAR** and **LONG VARBINARY**. The low-level support includes a new BLOB file type and a **BLOB_ID** field type. BLOB values can be any length. BLOB files can be configured with a user-specified page size. The low-level API allows for file-like operations on BLOB data. The application program can dynamically control the logging of BLOB data.
- ✓ **Custom Compare Function—**Enables assignment to a column/field of user-defined or system functions, which perform comparison operations when data is stored for that column/field.
- ✓ **File Extensions—** Most operating systems supported by RDM Server limit files to 2 (or 4) gigabytes. Given the increasing database size requirements in many applications (e.g., Internet, Data Warehouse etc.), this can be restrictive. RDM Server has the ability to define extension files for the base data, key, and BLOB files. When the base file has reached its maximum size, the additional database pages are written into the next extension file. The administrator using the RDM Server **ADMIN** utility declares extension files. Alternatively, new administration API functions have been defined that can add file extensions from within an application.
- ✓ **Very Large Database (VLDB) Support—**Very large databases are also supported by a change in the size of the **DB_ADDR** data type, implemented in previous releases. Formerly 4 bytes (1 byte to select the file and 3 bytes to select the slot), a **DB_ADDR** now contains 6 bytes (2 to select the file and 4 to select the slot), increasing the number of possible record slots in one file from 16,777,214 to 2,147,483,647.
- ✓ **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.
- ✓ **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 Windows95/NT). 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 now 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.

SQL Database Manipulation Language (DML)

Birdstep's commitment to standards ensures application portability and interoperability. RDM Server implements a SQL GAPI 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 4.0 SQL API FUNCTIONS

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

Some of the most notable features of RDM Server' DML are described in the following paragraphs.

- ✓ **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.
- ✓ **INSERT statement—**The insert values statement is used to insert a single row into a specified table. The **INSERT SELECT** statement is used to insert one or more rows from one table into another. The **INSERT FROM FILE** statement is used to perform a bulk load from data contained in an ASCII or Unicode text file.
- ✓ **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.
- ✓ **Scalar functions—**RDM Server contains a full complement of scalar functions, including math, string, and date calculation and manipulation capabilities.
- ✓ **Read-only (Static) Scrollable Cursors**—implemented in RDM Server 4.0 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.
- ✓ **Select Statement—**The **SELECT** statement capabilities include all of the ANSI capabilities except support for **UNION, JOINS, GROUP BY**; sub-query processing is supported. Moreover, RDM Server 4.0 includes an improved cost–based optimizer that utilizes all indexes as well as RDM Server' high performance predefined joins, supported by the **CREATE JOIN** statement.
  - ✓ **SQL Optimizer—**The RDM Server database engine maintains on–line counts of the total number of rows per table. This allows queries of the form

```
SELECT COUNT(*) FROM table
```

to return the values instantly without having to perform a table or index scan. In addition, the UPDATE STATS statement for all columns, now collects histograms in RDM Server, improving the accuracy of the optimizer's access cost estimates and simplifying the required analysis.

Along with the new optimizer statistics, the operation of the **UPDATE STATS** statement has been revamped to run much faster. Earlier versions required complete scans of every table and index in a database. For large databases, this could take a long time. In RDM Server 4.0, **UPDATE STATS** makes only a single scan of the relevant table and reads only a sampling of the rows in order to collect histogram counts for each column in

each table. The size of the histogram and the size of the sample are both configurable by editing the **RDM SERVER.INI** file options.

- ✓ **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.

- ✓ **Transaction processing—**The RDM Server database engine has been designed for high performance transaction-oriented applications. Among its state-of-the-art capabilities, the RDM Server engine optimizes transaction logging and recovery and performs asynchronous database I/O, allowing for greater concurrency on the server computer. As a faster option, asynchronous transactions are supported.

  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 begin, end, abort, mark, and roll back any transaction.

  Transaction processing is accomplished with a changelog file. Transaction commits are handled using a single write to the log, while transaction rollback is performed using undo operations. All changes made within a transaction are temporary until the transaction is ended. If the transaction ends normally, all changes are written to the database. If the transaction aborts, all changes are discarded.

  The database engine also groups output of contiguous database pages ("piggy-back" writes) and provides true row-level locking. At the lowest level, record and set locking are independent. Record locking is best used when an application needs to access, modify, or delete the data in a record. For example, obtaining a write lock to a set instance does not provide write access to the data in the record. Set locking supports navigation of a set, or connection or disconnection of a record. Each kind of lock only provides access to the related data.

  Using instance locks, the application can lock the current record, current owner, current member, current set, or a specified record by its database address. For table locks, the application can lock a record or set type.

- ✓ **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).

  RDM Server' triggers provide the full power of the SQL language, and they can also be associated with RDM Server' core API (C-API). Triggers (C or SQL) are programmed in C using user-defined functions and are called by a SQL statement.

