Librería de acceso a la base de datos relacional de TJ-II:

Guía del usuario

Edilberto Sánchez, Ana B. Portas, Jesús Vega

Versión Inicial: Agosto de 2001

Última modificación: jueves, 03 de febrero de 2005

CLASIFICACIÓN DOE Y DESCRIPTORES

COMPUTER NETWORKS, COMPUTER PROGRAM DOCUMENTATION, DATA TRANSMISIÓN, DATA ACQUISITION, DATA ACQUISITION SYSTEMS, DATA BASE MANAGEMENT, INFORMATION RETRIEVAL, DATA COMPILATION

Librería de acceso a la base de datos relacional de TJ-II: Guía del usuario

Edilberto Sánchez, Ana B. Portas, Jesús Vega

52 pp., 10 tbls., 8 refs.

Resumen:

Para almacenar datos con significado físico de las descargas de TJ-II se ha desarrollado una base de datos relacional que complementa la base de datos brutos de TJ-II ya existente. Esta base de datos reside en un servidor con sistema operativo Windows 2000 Server y es administrada por el gestor SQL Server.

Se ha desarrollado una librería de funciones que permite el acceso remoto a estos datos desde programas de usuario ejecutándose en ordenadores conectados a las redes de área local de TJ-II, vía llamada a procedimientos remotos. En este documento se proporciona una descripción general de la base de datos relacional y su organización, así como una descripción detallada de las funciones incluidas en la librería y ejemplos de uso de estas funciones en programas escritos en lenguajes FORTRAN y C.

The TJ-II relational database access library: A user's guide

Edilberto Sánchez, Ana B. Portas, Jesús Vega

52 pp., 10 tbls., 8 refs.

Abstract:

A relational database has been developed to store data representing physical values from TJ-II discharges. This new database complements the existing TJ-II raw data database. This database resides in a host computer running Windows 2000 Server operating system and it is managed by SQL Server.

A function library has been developed that permits remote access to these data from user programs running in computers connected to TJ-II local area networks via remote procedure call. In this document a general description of the database and its organization are provided. Also given are a detailed description of the functions included in the library and examples of how to use these functions in computer programs written in the FORTRAN and C languages.

Tabla de contenidos

1. Audiencia y objetivos

Este documento está dirigido a personas que necesitan hacer uso de la librería desarrollada por el Grupo de Adquisición de Datos (GAD) del TJ-II para el acceso a datos de la base de datos relacional (RDB) desarrollada para este dispositivo. Se dirige tanto a usuarios que necesitan hacer uso de la librería para leer o integrar datos en dicha base de datos desde sus programas de usuario, como a aquellos que necesiten usar las funciones que esta librería implementa para desarrollar aplicaciones cliente o librerías de más alto nivel.

El objetivo de este documento es proporcionar la información necesaria para el uso de las funciones que la librería cliente de acceso a datos implementa. No se pretende una descripción exhaustiva, desde un punto de vista técnico, del *software* desarrollado para el acceso a datos RDB del TJ-II. Una descripción más detallada de este software, incluyendo la librería cliente, se hace en un documento aparte, dirigido a los programadores encargados de su mantenimiento.

Comentarios sugerencias o correcciones al respecto de este documento, al respecto de la librería y su uso, así como errores detectados en su funcionamiento pueden ser enviados a la dirección de correo electrónico edi.sanchez@ciemat.es.

2. Introducción

Con el propósito de almacenar datos con significado físico sobre las descargas del TJ-II se ha desarrollado una nueva base de datos [Sanchez02] que complementa a la base de datos brutos ya existente [Vega00]. En esta base de datos se almacenan datos relacionados con los experimentos llevados a cabo en el dispositivo TJ-II. Se pretende que esta base de datos contenga datos con relevancia física, independientes de los detalles locales de diseño de sistemas de diagnóstico particulares.

Se trata de una base de datos relacional que reside en un servidor con sistema operativo Windows 2000 Server y es gestionada por Microsoft® SQL Server. Los datos se organizan internamente en forma de tablas, cada tabla conteniendo diferentes campos. Entre diferentes campos de diferentes tablas se mantienen relaciones. Estas relaciones entre campos, junto con el motor relacional proporcionan una gran flexibilidad en la búsqueda de datos, permitiendo hacer búsquedas complejas en la base de datos usando el lenguaje SQL (Structured Query Language)[SQLRes].

Los datos que se integran en esta base de datos provienen de diferentes fuentes. Algunos datos generales serán introducidos en la base de datos por los encargados de la operación del dispositivo TJ-II, otros datos más específicos se obtendrán del análisis detallado de los datos experimentales capturados por diferentes sistemas de diagnóstico. Estos datos experimentales habrán de ser introducidos en la base de datos por los experimentales responsables de cada sistema de diagnóstico en concreto. Finalmente, se insertarán datos en esta base de datos directamente, y de forma automatizada, desde códigos de análisis o simulación.

Para el acceso a los datos de la base de datos desde múltiples plataformas, con independencia de *drivers* de bases de datos de terceras partes, como ODBC por ejemplo, el Grupo de Adquisición de Datos (GAD) del TJ-II ha desarrollado un sistema de acceso basado en el estándar *de facto* ONC RPC [RPC88, Bloomer92], ya utilizado con anterioridad en otros desarrollos por este grupo [Sanchez01].

Se trata de un *software* que sigue el modelo cliente/servidor y usa ONC RPC (Open Network Computing Remote Procedure Call) para las comunicaciones entre cliente y servidor, usando XDR (eXternal Data Representation) [XDR87] para la transferencia de datos entre ambos.

Un proceso servidor controla los accesos a la base de datos en el servidor Microsoft Windows que gestiona la base de datos y manipula la base de datos localmente usando una conexión local ODBC (Open Data Base Connectivity). Los programas de usuario o del sistema de adquisición de datos actúan como clientes de este programa servidor, enviando consultas al gestor de base de datos a través de este programa. Para tener un programa cliente de la base de datos del TJ-II, se necesita enlazar el programa con la librería de acceso a datos del TJ-II y llamar a las rutinas que esta librería, desarrollada por el GAD, proporciona.

En este documento se describe con detalle cómo enlazar un programa con esta librería, cómo usar las rutinas que proporciona, así como algunos detalles relativos a los permisos de acceso a los datos de la base de datos relacional del TJ-II.

3. Organización de la base de datos y política de permisos de acceso

La política general de permisos de acceso a los datos de la base de datos relacional del TJ-II es la siguiente: **todos los datos deben ser de acceso público para lectura, mientras que el acceso para escritura, inserción o modificación de datos, debe ser restringido.**

Los datos están organizados internamente en la base de datos en forma de tablas. Cada una de estas tablas contiene una serie de campos que se pueden identificar gráficamente con columnas de una tabla. Cada uno de los campos de una tabla puede contener datos de diferente tipo (numéricos enteros, numéricos de punto flotante, cadenas de caracteres, ...) mientras que los datos de un campo (columna) son siempre del mismo tipo. Diferentes campos de diferentes tablas pueden mantener relaciones que permitan hacer búsquedas en varias tablas en base a las relaciones definidas entre campos. Las búsquedas de datos se pueden hacer usando el lengueje SQL (Structured Query Language) que proporciona gran flexibilidad para realizar búsquedas complejas que son interpretadas y procesadas por el motor relacional, MS SQL Server en este caso.

Con el fin de independizar la estructura interna de la base de datos y sus tablas del software de acceso por un lado, y por otro de simplificar en la medida de lo posible las sentencias SQL que los usuarios necesiten enviar para buscar datos se ha optado por definir vistas. Estas vistas son, a todos los efectos, iguales que otras tablas de la base de datos, y están organizadas en campos que representan magnitudes físicas como aquellas. El usuario sólo tiene acceso a las vistas que el GAD le proporciona. En estas vistas se aglutinan datos de las tablas verdaderas de la base de datos. Esto permite que mientras que la estructura interna de la base de datos puede ser muy compleja, por razones de eficiencia, conteniendo muchas tablas con diferentes relaciones entre sus campos, de cara al usuario la base de datos sólo contenga unas pocas vistas, facilitándole la búsqueda de datos, proceso que en muchos casos se reducirá simplemente a seleccionar qué campos de una vista desea visualizar. La complejidad de las búsquedas en las diferentes tablas de la base de datos reside así, no en la sentencia SQL que el usuario debe enviar para buscar datos, sino en las sentencias SQL usadas para rellenar las vistas, trabajo que realiza en este caso el GAD del TJ-II.

Por otra parte, al independizar el software de acceso de la estructura interna, quedan enmascarados en gran medida, de cara al usuario, los detalles de organización interna, pudiéndose modificar la estructura interna de la base de datos si se requiere (por ampliaciones por ejemplo) sin que ello afecte al software de acceso a los datos que usa el cliente ni a sus programas.

Cada campo de cada tabla representa una magnitud física relevante para los experimentos de TJ-II y tiene un nombre que permite identificar a qué se refieren los datos de este campo. La inserción de los datos en esta base de datos es cometido de las diferentes personas responsables de cada uno de los sistemas de diagnóstico del plasma del TJ-II a través de los cuales se obtienen los datos experimentales con los que se rellenan los campos de las tablas. Como contrapartida a la responsabilidad sobre los datos insertados en la base de datos por cada usuario o grupo de usuarios, el sistema de adquisición de datos proporciona un mecanismo de identificación de usuarios que permita garantizar que los datos son modificados solamente por las personas autorizadas para ello.

Con el fin de controlar los accesos a los datos, cada campo de cada tabla tiene asignado un usuario y grupo propietarios que se corresponden con un usuario o grupo de usuarios experimentales que son los responsables de la obtención e inserción de los datos correspondientes a ese campo (magnitud física). La inserción y modificación de datos en cada campo sólo se permitirá a los usuarios propietarios del campo o a usuarios autorizados por ellos.

El GAD del TJ-II proporciona un software que permite el acceso a la TJ-II RDB, tanto para lectura de datos como para inserción o modificación de datos en la RDB, desde múltiples plataformas a través de la red usando la librería cliente de acceso a datos proporcionada por el GAD del TJ-II (ver 5: *Funciones de la librería RDB*). Para comodidad de los usuarios, este *software* no requiere que cada vez que se realiza un acceso a la base de datos se proporcione un nombre de usuario y una clave que permitan identificar al cliente. El control de los accesos a la RDB se basa en la identificación del usuario en la máquina cliente desde la que se intenta acceder a la base de datos. Es decir, se confía en el sistema de seguridad de la estación cliente desde la que se introducen datos, en la que el usuario cliente ha tenido que abrir una sesión de usuario. Este mecanismo de identificación permite que la integración de datos se pueda realizar de forma automatizada, desde códigos de análisis por ejemplo, siempre que con

anterioridad se haya solicitado permiso al GAD del TJ-II para integrar datos en la RDB desde una estación cliente.

Antes de poder introducir datos en esta base de datos, desde cualquier estación cliente, será preciso solicitar permiso de acceso al GAD del TJ-II, proporcionando información sobre el usuario y grupo experimental del TJ-II en nombre del cual se va a insertar datos en la RDB, qué datos se pretende introducir, desde qué estación (o estaciones) cliente de la red y con qué usuario (o usuarios) se harán las operaciones de adición/modificación de datos**. Se necesita una autorización para cada estación cliente y usuario que se pretenda utilizar para agregar o modificar un dato.**

En caso de no tener esta autorización, las funciones de lectura de información (**TJ2RDBGI**, **TJ2RDBS**) funcionarán correctamente, pues no es necesaria una autorización previa para usar estas funciones, mientras que las funciones cuyo propósito es insertar o modificar datos de la RDB (**TJ2RDBUP**, **TJ2RDBDEL**) fallarán, devolviendo un error que indica que el usuario no está autorizado a realizar la operación de escritura sobre la base de datos.

