dbr.dkh

Funciones para administración de bases de datos relacionales

Versión 1

Contenido

[Conexiones 3](#_Toc78459477)

[Administrador de conexiones 3](#_Toc78459478)

[Nombre cualificado de conexión 3](#_Toc78459479)

[Funciones de administración de conexiones 3](#_Toc78459480)

[Estructuras de información de conexión 5](#_Toc78459481)

[MySQL 5](#_Toc78459482)

[SQL Server 6](#_Toc78459483)

[PostgreSQL 6](#_Toc78459484)

[Acceso a datos relacionales 7](#_Toc78459485)

[Acceso a datos relacionales con el modelo de Devkron 7](#_Toc78459486)

[Objetivos de diseño 7](#_Toc78459487)

[Confiabilidad 7](#_Toc78459488)

[Escalabilidad 8](#_Toc78459489)

[Adaptabilidad 8](#_Toc78459490)

[Patrón de diseño 8](#_Toc78459491)

[Campos de control 8](#_Toc78459492)

[Bloqueo pesimista 10](#_Toc78459493)

[Eliminación de entidades 12](#_Toc78459494)

[Funciones 12](#_Toc78459495)

La gestión de bases de datos relacionales en Devkron se realiza a través de bibliotecas especializadas. Esta biblioteca (dbr.dkh) se basa en induxsoft.API.DB que utiliza los mecanismos de conexión, patrones y prestaciones de induxsoft.Connections e induxsoft.DMA.

# Conexiones

## Administrador de conexiones

La información de conexión (credenciales y configuraciones) a múltiples orígenes de datos se gestiona (y almacena) a través del: Administrador de conexiones.

La organización de las conexiones se realiza con base en las aplicaciones que las utilizan.

Por lo tanto, el primer paso es definir una o más aplicaciones y para cada una de ellas las conexiones a bases de datos relacionales que sean necesarias.

Para obtener la mayor independencia de la plataforma subyacente a la base de datos, se emplean Adaptadores de acceso a datos.

Se incluyen los adaptadores para MySQL, MS SQL Server y PostgreSQL.

Cada adaptador requiere los conectores ADO.Net provistos por el fabricante.

### Nombre cualificado de conexión

Las conexiones pueden utilizarse mediante el nombre cualificado que está formado por el nombre de la aplicación y la conexión siguiendo la siguiente convención: conexión@aplicación.

También es posible indicar una conexión que está alojada en una base de datos con la sintaxis: conexión\_almacen\_conexiones@app\_almacen\_conexiones/conexión@aplicación.

Si se omite el nombre de la conexión se utilizará la predeterminada.

Ejemplos:

* miconexion\_empresa1@sistemaempresarial
* miconexion\_empresa2@sistemaempresarial
* @sistemaempresarial
* conexiones@miempresa/conexion1587@sistema\_empresa

Las conexiones se alojan de manera predeterminada en el archivo %ProgramData%\induxsoft\machine\winshell\v10\connections.xml

Use el Administrador de conexiones provisto con Devkron para gestionar esta información a través de una interfaz de usuario o las funciones de esta biblioteca.