- ✓ **User-defined functions (UDFs)—**UDFs are C functions that execute on the RDM Server server rather than being called from the client application. They are called from RDM Server SQL when the UDFs are used in an expression in a SQL statement. User defined functions have a variety of uses such as:

  - Translating coded values into easy-to-read strings
  - Performing special purpose computations
  - Adding new scalar or aggregate functions
  - Performing fast low-level database lookups
  - Implementing triggers

- ✓ **View Processing—**RDM Server SQL includes **CREATE VIEW** and **DROP VIEW** statements. Views can be used in **SELECT, INSERT, UPDATE, AND DELETE** statements. Updatable views with the **WITH CHECK OPTION** clause will be checked when the view is used in an **insert** or **update** statement.

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.

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.

TABLE OF RDM SERVER 4.0 ADMINISTRATION LEVEL FUNCTIONS

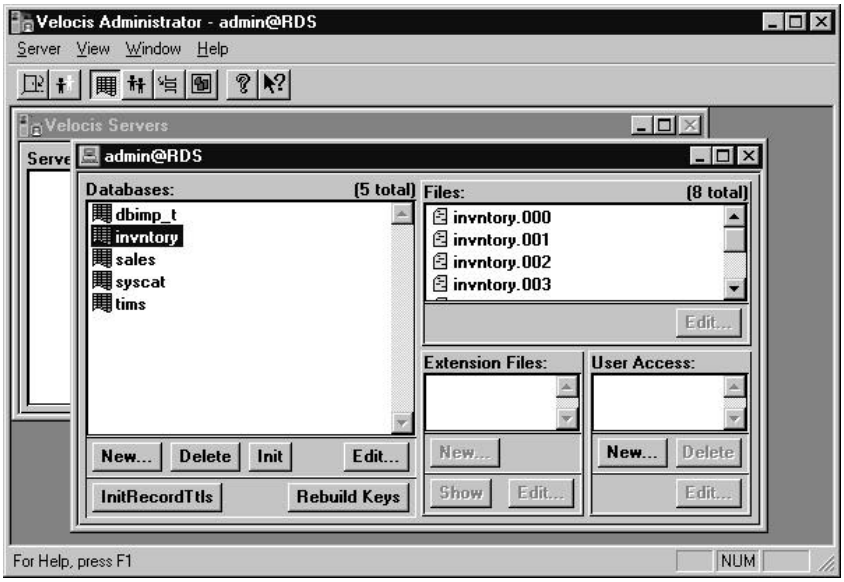
d_taskinfo	s_configGet	s_configModify	s_dbAdd	s_dbClone
s_dbDel	s_dbGet	s_dbInit	s_dbModify	s_dbUserAdd
s_dbUserDel	s_dbUserGet	s_devAdd	s_devDel	s_devGet
s_devModify	s_emAdd	s_emDel	s_emGet	s_emModify
s_familyAdd	s_familyDel	s_familyMemAdd	s_familyMemDel	s_familyModify
s_getMinFreeSpace	s_iniGet	s_iniGetPrivate	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

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, backup, and getting/setting parameters.

RDM Server 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



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 RDM Server 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 4.0 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.

Dbrepair

DBRepair is an asynchronous utility API provided in RDM Server 4.0, containing the following utilities:

- ✓ **Dbstat—**Database space consumption statistics and analysis utility

Client utilities	Client modules	Server modules	Public API
dbstat(.exe)	None	_edbs32.dll (Win32) _edbs.so (Unix)	

- ✓ **DBReplay** - Use of database families for controlling backups and roll-forward recovery have been eliminated in RDM Server 4.0 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 Extension—**The "database consistency check" program, called **DBCheck**, has been migrated from Raima Database Manager (RDM, Birdstep's original high performance database engine) to RDM Server. The **DBCheck** functionality is implemented in a Server Extension. The Server Extension is invoked by a new API

function named **s_dbcheck**. RDM Server 4.0 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 Extension—**The "key file rebuild" program, called **Keybuild**, has been migrated from RDM to RDM Server. As with **DBCheck**, **Keybuild** is a Server Extension, a new API function (named **s_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.
- ✓ **PRDBD—**RDM Server 4.0 includes a new database dictionary analysis utility called PRDB.

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. RDM Server includes the following features for extending its capabilities:

Server Extensions (Extension Modules)

RDM Server is an open database server allowing client- or server-side processing with Server Extensions. Programmers can customize and extend the server to meet their specific needs. Using Server Extensions, developers can host C or C++ application level code on the server or create custom APIs. RDM Server allows the extension of server capabilities to implement application-specific functions.