4. Librería de acceso a la base de datos relacional

Para el acceso a los datos de la base de datos relacional del TJ-II se ha desarrollado un software basado en el modelo cliente/servidor. Todos los accesos a los datos se gestionan a través de un proceso servidor que se ejecuta en el servidor Windows 2000 que alberga la base de datos. La iniciativa parte siempre del cliente, mientras que el servidor espera continuamente peticiones de conexión de clientes.

Para acceder a los datos se necesita un programa cliente que se comunique con este servidor de datos, enviándole consultas y recibiendo sus respuestas.

Para ello, el GAD del TJ-II ha desarrollado una librería que permite ser enlazada con un programa de usuario proporcionándole capacidades de acceso a los datos. Un programa enlazado con esta librería se comporta como un cliente del servidor de datos. La librería es proporcionada por el GAD, permite ser enlazada con programas desarrollados en C o en FORTRAN y gestiona todos los detalles de las comunicaciones con el servidor. El usuario sólo necesita enlazar su programa con esta librería y llamar a las funciones que implementa para acceder a los datos.

Esta librería está disponible para múltiples plataformas cliente.

En esta sección se describen los detalles acerca del uso de esta librería y sus funciones.

4.1. Plataformas cliente soportadas

La librería cliente se encuentra disponible para las siguientes plataformas (arquitectura / sistema operativo):

Tabla 1 Plataformas cliente soportadas

ALPHA AXP /UNIX	Se ha desarrollado y probado esta librería en True64 UNIX V 4.0
CRAY / UNICOS	Se ha probado el software en la versión 10.0 de UNICOS
ORIGIN / IRIX	Se ha probado el software en la versión 6.5.11 de IRIX, en la máquina jen50 del CIEMAT.

Power PC / Darwin	Se ha probado el software en la versión 10.2.5 de Mac OS X
Sparc / Solaris	Probada en Solaris V 4.0
Intel / Linux	Probada en Mandrake V 6.0 y RedHat 7.0
Intel / Windows 95,98,NT, 2000	Probada en MS Windows 95, Windows 98 y Windows NT 4.0, Windows 2000 SP3

Desde el punto de vista del usuario la plataforma en la que se instala la librería es indiferente, quedando enmascarados en la librería la mayoría de los detalles que diferencian unas plataformas de otras y hacen en muchos casos difícil la integración de diferentes plataformas. Las funciones que la librería proporciona para el usuario están diseñadas de forma que se puedan usar tanto desde códigos C, C^{++} como FORTRAN.

4.2. Instalación de la librería cliente

La instalación de la librería cliente de acceso a datos RDB del TJ-II se realiza de diferente forma en plataformas UNIX (o LINUX) y en plataformas Windows.

Para plataformas UNIX se han desarrollado un conjunto de *makefiles* que permiten, de forma sencilla y casi automatizada, la compilación del código fuente de la librería en las diferentes plataformas cliente de tipo UNIX ya soportadas (ALPHA AXP/UNIX, CRAY/UNICOS, ORIGIN/IRIS, PowerPC/ Darwin, Sparc/Solaris e Intel/LINUX). Tras la compilación del código, la instalación se completa con la copia de las librería en un directorio de acceso público del sistema.

En general esta instalación se llevará a cabo por el personal del GAD del TJ-II, quien realizará la compilación si se trata de una plataforma cliente nueva, o proporcionará la librería compilada para la plataforma de destino si se trata de una de las plataformas ya soportadas.

Para el caso de sistemas operativos MS Windows 9x/NT/2000 se ha desarrollado un programa instalador que carga en el sistema las librerías necesarias. Este instalador carga tanto las librerías de ONC RPC (**pwrpc32.lib y pwrpc32.dll**) proporcionadas por Netbula (**http://www.netbula.com**) y que son usadas en este desarrollo, como las diferentes versiones para Windows de la librería cliente de acceso a datos RDB del TJ-II

(**tj2RdbC.dll, tj2RdbC.lib, tj2RdbCMT.dll, tj2RdbCMT.lib**). Este instalador ha sido desarrollado con **InstallShield** y se podrá obtener del GAD del TJ-II o en la URL **http://tjiiws.ciemat.es**

Este proceso de instalación lo puede llevar a cabo el propio usuario en su ordenador personal con sistema operativo Windows. Tras la instalación del *software* será preciso pedir autorización para inserción de datos en la base de datos si se desea llevar a cabo ese tipo de operaciones desde un equipo personal (ver *3: Organización de la base de datos y política de permisos de acceso)*

4.3. Configuración de la librería

La librería cliente de acceso a datos RDB del TJ-II requiere tan sólo del ajuste de tres parámetros de configuración. Se trata de variables de entorno que permiten modificar algunos aspectos de funcionamiento de la librería cliente de acceso a datos.

Tabla 2 Variables de entorno para configuración de la librería

TJ2RDB_SERVER	Se trata de una variable de entorno que permite establecer el servidor en el que la librería cliente va a intentar ubicar el servidor RPC de la base de datos cuando se llame a una función de la librería. En general la librería, tal como es distribuida por el GAD del TJ-II, "conoce" el servidor con el que debe conectar para buscar datos RDB; sin embargo, si se produce un traslado de la base de datos RDB del TJ-II a otro servidor, cambiando el valor de esta variable de entorno, poniendo el nombre del nuevo servidor, no se requerirá recompilar la librería para seguir teniendo acceso a los datos.
TJ2RDB_TIMEOUT1	Con esta variable de entorno se controla el tiempo (en segundos) que se espera respuesta del servidor tras una consulta, usando las funciones de la librería (todas excepto TJ2RDBGI) . Por defecto este valor es 25 segundos. En función de la calidad de la conectividad entre el

	cliente y el servidor de datos pueden ser recomendables diferentes valores en los tiempos de espera de respuesta por parte del servidor.
TJ2RDB_TIMEOUT2	Con esta variable de entorno se controla el tiempo (en segundos) que se espera respuesta del servidor tras una consulta, usando la función TJ2RDBGI de la librería. Por defecto este valor es 25 segundos. En función de la calidad de la conectividad entre el cliente y el servidor de datos pueden ser recomendables diferentes valores en los tiempos de espera de respuesta por parte del servidor.
TJ2RDB_COL_SEPARATOR	Con esta variable de entorno se controla el carácter ASCII que se usa para separar campos (columnas) en las llamadas a las funciones TJ2RDBGI y TJ2RDBS. Por defecto este valor es el carácter de código ASCII 9 (tabulador). Ver secciones 5.1. La función TJ2RDBGI y 5.2. La función TJ2RDBS.
TJ2RDB_LINE_SEPARATOR	Con esta variable de entorno se controla el carácter ASCII que se usa para separar líneas (columnas) en las llamadas a las funciones TJ2RDBGI y TJ2RDBS. Por defecto este valor es el carácter de código ASCII 10 (nueva línea). Ver secciones 5.1. La función TJ2RDBGI y 5.2. La función TJ2RDBS.

Para modificar el valor de estas variables de entorno han de seguirse los siguientes pasos, dependiendo de la plataforma de la que se trate.

En sistemas de tipo UNIX o **LINUX**, se procede como es habitual en estos sistemas.

Desde la línea de comandos, ejecutar

```
PROMPT> export TJ2RDB_SERVER="nombreHost.ciemat.es"
```

para shells sh o ksh, o bien

```
PROMPT> setenv TJ2RDB_SERVER "nombreHost.ciemat.es"
```

para shells de tipo C (csh o tcsh)

Se puede hacer esto mismo, en lugar de hacerlo desde la línea de comandos, desde un *srcipt* de entrada, como **.profile**, .**csh_login**, **bash_profile**, ... De este modo no es necesario establecer el valor de la variable de entorno manualmente cada vez que se use la librería.

En plataformas Windows NT o Windows 2000 se pueden establecer valores para variables de entorno a través del panel de control de Sitema, sección Entorno. Cabe la opción de establecer el valor de las variables para todo el sistema (todas las cuentas de usuario) si se tienen permisos de administrador, o bien para la cuenta de usuario en uso.

En plataformas Windows 95 o Windows 98.

Definir las variables de entorno en el fichero **autoexec.bat**

```
PROMPT> SET TJ2RDB_SERVER=das08.ciemat.es
```

4.4. Como enlazar un programa con la librería RDB

La librería se encuentra disponible en diferentes versiones dependiendo de la plataforma. La forma de enlazar con la librería puede ser ligeramente diferente dependiendo de la plataforma, del compilador que se use y de la instalación de la librería propiamente dicha (el directorio donde se encuentre instalada).

4.4.1.1. Plataformas ALPHA AXP /UNIX (Servidor 8400 de Adquisición de datos)

La librería se encuentra en dos versiones, una versión estática, como librería de objetos (**libRdbC.a**) y versión de librería compartida o de enlace dinámico (**libRdbC.so**).

En el caso del servidor central del sistema de adquisición de datos del TJ-II (ALPHA 8400) estas librerías se encuentran instaladas en los directorios **/usr/local/lib** y **/usr/local/shlib** respectivamente.

Para enlazar con estas librerías usar la sentencia

```
PROMPT> cc –o myProg myProg.c –lRdbC.so
```

para enlazar con la librería de objetos compartidos (dinámica)

```
PROMPT> cc –o myProg myProg.c –lRdbC.a
```

para enlazar con la librería estática

En estos ejemplos se asume que *myProg.c* es un código fuente escrito en C que llama a funciones de la librería **libRdbC**. Igualmente serviría el ejemplo si se tratara de un código escrito en FORTRAN usando el compilador FORTRAN correspondiente, en lugar de **cc**

```
PROMPT> f77 –o myProg myProg.f –lRdbC.so
```

para enlazar con la librería de objetos compartidos (dinámica)

```
PROMPT> f77 –o myProg myProg.f –lRdbC.a
```

para enlazar con la librería estática

En caso de que los directorios **/usr/local/lib** y **/usr/local/shlib** no se encuentren incluidos en la lista de caminos de búsqueda de librerías del *linker* (**ld**) se puede compilar con:

```
PROMPT> cc –o myProg myProg.c –L/usr/local/shlib –lRdbC.so
```

para enlazar con la librería dinámica

```
PROMPT> cc –o myProg myProg.c –L/usr/local/shlib –lRdbC.a
```

para enlazar con la librería estática

4.4.1.2. Otras plataformas UNIX, LINUX y darwin ,

Se compilará el programa de usuario de la misma forma que en plataformas ALPHA, usando los directorios donde se encuentren instaladas las librerías, que dependerán de cada máquina, al usar la opción –L del *linker*.

4.4.1.3. Plataformas Windows

Para plataformas Windows se cuenta con varias librerías diferentes. Todas ellas se pueden instalar usando el programa instalador distribuido por el GAD del TJ-II.

Tabla 3 Librerías para plataformas Windows

tj2RdbC.dll	Librería de enlace dinámico (DLL) para aplicaciones con un solo hilo de ejecución. Las funciones siguen el convenio de llamada estándar
tj2RdbCMT.dll	Librería de enlace dinámico (DLL) con soporte para aplicaciones con múltiples hilos de ejecución. Las funciones siguen el convenio de llamada estándar

Todas las versiones de la librería RDB necesitan para su correcto funcionamiento que la librería ONC RPC (ficheros **pwrpc32.dll y pwrpc.lib**) esté instalada en el sistema. Esta librería se instala automáticamente cuando se usa el instalador de software proporcionado por el GAD del TJ-II (ver *4.2: Instalación* de la librería cliente).