## Funciones de administración de conexiones

|  |  |
| --- | --- |
| dba.newconnection  Crea una nueva conexión a una base de datos existente con la información indicada  dba.newconnection(qn, info) | qn – Cadena con el nombre cualificado de la nueva conexión  info – Referencia a un registro que contiene todos los datos necesarios para establecer la conexión (host, puerto, usuario, password, etc.) |
| dba.createconnection  Crea una nueva conexión a una base de datos existente con la información indicada  dba.createconnection(qn, info, default, sqldic) | qn – Cadena con el nombre cualificado de la nueva conexión  info – Referencia a un registro que contiene todos los datos necesarios para establecer la conexión (host, puerto, usuario, password, etc.)  default – Número (booleano) que indica si se establecerá la conexión como la predeterminada para el grupo de conexión de la aplicación indicada.  sqldic – Cadena con la ruta a un archivo de diccionario de consultas sql |
| dba.createdb  Crea una nueva base de datos y agrega la información de conexión al administrador de conexiones.  dba.createdb(qn, info) | qn – Cadena con el nombre cualificado de la nueva conexión  info – Referencia a un registro que contiene todos los datos necesarios para establecer la conexión (host, puerto, usuario, password, etc.) |
| dba.createschema  Crea una nueva base de datos y agrega la información de conexión al administrador de conexiones.  dba.createschema(qn, info, default,sqldic) | qn – Cadena con el nombre cualificado de la nueva conexión  info – Referencia a un registro que contiene todos los datos necesarios para establecer la conexión (host, puerto, usuario, password, etc.)  default – Número (booleano) que indica si se establecerá la conexión como la predeterminada para el grupo de conexión de la aplicación indicada.  sqldic – Cadena con la ruta a un archivo de diccionario de consultas sql |
| dba.getconnectionset  Devuelve una referencia a un almacén de conexiones en una base de datos.  dba.getconnectionset(qn) | qn-Cadena con el nombre cualificado de una conexión a una base de datos que contiene un almacén de conexiones. |
| dba.getconnection  Devuelve una referencia a un registro con la información de conexión del nombre cualificado indicado. | qn-Cadena con el nombre cualificado de una conexión existente. |
| dba.delconnection  Elimina la conexión indicada  dba.getconnection(qn) | qn-Cadena con el nombre cualificado de una conexión existente. |
| dba.cnnexists  Devuelve @true si la conexión indicada existe  dba.cnnexists(qn) | qn-Cadena con el nombre cualificado de una conexión. |
| dba.addapp  Agrega un nombre de aplicación al almacén de conexiones  dba.addapp(app) | app – Cadena con el nombre de aplicación a agregar |
| dba.delapp  Elimina un nombre de aplicación del repositorio de conexiones  dba.delapp(app) | app – Cadena con el nombre de aplicación a eliminar |
| dba.apps  Devuelve una referencia a una lista de cadenas que contienen todos los nombres de aplicación del repositorio de conexiones.  dba.apps(qn) | qn-Cadena que puede ser un nombre cualificado de conexión si el repositorio de conexiones está en una base de datos o una cadena vacía para referirse al repositorio local de conexiones. |
| dba.cnns  Devuelve una referencia a una lista de cadenas que contienen todos los nombres de conexión del repositorio de conexiones que corresponden a la aplicación indicada.  dba.cnns(qn) | qn-Cadena con el nombre de una aplicación para listas las conexiones del repositorio local o una cadena que incluye un nombre cualificado de conexión a una base de datos que contiene las conexiónes. |

## Estructuras de información de conexión

### MySQL

using cnnInfo

        {

            @"Engine":"MY\_SQL"

            @"Host":"Nombre o dirección IP"

            @"UserName":"Usuario"

            @"Password":"Contraseña"

            @"Port":3306

            @"DefaultSchema":"Nombre de la base de datos"

        }

### SQL Server

using cnnInfo

    {

        @"Engine":"SQL\_SERVER"

        @"Host":"Nombre o dirección IP"

        @"UserName":"Usuario"

        @"Password":"Contraseña"

        @"DefaultSchema":"Nombre de la base de datos"

        @"WindowsAuthMode":@true //@true or @false

    }

### PostgreSQL

using cnnInfo

    {

        @"Engine":"POSTGRE\_SQL"

        @"Host":"Nombre o dirección IP"

        @"UserName":"Usuario"

        @"Password":"Contraseña"

        @"Port":5432

        @"DefaultSchema":"Nombre de la base de datos"

        @"NameSpaceSchema":"Nombre del espacio de nombres del esquema"