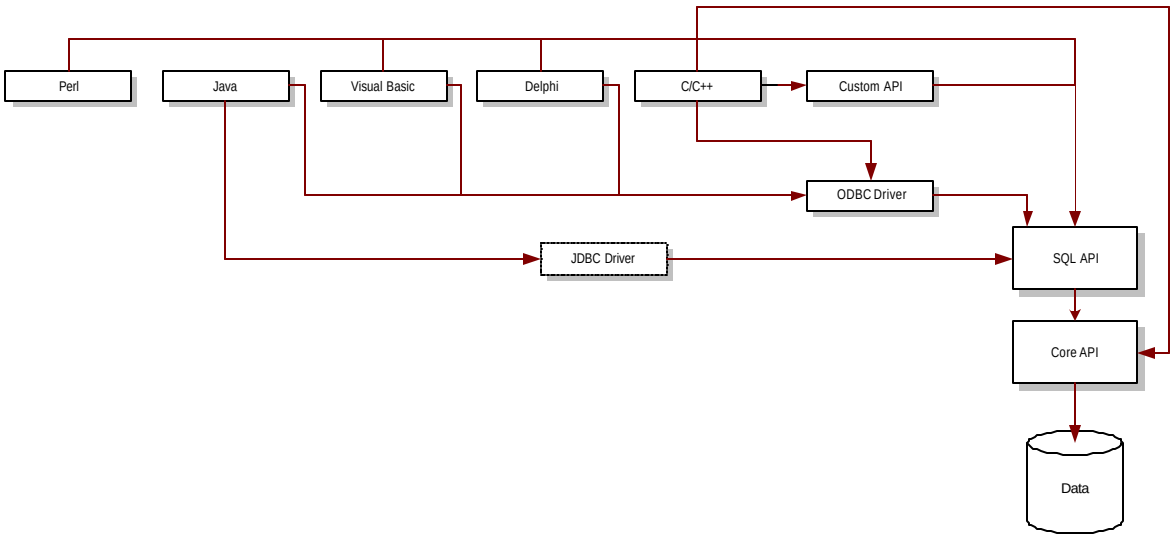
Server Extensions execute on the database server with a single RPC call from heterogeneous clients. Server Extensions can be used to perform a variety of database-intensive operations needed by an application. A primary benefit of Server Extensions is the flexibility they provide. Developers can build applications that fit their technical requirements precisely.

A configuration option, **AutoLoadModules**, loads listed Server Extensions automatically when the RDM Server server is started. These Server Extensions will not be unloaded until the server is shut down. The developer can specify as many Auto–Load Server Extensions as needed, or can even prevent the **'_sql'** Server Extension from loading on startup.

- ✓ **Server Extension Toolkit—**RDM Server 4.0 includes the RDM Server 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. It also supports the migration of RDM-based C++ applications to RDM Server Server Extensions.

Additional Programming Interfaces

Several custom interfaces that support many of today's popular development environments will be available for RDM Server 4.0 and can be obtained from the Birdstep Web page as a free downloads. Below is a diagram of the interfaces currently planned to be released to support RDM Server 4.0. Please check Birdstep's Web site at http://www.birdstep.com/ or contact your Birdstep Sales Representative for the availability of these interfaces.



RDM Server API's and Driver Support Diagram

Operating Systems and Network Support

RDM Server supports a wide range of operating systems and network protocols. Contact Birdstep for information on additional operating system support and availability.

Platform	Server	Client
Windows 2000	Yes	Yes
Windows NT 4.0 (Intel)	Yes	Yes
WinXP	Yes	Yes
Win98	Yes	Yes
WinME	Yes	Yes
HPUX 11.0	Yes	Yes
SUN Solaris 2.7, 2.8 (Sparc)	Yes	Yes
SUN Solaris 2.7, 2.8 (Intel)	Yes	Yes
AIX 4.3	Yes	Yes
Unixware 7.0.1	Yes	Yes
RedHat Linux 7.2	Yes	Yes
QNX Neutrino	Yes	Yes

Client/Server Communication

RDM Server network connectivity services provides access to multiple applications over a heterogeneous network (for example, Windows NT 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 RDM Server 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 NT—**RDM Server takes advantage of NT's extensive hardware support and support for

multiprocessor hardware. The supported transports are TCP/IP, Named Pipes, and Local (shared memory).

- ✓ **Windows 95/98—**As Windows 95/98 does not support creation of server-side Named Pipes, RDM Server only supports the following transports on that platform: TCP/IP, and Local (shared memory based). RDM Server also has an option to register the server as a service under Windows 95/98. This facilitates starting the server during the OS boot process.

Product Configuration

Documentation

RDM Server 4.0 comes with comprehensive, updated on-line documentation, in Adobe Acrobat format, from our award-winning technical publications team.

- RDM Server Database Server User's Guide
- RDM Server Database Server Reference Manual
- RDM Server Database Server SQL Language Guide
- RDM Server Database Server Installation and Administration Guide

This four-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. Printed documentation is also available please see Birdstep's Web page at http://www.birdstep.com/ or contact your Birdstep Sales Representative for ordering information at 1-206-748-5353.

Additional Birdstep Services

Training

Birdstep 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 Birdstep's Web page at http://www.birdstep.com/or contact your Birdstep Sales Representative for on our training schedule.

Professional Development and Consulting Services

Birdstep Technology has an experienced professional services and consulting division based in Seattle, Washington. It serves an international customer base that includes Fortune 1000 companies and some of the world's leading technology developers. We provide a wide range of services, and is particularly well-suited for projects involving customization or application development with Birdstep's high performance database technology. The professional services division provides professional C/C++ programming and Birdstep database design engineers who can tackle the toughest development jobs and satisfy the most demanding customers. For information on Birdstep Technology, call 1-206-748-5353 or visit our Web site at http://www.birdstep.com/.