La librería **tj2RdbCMT.dll** es la librería básica, con soporte para aplicaciones con múltiples hilos de ejecución, que se usará de forma general para el acceso a datos desde programas escritos en C. Usando esta librería, cada vez que se realiza un conexión con el servidor, ejecutando una de las funciones distribuidas en la librería, se genera un nuevo hilo de ejecución que concluye cuando finaliza la ejecución de la función.

La librería **tj2RdbC.dll** es una versión de la librería RDB sin soporte para múltiples hilos de ejecución. Es similar a la librería **tj2RdbCMT.dll**, pero **no se debe usar en aplicaciones que requieran soporte para múltiples hilos de ejecución**. Se usa esta librería, por ejemplo, en aplicaciones desarrolladas en Visual Basic 6, que no tiene soporte para múltiples hilos de ejecución.

Llamada a las funciones de la librería desde programas escritos en C/C++

Desde programas C en plataformas Windows no hay problemas para usar las funciones de la librería [1].

Estas funciones están declaradas como funciones C con el convenio de llamada estándar (directiva **__stdcall**). Estos son los únicos detalles que pueden ser necesarios para compilar un programa C o C++. Se puede usar tanto la librería con soporte para

[1] Se ha probado el uso de esta librería en programas C en plataformas Windows usando el entorno de programación Microsoft Visual Studio 6.

múltiples hilos (**tj2RdbCMT.dll**) como la librería sin soporte para múltiples hilos (**tj2RdbC.dll**)

Llamada a las funciones de la librería desde programas Visual Basic

Para mayor compatibilidad y comodidad de uso, las funciones exportadas por la librería se han definido con el convenio de llamada estándar. Este detalle es necesario para poder llamar desde Visual Basic a funciones escritas en C.

Visual Basic no soporta el uso de las librerías con soporte para múltiples hilos de ejecución, por lo que para aplicaciones Visual Basic será preciso usar la versión de la librería sin soporte para múltiples hilos (**tj2RdbC.dll**).

Para usar las funciones de la librería en Visual Basic, será preciso declararlas como funciones externas importadas de una librería, de la siguiente forma:

```
Public Declare Function TJ2RDBGI Lib " tj2RdbC.dll " (sizeData As Long, _
        nTabs As Long, ByVal Data As String, errC As Long) As Long

Public Declare Function TJ2RDBS Lib " tj2RdbC.dll " (ByVal selectCmd _
        As String, sizeSel As Long, sizeCNames As Long, sizeTyp As Long, _
        sizeData As Long, ByVal cNames As String, ByVal types As String, _
         ByVal data As String, nRows As Long, nCols As Long, errC As Long)
        As Long

Public Declare Function TJ2RDBUP Lib " tj2RdbC.dll " (ByVal keyNamStr As
        String, ByVal keyNamDStr As String, ByVal datoStr As String, ByVal _
        datoDStr As String,  errC As Long) As Long

Public Declare Function TJ2RDBDEL Lib " tj2RdbC.dll " (ByVal keyNamStr As
        String, ByVal keyNamDStr As String, ByVal datoStr As String, ByVal _
        datoDStr As String, errC As Long) As Long

Public Declare Function TJ2RDBGERR Lib " tj2RdbC.dll " (errCo As Long, _
      ByVal ErrTxt As String, size As Long) As Long
```

Llamada a las funciones de la librería desde programas escritos en FORTRAN

Para llamar a funciones escritas en C desde programas FORTRAN, y debido a las diferencias entre C y FORTRAN en cuanto los convenios de llamada a funciones y paso de argumentos a estas, es preciso proceder con cautela.

En el entorno de programación Visual Fortran, para poder llamar a las funciones de la librería es preciso compilar el programa con la opción del compilador ***/iface:nomixed_str_len_arg*** [Etzel99]. Esta opción permite que la longitud de las cadenas de caracteres que se pasan como argumentos a funciones no se pasen justo después de pasar la referencia a la cadena, como se haría por defecto en el caso de este compilador, sino al final de toda la lista de argumentos de la función, como se hace en multitud de compiladores FORTRAN, entre ellos **f77** instalado en el servidor 8400 del sistema de adquisición de datos del TJ-II.

El resto de particularidades a tener en cuenta cuando se llama a funciones C en programas FORTRAN quedan resueltas por el hecho de que las funciones se han declarado con el convenio de llamada estándar (**__stdcall**)[2]. Además, para facilitar la compatibilidad con lenguejes C y FORTRAN en diferentes plataformas, todas las funciones de la librería de uso general se han declarado con nombres en mayúsculas y minúsculas.

[2] Algunas funciones de la librería de uso interno a ala librería se han declarado con convenio de llamada C.

5. Funciones de la librería RDB

La librería cliente de acceso a datos implementa un juego de funciones básicas (de bajo nivel) que permiten acceder a los datos de la base de datos relacional del TJ-II, tanto para lectura como para inserción de datos, desde programas de usuario escritos en C, C++ o FORTRAN. Usando estas funciones se podrán implementar otras más sofisticadas a un nivel superior, ya sea por el propio usuario o por el GAD del TJ-II.

Para compatibilidad con lenguajes C, C++ y FORTRAN, todos los argumentos se pasarán a las funciones por referencia.

Por otra parte, también por compatibilidad con C, C++ y FORTRAN, y debido al diferente tratamiento que estos lenguajes de programación hacen de las cadenas de caracteres, se asumirá que las cadenas que se pasen como argumentos a las funciones no contienen necesariamente un carácter de terminación de cadena, y su dimensión habrá de ser pasada a la función en un argumento extra, o asumida por defecto, salvo que se indique lo contrario (ver 5.3*: La función TJ2RDBDEL* y 5.4: *La función TJ2RDBUP*). **No se hace uso de los modos implícitos que los compiladores FORTRAN usan para pasar las longitudes de cadenas de caracteres a funciones, sino que en el caso de que esas longitudes sean necesarias, se prevee el uso de un argumento extra de tipo entero para enviar esta información a la función**.

La mayoría de las funciones de esta librería comparten una estructura común, retornan un valor de tipo entero que sirve para indicar si la ejecución de la función se llevó a cabo satisfactoriamente o no. Todas tienen un argumento de salida de tipo entero en el que se retorna un código de error indicativo de lo sucedido en caso de que la ejecución de la función falle por alguna razón.

La librería RDB, además de algunas otras de uso interno a la librería para tareas intermedias, contiene las siguientes funciones para el usuario:

Tabla 4 Funciones de la librería

Nombre de función	Cometido de la función
TJ2RDBGI	Obtener información general del contenido de la base de datos accesible al usuario
TJ2RDBS	Realizar consultas (SELECT) sobre la base de datos
TJ2RDBUP	Realizar inserciones o modificaciones de datos en la base de datos relacional
TJ2RDBDEL	Eliminar datos de la base de datos relacional
TJ2RDBGERR	Obtener el mensaje de error asociado a un código de error de la librería RDB.
TJ2RDBPERR	Imprimir en pantalla el mensaje de error asociado a un código de error de la librería RDB. No se podrá usar si el programa desde el que es llamada no tiene una terminal de control asociada.
TJ2RDBGETSZ	Obtener los tamaños que se usan para almacenar nombres y valores de columnas de la base de datos en llamadas a funciones TJ2RDBDEL y TJ2RDBUP.
TJ2RDBSETSZ	Establecer los tamaños que se usan para almacenar nombres y valores de columnas de la base de datos en llamadas a funciones TJ2RDBDEL y TJ2RDBUP.
TJ2RDBGETSEP	Obtener los separadores de filas y columnas que se están usando en consultas a la base de datos con funciones TJ2RDBGI Y TJ2RDBS.
TJ2RDBSETSEP	Establecer los separadores de filas y columnas en consultas a la base de datos con funciones TJ2RDBGI Y TJ2RDBS.

Para facilitar la compatibilidad en diferentes plataformas con programas escritos en C, C++ y FORTRAN sin tener que usar opciones de compilación especiales para modificar el modo de resolución de símbolos por defecto, todas las funciones se han declarado en cuatro formas (MAYUSCULAS, MAYUSCULAS_, minusculas y

minusculas_). Así, por ejemplo, para la función TJ2RDBGI, se han definido cuatro funciones:

TJ2RDBGI, **TJ2RDBGI_, tj2rdbgi** y **tj2rdbgi_**

Todas estas funciones hacen exactamente lo mismo, de hecho tres de ellas llaman a la cuarta, que es la que finalmente se ejecuta siempre. A lo largo de este texto, todas las referencias a una función se harán con su nombre en mayúsculas.

5.1. La función TJ2RDBGI

El propósito de esta función es obtener información de las vistas de la base de datos accesibles al usuario. El usuario no accede a las tablas de la base de datos propiamente dichas, sino a determinadas vistas que aglutinan datos de diferentes tablas en las que se organiza internamente la información de la base de datos (ver 3*:Organización de la base de datos y política de permisos de acceso*) Esto permite independizar el código de acceso a la base de datos de la estructura interna de aquella y a la vez simplificar las consultas SQL que el usuario necesita hacer para buscar información, no necesitando conocer a priori la estructura interna de tablas, estructura que puede ser modificada.

Esta función está diseñada para facilitar el desarrollo de programas interactivos de consulta de la base de datos. Usando esta función, estos programas se pueden desarrollar de forma general e independiente de la estructura de la base de datos. Con esta función se obtendría información de las vistas de la base de datos disponibles para el usuario así como de su contenido, tipo de datos y significado. A partir de esta información básica el programa puede presentar al usuario, en un interfaz amigable, opciones para la selección de los campos de la base de datos que desea visualizar, así como elegir entre diferentes criterios de selección de los datos.

5.1.1. Descripción detallada de TJ2RDBGI

```
int TJ2RDBGI ( int   *sizeData,
               int   *nTabs,
               char *data,
               int   *errC)
```

Argumentos de entrada:

SizeData es una variable de tipo entero, declarada en el programa principal en la que se indica a la función el tamaño (en octetos) reservado

en la memoria del programa que llama a esta función para recibir los datos resultado de la consulta.

Argumentos de salida:

nTabs es una variable de tipo entero, declarada en el programa principal donde se recibirá, al retorno de la función, el número de vistas de la base de datos que son accesibles al usuario.

data es una variable de tipo cadena de caracteres donde se recibe, al retorno de la función, el resultado de la consulta, conteniendo los nombres de las vistas, los nombres de los campos que contiene cada vista, el tipo de dato de cada campo y un comentario sobre el significado del campo.

errC variable de tipo entero en la que se recibirá, al retorno de la función, un código de error indicativo de los sucedido, caso de que se produzca un error en la ejecución de la función.

La cadena de caracteres (**data**) ha de ser declarada en el programa principal con una dimensión suficientemente grande como para almacenar el resultado esperado. Su dimensión ha de ser mayor o igual que el valor introducido en la variable **sizeData**. Si la dimensión fuera menor que la declarada en aquella variable, puede darse el caso de que la función **TJ2RDBGI** escriba datos de retorno en posiciones de memoria fuera del espacio reservado para la variable **data**, lo que puede ocasionar una invasión de memoria y resultados inesperados en la ejecución posterior del programa.