    }

# Acceso a datos relacionales

|  |  |
| --- | --- |
| dbr.open  Devuelve una referencia hacia una conexión de bases de datos abierta  dbr.open(qn) | qn – Cadena con nombre cualificado de una conexión existente |
| dbr.close  Cierra una conexión  dbr.close(db) | db – Referencia a una conexión de bases de datos |
| dbr.begin  Inicia una transacción de bases de datos  dbr.begin(db) | db – Referencia a una conexión de bases de datos |
| dbr.rollback  Aborta una transacción de bases de datos  dbr.rollback(db) | db – Referencia a una conexión de bases de datos |
| dbr.commit  Confirma una transacción de bases de datos  dbr.commit(db) | db – Referencia a una conexión de bases de datos |
| dbr.in\_transaction  Devuelve @true si se está actualmente dentro de una transacción de bases de datos  dbr.in\_transaction(db) | db – Referencia a una conexión de bases de datos |
| dbr.is\_open  Devuelve @true si la referencia a una conexión de bases de datos está abierta.  dbr.is\_open(db) | db – Referencia a una conexión de bases de datos |
| dbr.full\_qname  Devuelve una cadena con el nombre cualificado completo de la referencia a una conexión de bases de datos.  dbr.full\_qname(db) | db – Referencia a una conexión de bases de datos |
| dbr.relative\_qname  Devuelve una cadena con el nombre cualificado relativo a otra conexión de la referencia a una conexión de bases de datos.  dbr.relative\_qname(db) | db – Referencia a una conexión de bases de datos |
| dbr.base\_qname  Devuelve una cadena con el nombre cualificado de base de la referencia a una conexión de bases de datos.  dbr.base\_qname(db) | db – Referencia a una conexión de bases de datos |
| dbr.sql  Devuelve una referencia a un objeto que representa sintaxis SQL nativa.  dbr.sql(expresión) | expresión – Cadena con una expresión SQL |
| dbr.execute  Ejecuta una consulta SQL  dbr.execute(db, sql, params) | db – Referencia a una conexión de bases de datos  sql – Cadena con una consulta SQL DML o DDL  params – Referencia a un registro con los parámetros que se pasarán a la consulta |
| dbr.table  Devuelve una referencia a una tabla que contiene el resultado de una consulta SQL  dbr.table(db, sql, params) | db – Referencia a una conexión de bases de datos  sql – Cadena con una consulta SQL SELECT  params – Referencia a un registro con los parámetros que se pasarán a la consulta |
| dbr.str  Devuelve como una cadena la primera columna y primera fila del resultado de una consulta SQL.  dbr.str(db, sql, params) | db – Referencia a una conexión de bases de datos  sql – Cadena con una consulta SQL SELECT  params – Referencia a un registro con los parámetros que se pasarán a la consulta |
| dbr.num  Devuelve como un número la primera columna y primera fila del resultado de una consulta SQL.  dbr.num(db, sql, params) | db – Referencia a una conexión de bases de datos  sql – Cadena con una consulta SQL SELECT  params – Referencia a un registro con los parámetros que se pasarán a la consulta |
| dbr.scalar  Devuelve como una referencia al valor de la primera columna y primera fila del resultado de una consulta SQL.  dbr.scalar(db, sql, params) | db – Referencia a una conexión de bases de datos  sql – Cadena con una consulta SQL SELECT  params – Referencia a un registro con los parámetros que se pasarán a la consulta |
| dbr.rec  Devuelve como una referencia a un registro que contiene todas las columnas (con sus valores) de la primera fila del resultado de una consulta SQL.  dbr.rec(db, sql, params) | db – Referencia a una conexión de bases de datos  sql – Cadena con una consulta SQL SELECT  params – Referencia a un registro con los parámetros que se pasarán a la consulta |
| dbr.update  Ejecuta la actualización de filas de una tabla.  dbr.update(db, tabla, registro, expresión, params) | db – Referencia a una conexión de bases de datos  tabla – Cadena con el nombre de la tabla cuyas filas se actualizarán  registro – Referencia a un registro que contiene los datos a asignar a los campos  expresión – Cadena de filtro (SQL) válido para identificar las filas a actualizar  params – Referencia a un registro con los parámetros que se pasarán a la expresión de filtro |
| dbr.insert  Ejecuta la inserción de una fila en una tabla.  dbr.insert(db, tabla, registro) | db – Referencia a una conexión de bases de datos  tabla – Cadena con el nombre de la tabla en la cual se insertará  registro – Referencia a un registro con los datos de la nueva fila |
| dbr.delete  Eliminar una fila de una tabla.  dbr.delete(db, tabla, expresión, params) | db – Referencia a una conexión de bases de datos  tabla – Cadena con el nombre de la tabla  expresión – Cadena de filtro (SQL) válido para identificar las filas a eliminar  params – Referencia a un registro con los parámetros que se pasarán a la expresión de filtro |

# Acceso a datos relacionales con el modelo de Devkron

El modelo de datos de una aplicación es una abstracción del conjunto de elementos del mundo real que le son relevantes.

Ésta información se preservará en una base de datos relacional subyacente y será manipulada a través de servicios.

## Objetivos de diseño

### Confiabilidad

Información confiable que pueda usarse para tomar decisiones es una prioridad y objetivo de diseño, para alcanzarla es necesario que se prevengan la mayor cantidad de posibles situaciones de riesgo a la veracidad y precisión de los datos.

La utilización correcta de claves primarias, foráneas, restricciones y el uso de transacciones son obligatorias y fundamentales.

En el mismo sentido, manejar la concurrencia es vital para asegurar que los datos no sean sobrescritos inadvertidamente.

### Escalabilidad

La posibilidad de migrar fácilmente los datos a plataformas más robustas con nulos o mínimos cambios al código fuente es un objetivo clave.

Las tareas de respaldo, integración y replicación de datos deben ser consideradas desde el diseño de estructuras y la implementación de su gestión desde el código fuente.

### Adaptabilidad

Es importante permitir el crecimiento y evolución de los sistemas con el mínimo esfuerzo en cuanto a la gestión del acceso a datos.

También es necesario adoptar prácticas adecuadas para el control de los cambios y la documentación de los mismos.

1. Facilitar el aprendizaje de las técnicas de manejo de datos a programadores con poca experiencia utilizando una misma API para diferentes plataformas y lenguajes de desarrollo.
2. Limitar la intervención humana en la gestión del acceso a datos y disminuir la necesidad de que tomen previsiones a la hora de implementar aplicaciones y permitirles centrarse en la funcionalidad.