Al retorno de la función, y supuesto que el proceso se ha llevado a cabo satisfactoriamente, la variable data contendrá una cadena de caracteres con la siguiente estructura:

```
TabName1, TAB, nFields1, RETURN
NameField1, TAB, TypeField1, TAB, ComField1, RETURN
...
NameFieldN, TAB, TypeFieldN, TAB, ComFieldN, RETURN
TabName2, TAB, nFields2, RETURN
NameField1, TAB, TypeField1, TAB, ComField1, RETURN
...
NameFieldN, TAB, TypeFieldN, TAB, ComFieldN, RETURN
```

```
TabNameM, TAB, nFieldsM, RETURN
NameField1, TAB, TypeField1, TAB, ComField1, RETURN
...
NameFieldN, TAB, TypeFieldN, TAB, ComFieldN, RETURN
```

Esta cadena de caracteres acabará en el carácter NULL y el significado de los elementos que la componen es el siguiente:

TabName1, TabName2, ..., son cadenas de caracteres conteniendo el nombre de las tablas (vistas) de la base de datos accesibles al usuario.

Para cada vista se recibe una línea conteniendo el nombre de la vista, el número de campos que contiene y una cadena de caracteres con comentarios generales acerca del contenido de la vista. Por defecto, estos datos vendrán separados por el carácter de tabulación, TAB (código ASCII 9) y al final vendrá un carácter de nueva línea (código ASCII 13).

A continuación de la línea correspondiente a cada vista, vendrán tantas líneas de datos como campos contiene la vista. Cada una de estas líneas contiene el nombre del campo, un carácter indicativo del tipo de datos que contiene ese campo, por ejemplo I para datos de tipo entero, F para datos de punto flotante, C para caracteres, ... Finalmente, se recibe una cadena de caracteres con comentarios sobre el significado del campo en cuestión. Por defecto, los datos que contiene la línea de un campo están separados por el carácter de tabulación (TAB) y la línea termina con un carácter de nueva línea.

Puesto que en la estructura actual de la base de datos algunos campos pueden contener los caracteres de tabulación y/o nueva línea, se ha introducido la posibilidad de modificar los caracteres usados como separadores de campos y líneas respectivamente. Se puede modificar los caracteres usados para separar campos y líneas a través de las variables de entorno TJ2RDB_COL_SEPARATOR y TJ2RDB_LINE_SEPARATOR (ver sección 4.3. Configuración de la librería) , o haciendo uso desde el programa principal de la función TJ2RDBSETSEP (ver sección 5.10. La función TJ2RDBSETSEP). Con la función TJ2RDBGETSEP se pueden obtener en cualquier momento los caracteres que se están usando como separadores de campos y líneas en llamadas a las funciones TJ2RDBGI y TJ2RDBS. **Se recomienda, para evitar confusión entre los caracteres separadores de campos y líneas y el contenido de**

campos, usar como separadores caracteres con código decimal ASCII por encima del 128.

La función devuelve un dato de tipo entero indicando si se ejecutó con éxito (0) o si ocurrió algún problema en la ejecución de la función (distinto de cero). En caso de error hay que consultar el valor de la variable **errC**, que será un código de error indicativo de lo sucedido. Se puede obtener una descripción del error, a través del código de error usando las funciones **TJ2RDBPERR** y **TJ2RDBGERR** (ver 5.5: *La función TJ2RDBPERR* y 5.6*: La función TJ2RDBGERR*).

5.1.2. Ejemplos de uso de TJ2RDBGI

5.1.2.1. Código FORTRAN que usa la función TJ2RDBGI

```
PROGRAM TESTRDBGI
implicit none
integer             nData
parameter           ( nData = 32767 )
character*1         data(1:nData)
integer             nTabs, errC, err1, err2, sizeData, i
integer             TJ2RDBPERR, TJ2RDBGI
external            TJ2RDBPERR, TJ2RDBGI

sizeData = nData
write(*,*)' Programa de prueba de GetInfo'
write(*,*)'     Maximo numero de datos: ', nData
err1 = -1
err1 =  TJ2RDBGI(sizeData, nTabs, data, errC)
write(*,*)'**************************'
if (err1 .ne. 0) then
        err2 = TJ2RDBPERR(errC)
else
        i=1
        dowhile( data(i) .ne. char(0) .and. i .lt. sizeData)
                write(*,'(A1,$)')data(i)
                i = i + 1
        enddo
end if
write(*,*)'**************************'
STOP
END
```

5.1.2.2. Código C que usa la función TJ2RDBGI

```
#include <stdio.h>
#include <malloc.h>

extern int TJ2RDBGI_ ( int *, int *, char *, int *);
int main()
{
```

```
        char            *data;
        int             sizeData, nTabs, errC, merrCor;

        /* Se define el tamaño (en Octetos) de datos que se pueden recibir */
        sizeData = 100000;
        data = NULL;
        if ( (data = (char *) calloc ( (size_t) sizeData, sizeof(char) )) == NULL){
                printf("Se produjo un error reservando memoria para la informacion
        de la base de datos\n");
                return(0);
        }
        /*Se envia la consulta al servidor de datos */
        merrCor = TJ2RDBGI_ ( &sizeData, &nTabs, data, &errC);

        if (merrCor != 0){
        printf("Se produjo un error en la consulta\n");
        TJ2RDBPERR (&errC);
        }
        else{
                printf("%s", data);
                fflush(stdout);
        }
        free(data);
        fflush(stdin);

        return(0);

}
```

5.2. La función TJ2RDBS

El propósito de esta función es enviar una consulta al gestor de base de datos. Permite enviar una consulta de tipo SELECT arbitraria y recibir el resultado en forma de tabla de datos. En el *Apéndice 1. Bases de datos relacionales y consultas con SQL* se dan unas notas sobre la sintaxis de las sentencias de consulta a la base de datos que se pueden introducir usando esta función.

Esta función no permitirá la inserción o modificación de datos en la base de datos, usando sentencias SQL del tipo INSERT, UPDATE o DELETE por ejemplo. Si se intenta usar una sentencia de este tipo, la ejecución de la función fallará, devolviendo un código de error indicativo de lo sucedido.

5.2.1. Descripción detallada de TJ2RDBS

```
int TJ2RDBS ( char    *select,
              int     *sizeSelect,
              int     *sizeCNames,
              int     *sizeTypes,
              int     *sizeData,
              char    *cNames,
              char    *types,
```

```
char    *data,
int     *nRows,
int     *nCols,
int     *errC)
```

Argumentos de entrada:

select variable de tipo cadena de caracteres, declarada en el programa que llama a esta función, en la que se pasa a la función la consulta SQL para la base de datos.

sizeSelect variable de tipo entero en que se pasa a la función el tamaño (en octetos) reservado en la memoria del programa que llama a esta función para almacenar la consulta SQL. El valor de este argumento ha de ser menor o igual que el tamaño reservado para la variable **select**; en caso contrario pueden producirse invasiones de memoria o errores inesperados.

sizeCNames variable de tipo entero en la que se pasa a la función el tamaño (en octetos) reservado en la memoria del programa que llama a esta función para almacenar los nombres de las columnas que se reciben como resultado de la consulta (argumento **cNames**).El valor de este argumento ha de ser menor o igual que el tamaño reservado para la variable **cNames**; en caso contrario pueden producirse invasiones de memoria o errores inesperados.

sizeTypes variable de tipo entero en la que se pasa a la función el tamaño (en octetos) reservado en la memoria del programa que llama a esta función para almacenar los tipos de datos que contienen las columnas de datos obtenidos como resultado de la consulta a la base de datos (argumento **types**). El valor de este argumento ha de ser menor o igual que el tamaño reservado para la variable **types**; en caso contrario pueden producirse invasiones de memoria o errores inesperados.

sizeData variable de tipo entero en la que se pasa a la función el tamaño (en octetos) reservado en la memoria del programa que llama a esta función para almacenar el resultado de la consulta. El valor de este argumento ha de ser menor o igual que el tamaño reservado

para la variable **data**; en caso contrario pueden producirse invasiones de memoria o errores inesperados.

Argumentos de salida:

cNames variable de tipo cadena de caracteres declarada en el programa que llama a esta función con tamaño de al menos **sizeCNames** octetos. En esta variable se reciben los títulos de las columnas obtenidas como resultado de la consulta.

types variable de tipo cadena de caracteres declarada en el programa que llama a esta función con tamaño de al menos **sizeTypes** octetos. En esta variable se reciben los tipos de datos de las columnas de datos obtenidas como resultado de la consulta.

data variable de tipo cadena de caracteres declarada en el programa que llama a esta función con tamaño de al menos **sizeData** octetos. En esta variable se reciben las columnas de datos obtenidas como resultado de la consulta.

nRows variable de tipo entero en la que se recibe el número de filas de datos obtenidas en la consulta.

nCols variable de tipo entero en la que se recibe el número de columnas de datos obtenidas en la consulta.

errC variable de tipo entero en la que se recibe un código de error indicativo de lo sucedido en caso de que se produzca algún error en la ejecución de la función.

El resultado de una consulta de este tipo (SELECT) es siempre una tabla de datos, que se reciben en la variable **data**, como una cadena de caracteres ASCII con una estructura de tabla con **nCols** columnas y **nRows** filas, tal como la que sigue:

```
Dato1_1, TAB, Dato1_2, TAB, ..., DATO1_N, RETURN
Dato2_1, TAB, Dato2_2, TAB, ..., DATO2_N, RETURN
...
DatoM_1, TAB, DatoM_2, TAB, ..., DATOM_N, RETURN
```

donde **Datoi_j** es el dato obtenido para la columna **j** y la fila **i**, TAB representa un carácter de tabulación (o el carácter que se use para separar campos) y RETURN un carácter de nueva línea (o el carácter que se use como separador de líneas). Esta cadena de caracteres acabará en un carácter NULL.

Puesto que en la estructura actual de la base de datos algunos campos pueden contener los caracteres de tabulación y/o nueva línea, se ha introducido la posibilidad de modificar los caracteres usados como separadores de campos y líneas respectivamente. Se puede modificar los caracteres usados para separar campos y líneas a través de las variables de entorno TJ2RDB_COL_SEPARATOR y TJ2RDB_LINE_SEPARATOR (ver sección 4.3. Configuración de la librería) , o haciendo uso desde el programa principal de la función TJ2RDBSETSEP (ver sección 5.10. La función TJ2RDBSETSEP). Con la función TJ2RDBGETSEP se pueden obtener en cualquier momento los caracteres que se están usando como separadores de campos y líneas en llamadas a las funciones TJ2RDBGI y TJ2RDBS. **Se recomienda, para evitar confusión entre los caracteres separadores de campos y líneas y el contenido de campos, usar como separadores caracteres con código decimal ASCII por encima del 128.**

En la variable **cNames** se recibirán los nombres de las columnas consultadas separados por un carácter de tabulación

En la variable **types** se recibirán caracteres indicativos del tipo de dato que contiene cada columna recibida separados por caracteres de tabulación. C corresponde a un dato de tipo cadena de caracteres F corresponde a un dato de punto flotante I corresponde a un dato de tipo entero.

Los valores **nCols**, **nRows** y **types** pueden ser útiles para permitir a un programa procesar automáticamente el resultado de la consulta a la base de datos.

En realidad, una vez que se ha introducido una consulta SELECT en esta función, se conoce la estructura del resultado. El resultado será una tabla de datos, con un número de columnas conocido y cuyos nombres son los seleccionados en la sentencia de consulta. A pesar de esto, se ha diseñado esta función de modo que devuelva esa información en las variables **nCols**, **cNames** y **types** para facilitar la automatización de consultas SELECT a la base de datos desde programas, pudiendo desarrollarse programas interactivos de consulta que procesen consultas SELECT arbitrarias

introducidas por el usuario del programa y traten de forma automatizada el resultado obtenido en la consulta a la base de datos para su representación.

La función devuelve un dato de tipo entero para indicar si la operación se completó con éxito (0) o si ocurrió algún error (distinto de cero).

Nota: Si el resultado de la consulta SELECT no cabe en el espacio reservado por el programa que llama a la función para recibir el resultado (variable **data**), la función devolverá el resultado de la consulta rellenando la última fila que cabe completa en el espacio reservado en **data**, devolviendo además un código de error que indica que no se reservó espacio suficiente en la variable data para recibir el resultado de la consulta..

5.2.2. Ejemplos de uso de TJ2RDBS

5.2.2.1. Código FORTRAN que usa la función TJ2RDBS

```
      PROGRAM TESTRDBS
      implicit none
      integer       nData, sSelect, sTypes, sCNames
      parameter      ( nData = 32767, sSelect = 2048)
      parameter      ( sTypes = 256, sCNames = 2048 )

      character*1    data(1:nData)
      character*1    types(1:sTypes)
      character*1    cNames(1:sCNames)
      character*2048  select
      integer       nTabs, iErr, err1, err2
      integer       sizeData, i
      integer       TJ2RDBPERR, TJ2RDBS
      integer       sizeSelect, sizeCNames, sizeTypes
      integer       nRows, nCols

      external TJ2RDBPERR, TJ2RDBS

      sizeData = nData
      sizeCNames = sCNames
      sizeTypes = sTypes
      sizeSelect = sSelect

      write(*,*)' Programa de prueba de Select'
      write(*,*)'Maximo numero de datos: ', nData, ' octetos'
      write(*,*)'Reservados para los tipos : ', sTypes, ' octetos'
      write(*,*)'Reservados para los titulos : ', sCNames, ' octetos'
      write(*,*)'Reservados para la select : ', sSelect, ' octetos'
      write(*,*)
      write(*,'(a)')'Introducir la consulta:'
      read(5,'(a)')select
      write(*,*)'**************************'

      err1 = -1
      err1 = TJ2RDBS( select, sizeSelect, sizeCNames, sizeTypes,
     +         sizeData, cNames, types, data, nRows, nCols, iErr)
```

```
      err2 = -1
      if (err1 .ne. 0) then
            err2 = TJ2RDBPERR(iErr)
      else
            i=1
            write(*,*)'Recibidas '
            write(*,*) nRows, ' filas'
            write(*,*)nCols, ' columnas'
            write(*,*)'===================='
            write(*,*)'     TITULOS de las columnas :'
            write(*,*)'_____________________'
            dowhile( cNames(i) .ne. char(0) .and. i .lt. sizeCNames)
                  write(*,'(A1,$)')cNames(i)
            i = i + 1
            enddo
            i=1
            write(*,*)
            write(*,*)'===================='
            write(*,*)'     TIPOS de datos :'
            write(*,*)'_____________________'
            dowhile( types(i) .ne. char(0) .and. i .lt. sizeCNames)
                  write(*,'(A1,$)')types(i)
            i = i + 1
            enddo
            i=1
            write(*,*)
            write(*,*)'===================='
            write(*,*)'     DATOS :'
            write(*,*)'_____________________'
            dowhile( data(i) .ne. char(0) .and. i .lt. sizeData)
                  write(*,'(A1,$)')data(i)
            i = i + 1
            enddo
      end if
      write(*,*)'*************************'

      STOP
      END
```

5.2.2.2. Código C que usa la función TJ2RDBS

```
#include <stdio.h>
#include <malloc.h>

extern int TJ2RDBS_ ( char *, int *, int *, int *, int *,
                char *, char *, char *, int *, int *, int *);
extern int TJ2RDBPERR_ (int *);

main () {
        int             szSel, szTi, szTy, szDa;
        int             nRows, nCol, errC, mierror;
        char            *select, *cNames, *types, *data;

        szSel = 1024;
        select = NULL;
        szTi = 40960;
```

```
        cNames = NULL;
        szTy = 40960;
        types = NULL;
        szDa = 640000;
        data = NULL;
        if ( ( select = (char *) calloc ( (size_t) szSel, sizeof(char) )) == NULL){
                printf("Se produjo un error reservando memoria para la consulta\n");
        }

        if ( ( cNames = (char *) calloc ( (size_t) szTi, sizeof(char) )) == NULL)
        {
                printf("Se produjo un error reservando memoria para los titulos\n");
                free( select);
        }

        if ( ( types = (char *) calloc ( (size_t) szTy, sizeof(char) )) == NULL) {
                printf("Se produjo un error reservando memoria para los tipos\n");
                free( select);
                free( cNames);
        }

        if ( ( data = (char *) calloc ( (size_t) szDa, sizeof(char) )) == NULL){
                printf("Se produjo un error reservando memoria para el
resultado\n");
                free( select);
                free( cNames);
                free ( types);
        }

        printf("Introducir la consulta\n");
        fflush(stdout);
        fflush(stdin);
        fgets( select, szSel, stdin);
        errC = 0;
        mierror=0;

        mierror = TJ2RDBS_( select, &szSel, &szTi, &szTy, &szDa,
                cNames, types, data, &nRows, &nCol, &errC);

        if (mierror != 0){
                printf("Se produjo un error en la consulta\n");
                TJ2RDBPERR_ (&errC);
        }
}
```

5.3. La función TJ2RDBDEL

Esta función permite eliminar un valor existente en un campo de la base de datos. Para seleccionar el valor a eliminar habrá que especificar el nombre de la columna afectada por la supresión del valor. Para identificar la fila afectada hay que especificar un nombre de columna que se usa como clave y el valor de esta columna para el que se eliminará el dato en la columna afectada. En general la columna que se usará como clave para identificar la fila afectada será el campo **NDES** (número de descarga de

plasma de TJ-II), puesto que en esta base de datos la mayoría de datos están asociados a una descarga de plasma de TJ-II.

Todos los argumentos se han de pasar por referencia, por compatibilidad con códigos FORTRAN.

5.3.1. Descripción detallada de TJ2RDBDEL

int **TJ2RDBDEL** (char **keyNam*,
char **keyVal*,
char **fieNam*,
int **errC*)

Argumentos de entrada:

keyNam es una cadena de caracteres acabada en el carácter NULL indicando el nombre de la columna que se usa como clave para seleccionar la fila de la tabla afectada.

keyVal es una cadena de caracteres, declarada en el programa que llama a esta función, en la que se envía el valor de la columna clave para la que se quieren eliminar datos.

colNam es una cadena de caracteres, acabada en el carácter NULL, declarada en el programa que llama a esta función, en la que se identifica el nombre de la columna para la que se quiere eliminar un dato de la fila identificada por los valores de los argumentos **keyNam** y **keyVal**.

Argumentos de salida:

errC es una variable de tipo entero, declarada en el programa principal, en el que se recibe, al retorno de la función, un código de error indicativo de lo sucedido en caso de error. Se puede obtener una descripción de la situación asociada a un código de error usando la función **TJ2RDBGERR**.

Como el resto de funciones, retorna un número entero para indicar el éxito o no de la operación sobre la base de datos. Cuando se produce algún error en la manipulación de la base de datos, esta función retorna un valor diferente de cero, y cero en caso contrario.

Notas:

En los valores de la clave no se pueden usar meta-caracteres como el "*" para representar varios valores de clave. Una llamada a esta función que contenga el carácter "*" en el valor de **keyVal** fallará.

Tanto los nombres como los valores de las columnas de vistas de la base de datos se tratan, para mayor generalidad, como cadenas de caracteres por esta función, aunque los valores de algunas columnas de la base de datos serán datos numéricos, enteros o de coma flotante. Habrá que hacer las oportunas conversiones de datos antes de llamar a esta función

5.3.2. Ejemplos de uso de TJ2RDBDEL

5.3.2.1. Código FORTRAN que usa la función TJ2RDBDEL

```
PROGRAM TESTRDBDEL

implicit none

integer        sC, sV
parameter      ( sC = 256, sV = 2048 )

integer        iErr, err1, err2
integer        sizeC, sizeV
integer        TJ2RDBPERR, TJ2RDBUP, TJ2RDBDEL
character*512   clave, claveV, dato
external         TJ2RDBPERR, TJ2RDBDEL

sizeC = sC
sizeV = sV

write(*,*)' Programa de prueba de borrado'
write(*,*)'Reservados para nombre de campo : ', sC, ' octetos'
write(*,*)'Reservados para valores de campos : ', sV, ' octetos'
write(*,*)

write(*,*)'**************************'
write(clave,'(A)')'NDes'
write(*,*)'Introducir el nombre del campo a modificar '
read(*,'(A)')dato
write(*,*)'Numero de descarga?'
read(*,'(A)')claveV
write(*,*)
write(*,*)'     enviando datos, ...'

err1 = -1
err1 = TJ2RDBDEL( clave, claveV, dato, iErr)
write(*,*)
write(*,*)'Fin de la consulta'

err2 = -1
if (err1 .ne. 0) then
```

```
            err2 = TJ2RDBPERR(iErr)
      else
            write(*,*)
            write(*,*)'Operacion concluida'
      end if
      write(*,*)'*************************'

      STOP
      END
```

5.3.2.2. Código C que usa la función TJ2RDBDEL

```
#include <stdio.h>
#include <malloc.h>

extern TJ2RDBDEL_ ( char *, char *, char *, char *, int *);
extern int TJ2RDBPERR_ (int *);

int main(){
        char            keyNam[32], value[32], campo[32];
        int             mierror, errC;
                /* se inicializan variables */
        memset(keyNam, 0, sizeof(keyNam));
        memset(value, 0, sizeof(value));
        memset(campo, 0, sizeof(campo));


        printf("Introducir el nombre de la columna afectada\n");
        fflush(stdout);
        fflush(stdin);
        gets( campo);

        sprintf(keyNam, "NDes");
        fflush(stdout);
        printf("Numero de descarga ?\n");
        fflush(stdout);
        fflush(stdin);
        gets( value);
        fflush(stdin);
        fflush(stdout);

        mierror = TJ2RDBDEL_( keyNam, value, campo, &errC);
        if (mierror != 0)       {
                printf("Se produjo un error al eliminar el dato solicitado\n");

                TJ2RDBPERR_ (&errC);

        }
}
```

5.4. La función TJ2RDBUP

Esta función permite modificar un valor existente o introducir un valor para un campo de la base de datos. Para seleccionar el valor a modificar / introducir habrá que indicar el nombre de la columna objeto de manipulación. Para identificar la fila (o filas)

afectada hay que especificar un nombre de columna que se usa como clave y el valor de esta columna para el que se eliminará el dato en la columna afectada. En general la columna que se usará como clave para identificar la fila afectada será el campo **NDES** (número de descarga de plasma de TJ-II), puesto que en esta base de datos la mayoría de datos están asociados a una descarga de plasma de TJ-II.

Todos los argumentos se han de pasar por referencia, por compatibilidad con códigos FORTRAN.

5.4.1. Descripción detallada de TJ2RDBUP

int **TJ2RDBUP** (char **keyNam*,
char **keyVal*,
char **fieNam*,
char **colVal*,
int **errC*)

Argumentos de entrada:

keyNam es una cadena de caracteres acabada en el carácter NULL indicando el nombre de la columna que se usa como clave para seleccionar la fila de la tabla afectada.

keyVal es una cadena de caracteres declarada en el programa que llama a esta función en la que se envía el valor de la columna clave para la que se quieren introducir datos.

colNam es una cadena de caracteres, acabada en el carácter NULL y declarada en el programa que llama a esta función, en la que se identifica el nombre de la columna para la que se quiere modificar o añadir un dato en la fila identificada por los valores de los argumentos **keyNam** y **keyVal**.

colVal es una cadena de caracteres, acabada en el carácter NULL y declarada en el programa que llama a esta función, en la que se identifica el valor que se quiere introducir/modificar en la columna identificada por el argumento campo y la fila identificada por **keyNam** y **keyVal**.

Argumentos de salida:

errC es una variable de tipo entero, declarada en el programa principal, en la que se recibe, al retorno de la función, un código de error indicativo de lo sucedido en caso de error. Se puede obtener una descripción de la situación asociada a un código de error usando la función TJ2RDBGERR.

Como el resto de funciones, retorna un número entero para indicar el éxito o no de la operación sobre la base de datos. Cuando se produce algún error en la manipulación de la base de datos, esta función retorna un valor diferente de cero, y cero en caso contrario.

Notas:

En los valores de la clave no se pueden usar meta-caracteres como el "*****" para representar varios valores de clave. Una llamada a esta función que contenga el carácter "*" en el valor de **keyVal** fallará.

Tanto los nombres como los valores de las columnas de vistas de la base de datos se tratan, para mayor generalidad, como cadenas de caracteres por esta función, aunque los valores de algunas columnas de la base de datos serán datos numéricos, enteros o de coma flotante. Habrá que hacer las oportunas conversiones de datos antes de llamar a esta función

5.4.2. Ejemplos de uso de TJ2RDBUP

5.4.2.1. Código FORTRAN que usa la función TJ2RDBUP

```
PROGRAM TESTRDBUP

implicit none

integer        sC, sV
parameter      ( sC = 256, sV = 2048 )

integer        iErr, err1, err2
integer        sizeC, sizeV
integer        TJ2RDBPERR, TJ2RDBUP, TJ2RDBDEL
character*512   clave, claveV, dato, datoV
external         TJ2RDBPERR, TJ2RDBUP

sizeC = sC
sizeV = sV

write(*,*)'Programa de prueba de Insercion'
```

```
      write(*,*)'Reservados para nombre de campo : ', sC, ' octetos'
      write(*,*)'Reservados para valores de campos : ', sV, ' octetos'
      write(*,*)
      write(*,*)'     Trabajando en modo modificar'
      write(*,*)'**************************'
      write(clave,'(A)')'NDes'
      write(*,*)'Introducir el nombre del campo a modificar '
      read(*,'(A)')dato
      write(*,*)'Introducir el valor del campo '
      read(*,'(A)')datoV
      write(*,*)'Numero de descarga?'
      read(*,'(A)')claveV
      write(*,*)
      write(*,*)'     enviando datos, ...'

      err1 = -1
      err1 = TJ2RDBUP( clave, claveV, dato, datoV, iErr)
      write(*,*)
      write(*,*)'Fin de la consulta'

      err2 = -1
      if (err1 .ne. 0) then
            err2 = TJ2RDBPERR(iErr)
      else
            write(*,*)
            write(*,*)'Operacion concluida'
      end if
      write(*,*)'**************************'

      STOP
      END
```

5.4.2.2. Código C que usa la función TJ2RDBUP

```
#include <stdio.h>
#include <malloc.h>

extern TJ2RDBUP_ ( char *, char *, char *, char *, int *);
extern int TJ2RDBPERR_ (int *);

int main(){
        char            keyNam[32], value[32], campo[32], campov[32];
        int             mierror, errC;
                /* se inicializan variables */
        memset(keyNam, 0, sizeof(keyNam));
        memset(value, 0, sizeof(value));
        memset(campo, 0, sizeof(campo));
        memset(campov, 0, sizeof(campov));


        printf("Introducir el nombre de la columna afectada\n");
        fflush(stdout);
        fflush(stdin);
        gets( campo);
        printf("Introducir el valor\n");
```

```
        fflush(stdout);
        fflush(stdin);
        gets( campov);

        sprintf(keyNam, "NDes");
        fflush(stdout);
        printf("Numero de descarga ?\n");
        fflush(stdout);
        fflush(stdin);
        gets( value);
        fflush(stdin);
        fflush(stdout);

        mierror = TJ2RDBUP_( keyNam, value, campo, campov, &errC);
        if (mierror != 0)      {
                printf("Se produjo un error al modificar el dato solicitado\n");

                TJ2RDBPERR_ (&errC);

        }
}
```

5.5. La función TJ2RDBPERR

Esta función se encarga de imprimir en la salida estándar del programa que la llama un mensaje de error asociado a un código de error (entero) que se le pasa como argumento.

Se supone que el programa principal tiene asociada una salida estándar y está disponible para enviar mensajes al usuario por ella. En caso contrario no funcionará correctamente.

5.5.1. Descripción detallada de TJ2RDBPERR

int **TJ2RDBPERR** (int **errCo*)

La función sólo tiene un argumento de entrada, **errCo**, que ha de ser pasado por referencia, para compatibilidad con FORTRAN. **errCo** es un entero que ha de ser declarado en el programa que llama a esta función. En este argumento se le pasa a la función un código de error. La función imprime en la salida estándar del programa un mensaje de error descriptivo de la situación de error asociada al código que recibe como argumento.

Si no se produce ningún error al intentar obtener el mensaje asociado al código **TJ2RDBPERR** retorna 0. En caso de error retorna -1.

Para obtener el mensaje de error asociado a un código **TJ2RDBPERR** llama a la función **TJ2RDBGERR** (ver 5.6:*La función TJ2RDBGERR*).

5.5.2. Ejemplos de uso de TJ2RDBPERR

5.5.2.1. Código FORTRAN que usa la función TJ2RDBPERR

Para este ejemplo se remite al lector a los ejemplos de uso de las funciones **TJ2RDBS**, **TJ2RDBUP** y **TJ2RDBDEL** en programas FORTRAN, ejemplos en los que aparecen también llamadas a la función **TJ2RDBPERR**.

5.5.2.2. Código C que usa la función TJ2RDBPERR

Para este ejemplo se remite al lector a los ejemplos de uso de las funciones **TJ2RDBS**, **TJ2RDBUP** y **TJ2RDBDEL** en programas C, ejemplos en los que aparecen también llamadas a la función **TJ2RDBPERR**.

5.6. La función TJ2RDBGERR

La función **TJ2RDBGERR** permite obtener, en una cadena de caracteres, un mensaje descriptivo de una situación de error cuyo código de error se le pasa como argumento.

5.6.1. Descripción detallada de TJ2RDBGERR

int **TJ2RDBGERR** (int **ierrCo*,
char **msg*,
int **sizeMsg*)

Argumentos de entrada:

iErrCo es un argumento de entrada de tipo entero que ha de ser declarado en el programa que llama a esta función. En este argumento se le pasa a **TJ2RDBGERR** un código de error, en general, recibido tras la ejecución de otra función de la librería **tj2RdbC**.

msg es una cadena de caracteres que ha de ser declarada en el programa que llama a esta función. En esta cadena de caracteres la función escribirá un mensaje de error asociado con el código de error **ierrCo**. Ha de ser declarada con una dimensión suficiente para almacenar el mensaje de error; esta dimensión ha de ser

mayor o igual que el valor que se pasa a la función en el argumento **sizeMsg**.

sizeMsg es un entero declarado en el programa principal que llama a esta función. En este argumento se pasa a la función el tamaño (en octetos) que se ha reservado en el programa que llama a la función para almacenar el mensaje descriptivo del error asociado al código de error que se le pasa en **ierrCo**.

Para independizar al máximo el código de la librería cliente **tj2RdbC** del código del servidor RDB del TJ-II se han distribuido también los errores. Hay diferentes tipos de errores. Errores del sistema ONC RPC, errores del lado del cliente y errores del lado del servidor. Los errores están tabulados, con códigos asociados de forma separada para cada tipo. Para el caso de los errores de RPC se usan los códigos proporcionados por la librería RPC [Bloomer92] incrementados en una constante para evitar coincidencias con códigos del cliente o del servidor.

Los errores del lado del cliente están tabulados, habiendo una correspondencia biunívoca entre códigos y mensajes de error asociados. Cuando esta función recibe uno de los códigos de error que se encuentra en la tabla de códigos de error del lado cliente simplemente rellena la cadena **msg** con el mensaje asociado al código recibido.

Los errores del lado del servidor están tabulados en el servidor. Para acceder a estos códigos esta función establece una nueva conexión RPC con el servidor y consulta el mensaje asociado a un código. Esto permite que en el lado del servidor se puedan modificar los códigos de error para contemplar nuevas situaciones y puedan estar disponibles para el cliente sin necesidad de recompilar ni el servidor ni la librería cliente, que puede estar instalada en diferentes plataformas cliente. Por el contrario, debido a esta conexión con el servidor para consultar los mensajes asociados a códigos de errores producidos en el lado del servidor, es posible que esta función no consiga obtener el mensaje de error asociado a uno de estos códigos de error, si se produce un problema de conectividad entre el servidor y el cliente.

La función **TJ2RDBGERR** retorna 0 cuando consigue obtener el mensaje de error asociado al código recibido como argumento, o -1 en caso de que se produzca algún error que lo impida.

5.6.2. Ejemplos de uso de TJ2RDBGERR

5.6.2.1. Código FORTRAN que usa la función TJ2RDBGERR

```
      program TESTTJ2RDBGERR

      parameter     tam=512

      integer code,  ierr, errc
      character*512 msg

      external TJ2RDBGERR
      integer TJ2RDBGERR

      write(*,*)'Introducir el codigo de error a consultar'
      read(*,'(i)')code
      errc = TJ2RDBGERR (code, msg, tam)
      if ( errc .eq. 0 ) then
        write(*,'(a,i,a,a)')'codigo ', code, ' ->', msg
      else
        write(*,*)'Se produjo un error al consultar el mensaje'
      end if

      STOP
      END
```

5.6.2.2. Código C que usa la función TJ2RDBGERR

```
#include <stdio.h>

extern  int   TJ2RDBGERR( int *, char *, int *);

int main(){
        char            Msg[512];
        int                     tamano, code;

        tamano = 512;
        printf("Introducir el código de error\n");
        fflush(stdout);
        scanf("%d", &code);
        fflush(stdin);
        if ( TJ2RDBGERR( &code, Msg, &tamano) == 0){
                printf("\nCodigo de error %d -> : %s\n", code, Msg);
                fflush(stdout);
                return(0);
        }
        else{
                printf("\n%d - %s\n", code, Msg);
                fflush(stdout);
                fflush(stdout);
                return(-1);
        }
        return (0);
```

```
}
```

5.7. La función TJ2RDBGETSZ

Esta función permite conocer el tamaño preestablecido para almacenar los nombres y valores de columnas de la base de datos que se pasan a las funciones **TJ2RDBUP** y **TJ2RDBDEL**. Esas funciones asumen, por defecto, que se usan **SIZE_COLUMN_DEF (32)** octetos para los nombres de las columnas y **SIZE_VALUE_DEF (32)** octetos para almacenar los valores de las columnas. Estos valores por defecto pueden ser modificados usando la función **TJ2RDBSETSZ** (ver 5.8). Por medio de **TJ2RDBGETSZ** se pueden obtener los valores que están siendo usados, que podrían haber sido establecidos previamente con **TJ2RDBSETSZ**

La función **TJ2RDBGETSZ** devuelve un valor 0 si la ejecución se realiza con éxito o un valor diferente de 0 si se produce algún error durante la ejecución de la función.

Todos los argumentos han de ser pasados a la función por referencia, por compatibilidad con FORTRAN.

5.7.1. Descripción detallada de la función TJ2RDBGETSZ

int **TJ2RDBGETSZ** (int **szCol*,
int **szVal*,
int **errC*)

Argumentos de entrada:

szCol Variable de tipo entero en la que se pasa el tamaño en octetos que se debe asumir para los nombres de columnas que se pasan a las funciones **TJ2RDBUP** y **TJ2RDBDEL**.

szVal Variable de tipo entero en la que se pasa el tamaño en octetos que se debe asumir para los valores de columnas que se pasan a las funciones **TJ2RDBUP** y **TJ2RDBDEL**.

Argumentos de salida:

errC Variable de tipo entero en la que se recibe, al retorno de la función, un código indicativo del error ocurrido, caso de haberse producido alguno.

5.7.2. Ejemplos de uso de TJ2RDBGETSZ

5.7.2.1. Código FORTRAN que usa la función TJ2RDBGETSZ

```
      PROGRAM testGetsz

      integer   TJ2RDBGETSZ, TJ2RDBSETSZ
      external  TJ2RDBGETSZ, TJ2RDBSETSZ

      integer   szCols, szVals, errR, errC

      errR = TJ2RDBGETSZ( szCols, szVals, errC)
      if ( errC .eq. 0 ) then
        write(*,*)'Los tamanos actuales son:'
        write(*,'(a,i)')' nombres de columnas : ', szCols
        write(*,'(a,i)')' valores de columnas : ', szVals

        write(*,*)'Introducir los nuevos valores'
        write(*,*)'nombres de columnas'
        read(*,'(i)')szCols
        write(*,*)'Valores'
        read(*,'(i)')szVals
        errR = TJ2RDBSETSZ( szCols, szVals, errC)
        if ( errC .eq. 0 ) then
          szCols = 0
          szVals = 0
          errR = TJ2RDBGETSZ( szCols, szVals, errC)
          if ( errC .eq. 0 ) then
            write(*,*)'Los tamanos actuales son:'
            write(*,'(a,i)')' nombres de columnas : ', szCols
            write(*,'(a,i)')' valores de columnas : ', szVals
          else
            write(*,*)'Se produjo un error en la consulta'
          end if
         endif

      else
        write(*,*)'Se produjo un error en la consulta'

      end if

      STOP
      END
```

5.7.2.2. Código C que usa la función TJ2RDBGETSZ

```
#include <stdio.h>
extern int TJ2RDBGETSZ( int * , int *, int *);
extern int TJ2RDBSETSZ( int * , int *, int *);


int main(){
        int             szCols, szVals, errR, errC;

        errR = 0;
        errR = TJ2RDBGETSZ ( &szCols, &szVals, &errC);
        if ( errR == 0 ){
```

```
            printf("Los tamanos actuales son :\n\tpara nombres de columnas:
%d\n\tpara valores: %d", szCols, szVals);
            fflush(stdout);
            printf("\n\nIntrocudir los nuevos valores\npara los nombres de
columnas ");
            scanf("%d", &szCols);
            printf("\npara los valores ");
            scanf("%d", &szVals);

            errR = TJ2RDBSETSZ( &szCols, &szVals, &errC);
            if ( errR == 0) {
                    errR = TJ2RDBGETSZ ( &szCols, &szVals, &errC);
                    if ( errR == 0 ){
                            printf("Los tamanos actuales son :\n\tpara nombres de
columnas: %d\n\tpara valores: %d", szCols, szVals);
                            fflush(stdout);
                    }
            }
      }
      else{
            printf("Se produjo un error");
            fflush(stdout);
      }

    }
```

5.8. La función TJ2RDBSETSZ

La función **TJ2RDBSETSZ** permite modificar el tamaño reservado para almacenar los nombres y valores de columnas de la base de datos que se pasan a las funciones **TJ2RDBUP** y **TJ2RDBDEL** como argumentos de entrada. Esas funciones suponen por defecto tamaños de **SIZE_COLUMN_DEF** (32) octetos para los nombres de las columnas y **SIZE_VALUE_DEF** (32) octetos para almacenar los valores de las columnas. Por medio de esta función se pueden especificar otros valores, si es necesario, de modo que sólo hay que especificar estos tamaños una vez, y no en cada llamada a una de las funciones **TJ2RDBUP** y **TJ2RDBDEL**. Los valores establecidos por esta función se guardarán como variables de entorno del proceso, de modo que sólo se necesita llamar a esta función una vez para ajustar los tamaños necesarios por cada ejecución del programa cliente, aunque después de la llamada a esta función se produzcan varias llamadas a las funciones **TJ2RDBDEL** y **TJ2RDBUP**.

La función **TJ2RDBSETSZ** devuelve un valor 0 si la ejecución se realiza con éxito o un valor diferente de 0 si se produce algún error durante su ejecución

Todos los argumentos han de ser pasados a la función por referencia, por compatibilidad con FORTRAN.

5.8.1. Descripción detallada de la función TJ2RDBSETSZ

int **TJ2RDBSETSZ** (int **szCol*,
int **szVal*,
int **errC*)

Argumentos de entrada:

szCol Variable entera en la que se pasa el tamaño en octetos que se debe asumir para los nombres de columnas que se pasan a las funciones **TJ2RDBUP** y **TJ2RDBDEL**.

szVal Variable entera en la que se pasa el tamaño en octetos que se debe asumir para los valores de columnas que se pasan a las funciones **TJ2RDBUP** y **TJ2RDBDEL**.

Argumentos de salida:

errC Variable de tipo entero en la que se recibe, al retorno de la función, un código indicativo del error ocurrido, caso de haberse producido alguno.

5.8.2. Ejemplos de uso de TJ2RDBSETSZ

5.8.2.1. Código FORTRAN que usa la función TJ2RDBSETSZ

Para este ejemplo se remite al lector a la sección 5.7.2.1. (*Código FORTRAN que usa la función TJ2RDBGETSZ*) donde se hace uso también de la función **TJ2RDBSETSZ**.

5.8.2.2. Código C que usa la función TJ2RDBSETSZ

Para este ejemplo se remite al lector a la sección 5.7.2.2. *(Código C que usa la función TJ2RDBGETSZ*) en la cual se muestra un ejemplo de uso de la función TJ2RDBSETSZ desde un programa C.

5.9. La función TJ2RDBGETSZ

Esta función permite conocer el tamaño preestablecido para almacenar los nombres y valores de columnas de la base de datos que se pasan a las funciones **TJ2RDBUP** y **TJ2RDBDEL**. Esas funciones asumen, por defecto, que se usan **SIZE_COLUMN_DEF (32)** octetos para los nombres de las columnas y **SIZE_VALUE_DEF (32)** octetos para almacenar los valores de las columnas. Estos

valores por defecto pueden ser modificados usando la función **TJ2RDBSETSZ** (ver 5.8). Por medio de **TJ2RDBGETSZ** se pueden obtener los valores que están siendo usados, que podrían haber sido establecidos previamente con **TJ2RDBSETSZ**

La función **TJ2RDBGETSZ** devuelve un valor 0 si la ejecución se realiza con éxito o un valor diferente de 0 si se produce algún error durante la ejecución de la función.

Todos los argumentos han de ser pasados a la función por referencia, por compatibilidad con FORTRAN.

5.9.1. Descripción detallada de la función TJ2RDBGETSZ

int **TJ2RDBGETSZ** (int **szCol*,
int **szVal*,
int **errC*)

Argumentos de entrada:

szCol Variable de tipo entero en la que se pasa el tamaño en octetos que se debe asumir para los nombres de columnas que se pasan a las funciones **TJ2RDBUP** y **TJ2RDBDEL**.

szVal Variable de tipo entero en la que se pasa el tamaño en octetos que se debe asumir para los valores de columnas que se pasan a las funciones **TJ2RDBUP** y **TJ2RDBDEL**.

Argumentos de salida:

errC Variable de tipo entero en la que se recibe, al retorno de la función, un código indicativo del error ocurrido, caso de haberse producido alguno.

5.9.2. Ejemplos de uso de TJ2RDBGETSZ

5.9.2.1. Código FORTRAN que usa la función TJ2RDBGETSZ

```
PROGRAM testGetsz

integer   TJ2RDBGETSZ, TJ2RDBSETSZ
external  TJ2RDBGETSZ, TJ2RDBSETSZ

integer   szCols, szVals, errR, errC

errR = TJ2RDBGETSZ( szCols, szVals, errC)
if ( errC .eq. 0 ) then
```

```
      write(*,*)'Los tamanos actuales son:'
      write(*,'(a,i)')' nombres de columnas : ', szCols
      write(*,'(a,i)')' valores de columnas : ', szVals

      write(*,*)'Introducir los nuevos valores'
      write(*,*)'nombres de columnas'
      read(*,'(i)')szCols
      write(*,*)'Valores'
      read(*,'(i)')szVals
      errR = TJ2RDBSETSZ( szCols, szVals, errC)
      if ( errC .eq. 0 ) then
        szCols = 0
        szVals = 0
        errR = TJ2RDBGETSZ( szCols, szVals, errC)
        if ( errC .eq. 0 ) then
          write(*,*)'Los tamanos actuales son:'
          write(*,'(a,i)')' nombres de columnas : ', szCols
          write(*,'(a,i)')' valores de columnas : ', szVals
        else
          write(*,*)'Se produjo un error en la consulta'
        end if
       endif

    else
      write(*,*)'Se produjo un error en la consulta'

    end if

    STOP
    END
```

5.9.2.2. Código C que usa la función TJ2RDBGETSZ

```
#include <stdio.h>
extern int TJ2RDBGETSZ( int * , int *, int *);
extern int TJ2RDBSETSZ( int * , int *, int *);


int main(){
        int             szCols, szVals, errR, errC;

        errR = 0;
        errR = TJ2RDBGETSZ ( &szCols, &szVals, &errC);
        if ( errR == 0 ){
                printf("Los tamanos actuales son :\n\tpara nombres de columnas:
%d\n\tpara valores: %d", szCols, szVals);
                fflush(stdout);
                printf("\n\nIntrocudir los nuevos valores\npara los nombres de
columnas ");
                scanf("%d", &szCols);
                printf("\npara los valores ");
                scanf("%d", &szVals);

                errR = TJ2RDBSETSZ( &szCols, &szVals, &errC);
                if ( errR == 0) {
                        errR = TJ2RDBGETSZ ( &szCols, &szVals, &errC);
                        if ( errR == 0 ){
```

```
                        printf("Los tamanos actuales son :\n\tpara nombres de
columnas: %d\n\tpara valores: %d", szCols, szVals);
                        fflush(stdout);
                    }
                }
        }
        else{
                printf("Se produjo un error");
                fflush(stdout);
        }
    }
```

5.10. La función TJ2RDBSETSEP

La función **TJ2RDBSETSEP** permite modificar el carácter usado para separar el contenido de diferentes campos y líneas en las llamadas a las funciones TJ2RDBGI y TJ2RDBS. Los valores establecidos por esta función se guardarán como variables de entorno del proceso, de modo que sólo es necesario llamar una vez al principio del programa principal a esta función para establecer los caracteres usados; en las llamadas posteriores a TJ2RDBGI o TJ2RDBS se usarán siempre los separadores establecidos.

Los caracteres separadores se pueden establecer también, de forma permanente para varias ejecuciones de un mismo programa, usando las variables de entorno TJ2RDB_COL_SEPARATOR y TJ2RDB_LINE_SEPARATOR.

Por defecto los separadores de campos y líneas son los caracteres ASCII 9 (tabulador) y 10 (nueva línea) respectivamente.

La función **TJ2RDBSETSEP** devuelve un valor 0 si la ejecución se realiza con éxito o un valor diferente de 0 si se produce algún error durante su ejecución

Todos los argumentos han de ser pasados a la función por referencia, por compatibilidad con FORTRAN.

5.10.1. Descripción detallada de la función TJ2RDBSETSEP

int **TJ2RDBSETSZ** (int **fSepl*,
int **lSepl*,
int **errC*)

Argumentos de entrada:

fSep Variable entera en la que se pasa el código ASCII del carácter separador de campos. Se recomienda un valor mayor que 128.

lSep Variable entera en la que se pasa el el código ASCII del carácter separador de campos. Se recomienda un valor mayor que 128.

Argumentos de salida:

errC Variable de tipo entero en la que se recibe, al retorno de la función, un código indicativo del error ocurrido, caso de haberse producido alguno.

5.10.2. Ejemplos de uso de TJ2RDBSETSEP

5.10.2.1. Código FORTRAN que usa la función TJ2RDBSETSEP

```
PROGRAM testTJ2RDBGSSEP

integer   TJ2RDBGETSEP, TJ2RDBSETSEP
external  TJ2RDBGETSEP, TJ2RDBSETSEP

integer   fSep, lSep, errR, errC

errR = TJ2RDBGETSEP( fSep, lSep, errC)
if ( errC .eq. 0 ) then
  write(*,*)'Los separadores actuales son:'
  write(*,'(a,i)')' columnas : ', fSep
  write(*,'(a,i)')' lineas : ', lSep

  write(*,*)'Introducir los nuevos separadores'
  write(*,*)'separador de columnas'
  read(*,'(i)')fSep
  write(*,*)'lineas'
  read(*,'(i)')lSep
  errR = TJ2RDBSETSEP( fSep, lSep, errC)
  if ( errC .eq. 0 ) then
    fSep = 0
    lSep = 0
    errR = TJ2RDBGETSEP( fSep, lSep, errC)
    if ( errC .eq. 0 ) then
      write(*,*)'Los separadores actuales son:'
      write(*,'(a,i)')' separador de columnas : ', fSep
      write(*,'(a,i)')' separador de lineas : ', lSep
    else
      write(*,*)'Se produjo un error'
    end if
   endif

else
  write(*,*)'Se produjo un error en'

end if

STOP
```

```
END
```

5.10.2.2. Código C que usa la función TJ2RDBSETSEP

```
#include <stdio.h>
extern int TJ2RDBGETSEP( int * , int *, int *);
extern int TJ2RDBSETSEP( int * , int *, int *);


int main(){
      int          fSep, lSep, errR, errC;

      errR = 0;
      errR = TJ2RDBGETSEP ( &fSep, &lSep, &errC);
      if ( errR == 0 ){
            printf("Los separadores actuales son :\n\tpara columnas: %d\n\tpara
lineas: %d", fSep, lSep);
            fflush(stdout);
            printf("\n\nIntrocudir los nuevos separadores\npara columnas ");
            scanf("%d", &fSep);
            fflush(stdin);
            printf("\npara filas ");
            scanf("%d", &lSep);

            errR = TJ2RDBSETSEP( &fSep, &lSep, &errC);
            fSep = 0;
            lSep = 0;
            if ( errR == 0) {
                  errR = TJ2RDBGETSEP ( &fSep, &lSep, &errC);
                  if ( errR == 0 ){
                        printf("Los separadores actuales son :\n\tpara columnas:
%d\n\tpara lineas: %d", fSep, lSep);
                        fflush(stdout);
                  }
            }
      }
      else{
            printf("Se produjo un error");
            fflush(stdout);
      }



   printf("\n\n");fflush(stdout);
}
```

5.11. La función TJ2RDBGETSEP

La función **TJ2RDBSETSZ** permite conocer los caracteres usados como separadores de campos y de líneas en llamadas a las funciones TJ2RDBGI y TJ2RDBS.

La función **TJ2RDBSETSZ** devuelve un valor 0 si la ejecución se realiza con éxito o un valor diferente de 0 si se produce algún error durante su ejecución

Todos los argumentos han de ser pasados a la función por referencia, por compatibilidad con FORTRAN.

5.11.1. Descripción detallada de la función TJ2RDBGETSEP

int **TJ2RDBGETSZ** (int **fSep*,
int **lSep*,
int **errC*)

Argumentos de entrada:

fSep Variable entera en la que se recibe el código ASCII del carácter usado como separador de campos en llamadas a TJ2RDBGI o TJ2RDBS.

lSep Variable entera en la que se recibe el código ASCII del carácter separador de líneas en llamadas a las funciones TJ2RDBGI o TJ2RDBS.

Argumentos de salida:

errC Variable de tipo entero en la que se recibe, al retorno de la función, un código indicativo del error ocurrido, caso de haberse producido alguno.

5.11.2. Ejemplos de uso de TJ2RDBGETSEP

5.11.2.1. Código FORTRAN que usa la función TJ2RDBGETEP

Para este ejemplo se remite al lector a la sección 5.10.2.1. (Código FORTRAN que usa la función TJ2RDBSETSEP) donde se hace uso también de la función **TJ2RDBSETSEP**.

5.11.2.2. Código C que usa la función TJ2RDBGETSEP

Para este ejemplo se remite al lector a la sección 5.11.2.2. *(*Código C que usa la función TJ2RDBSETSEP) en la cual se muestra un ejemplo de uso de la función TJ2RDBSETSEP desde un programa C.

6. Apéndice 1. Bases de datos relacionales y consultas con SQL

En este apéndice se pretende tan sólo dar unas ideas básicas sobre la organización de información en bases de datos relacionales así como unas ideas básicas sobre la sintaxis de las sentencias SELECT de SQL para que los usuarios de esta librería puedan hacer consultas sencillas a la base de datos. Para una consulta detallada sobre este tema se sugiere visitar http://www.sql.org , donde se encontrarán multitud de referencias relacionadas con el tema.

6.1. Tablas

En una base de datos relacional los datos se almacenan en tablas. Se puede representar de forma aproximada una tabla de la base de datos de la siguiente forma:

Tabla **prueba**

Tabla 5 Ejemplo de tabla de una base de datos

Campo1	Campo2	Campo3	Campo4
1.2	Pepe	Esto es un texto largo	4
0.7	Juan	Esto es un texto corto	1
-3.7	Emilio	Un texto	45
0.0	Rosa		1234

Dentro de la tabla, las columnas se conocen como campos de la tabla. Los datos de una columna son siempre del mismo tipo, pudiendo diferentes columnas almacenar datos de diferente tipo.

En el ejemplo el **Campo1** es de tipo punto flotante, mientras que el **Campo2** es un texto y el **Campo4** es de tipo entero.

Normalmente en cada tabla de la base de datos existe un campo que es clave primaria, esto implica, entre otras cosas, que sus valores no se pueden repetir en diferentes filas de la tabla. Normalmente, la información en una base de datos relacional se almacena en diferentes tablas para disminuir el espacio de disco requerido para guardar la información y agilizar los accesos a la información.

Diferentes tablas dentro de la base de datos pueden estar relacionadas entre si. Las relaciones entre tablas se establecen a través de las claves primarias.

6.2. Sintaxis básica de las sentencias de consulta

Para extraer información de una base de datos relacional se usa el lenguaje SQL. Se envían de algún modo consultas escritas en este lenguaje al gestor de base de datos, quien interpreta la consulta y se encarga de buscar la información solicitada en la base de datos.

El lenguaje SQL permite tanto la consulta de información como la inserción de información nueva en la base de datos o la eliminación o modificación de información ya existente. Para la consulta de información se usan sentencias de tipo SELECT que permiten con gran flexibilidad seleccionar la información que queremos extraer de la base de datos.

La sintaxis básica de una sentencia SELECT es tal como esta:

SELECT *campo1[, campo2, ...]* **FROM** *tabla* [**WHERE** *condición*]

SELECT, FROM y **WHERE** son palabras reservadas de SQL.

campo1, campo2, ... son los nombres de los campos de la tabla *tabla* cuyo contenido queremos visualizar.

condición permite filtrar la información de acuerdo con algún criterio.

El siguiente ejemplo

```
SELECT campo1, campo3, campo4 FROM prueba WHERE campo4>5
```

nos devolvería

Tabla 6 Resultado de una consulta de ejemplo

Campo1	Campo3	Campo4
-3.7	Un texto	45
0.0		1234

El resultado de una consulta **SELECT** es siempre una tabla cuyos campos son los que se han seleccionado en la consulta y cuyas filas están determinadas tanto por el

contenido de la base de datos como por las condiciones impuestas en la sentencia de consulta.

Las condiciones en una sentencia **SELECT** pueden incluir los operadores relacionales para establecer condiciones sobre campos:

Tabla 7 Operadores relacionales en SQL

Operadores relacionales
=
!=
<
>
<=
>=

Y los siguientes operadores lógicos para concatenar condiciones

Tabla 8 Operadores lógicos en SQL

Operadores lógicos
AND
OR
NOT

Para establecer condiciones sobre cadenas de caracteres se puede usar el operador **LIKE**. El uso de **LIKE** es tal como en este ejemplo

SELECT campo3 FROM prueba WHERE campo3 LIKE'Esto%'

Esta sentencia nos devolvería

Tabla 9 Resultado de una consulta de ejemplo (II)

Campo3
Esto es un texto largo

Esto es un texto corto

El signo "%" representa cualquier número de caracteres cualesquiera.

Los resultados de la consulta se pueden ordenar usando la sentencia **ORDER BY**

Ejemplo

SELECT campo1, campo4 FROM prueba ORDER BY campo4

Nos dará

Tabla 10 Resultado de una consulta de ejemplo (III)

Campo1	**Campo4**
0.7	1
1.2	4
-3.7	45
0.0	1234

7. Referencias

[Bloomer92] J. Bloomer, Power Programming with RPC. O´Reilly & Associates, Inc.1992.

[Etzel99] M. Etzel and K. Dickinson, Digital Visual Fortran Programmer's Guide.1999

[RPC88] RPC: Remote Procedure Call Protocol Specification Version 2. DARPA RFC 1057. June 1988.

[Sanchez01] E. Sánchez, J. Vega, C. Crémy and A. Portas. Review of Scientific Instruments, 72(1) (2001) 525-529.

[Sanchez02] E. Sánchez, A. Portas and J. Vega.. Fus. Eng. & Design 60 (2002) 341-346.

[SQLRes] ver, por ejemplo, http://www.sql.org , donde se pueden encontrar referencias sobre el lenguaje SQL.

[Vega00] J. Vega, C. Crémy, E. Sánchez, A. Portas, J. A. Fábregas, R. Herrera. Fusion Engineering and Design, 47 (2000) 69-79.

[XDR87] XDR External Data Representation Specifications, Sun Microsystems, Inc. DARPA RFC 1014, June 1987.

8. Índice de materias