## Patrón de diseño

Para conseguir los objetivos de diseño hemos previsto la utilización de un patrón de diseño en la estructura relacional con las siguientes características:

1. Un conjunto de elementos estructurales de control con nombres y propósitos específicos que faciliten la adopción de mejores diseños y su interpretación, así como el trabajo de manipulación de los datos.
2. Prever problemas de concurrencia desde la capa más baja de los programas de aplicación mediante técnicas desarrolladas en base a la experiencia propia y de mejores prácticas de la industria.

### Campos de control

|  |  |  |
| --- | --- | --- |
| **Campo** | **Tipo** | **Descripción** |
| sys\_pk | Entero | Clave primaria para la tabla (alcance local) |
| sys\_guid | Texto(32) | GUID que puede usarse como clave primaria de alcance global |
| sys\_dtcreated | Fecha/Hora | Fecha y hora de creación del registro |
| sys\_timestamp | Fecha/Hora | Fecha y hora de la última actualización del registro (marca de tiempo con precisión hasta segundos) |
| sys\_exported | Cierto/falso | Indica si el registro ha sido exportado por las utilidades de transporte de datos |
| sys\_dtexported | Fecha/Hora | Indica la fecha en que se exportó por última vez |
| sys\_recver | Entero | Número de versión de instancia de registro |
| sys\_lock | Entero | Clave foránea que indica si se ha establecido un bloqueo pesimista sobre la fila.  Este campo admite valores Nulos que significan que el registro no está bloqueado.  No se admiten duplicados |
| sys\_deleted | bool | Indica si la fila está marcada como eliminada |

#### Clave primaria

Siempre tiene el nombre **sys\_pk** y se trata de un entero (de 32 bits o 64 bits dependiendo la plataforma y la configuración para el tipo SQL INT). Tiene los atributos incremental, único y sin permisión de valores nulos. Indexado ascendentemente.

#### Claves foráneas

Todas las claves foráneas son de tipo Entero (SQL INT) y contienen el valor del campo **sys\_pk** de la tabla a la que apuntan.

#### GUID’s

¿Qué es una GUID?

Es la abreviatura de Global Unique Identifier (Identificador Global Único) y constituye un valor numérico extremadamente alto (32 dígitos hexadecimales) generado por un algoritmo computacional que prácticamente asegura que es irrepetible independientemente del lugar y momento en que se cree.

¿Para qué una GUID en cada registro?

Las claves primarias usadas en todas las tablas son un entero auto-incremental generado por el motor de la base de datos, esto garantiza el mejor rendimiento del sistema y obedece a las mejores prácticas de diseño. Este valor es irrepetible en la misma tabla y base de datos.

No obstante, un cliente “X” en la tabla “Cliente” de la base de datos “A” puede tener la misma clave primaria (valor para Sys\_PK) que un cliente “Y” en la tabla “Cliente” pero en la base de datos “B”. Esta situación no reviste ningún problema salvo si quisiéramos comprobar que el cliente “X” de la base de datos “A” no existe también en la base de datos “B”. (O si quisiéramos copiarlo).

Es decir, cada registro solo es único en la tabla y base de datos que lo contiene (alcance local de la clave primaria).

Aunque la mayoría de las entidades por su naturaleza implementan códigos alfanuméricos (el campo Código de la tabla Cliente por ejemplo), no existe la garantía de que en otras bases de datos no se repitan.

Esto se soluciona con la implementación de GUIDs que son prácticamente únicas universalmente.

¿Por qué una GUID de 32 caracteres?

Algunos motores de bases de datos implementan el tipo GUID (o equivalente), que está optimizado para contener este tipo de valores, sin embargo utilizarlo cuando esté disponible supondría una dependencia en particular.

Por esta razón se implementa el campo **sys\_guid** como texto (generalmente Varchar(32)).

#### Versión de instancia de registro

El campo **sys\_recver** establece un control de concurrencia mediante bloqueo optimista, contiene un número entero incremental y está implementado generalmente como un INT con el propósito de resolver el problema de sobre-escritura inadvertida en un entorno multiusuarios.

El campo **sys\_recver** identifica una instancia de un registro en particular en el tiempo.

Así el patrón de la instrucción UPDATE se implementa como:

**UPDATE tabla SET sys\_recver=sys\_recver+1,… WHERE (sys\_pk=X AND sys\_recver=Z)**

Donde X es la clave primaria del registro y Z es el número de versión que tenía el registro en el momento en que se leyó. Obsérvese que se incrementa el contador de versión con cada instrucción UPDATE.

Aunque es teóricamente posible exceder los límites del tipo INT con sucesivos incrementos, es poco probable que ocurra en la práctica y además, puede implementarse una rutina de re-inicialización en función del escenario previsto.

### Bloqueo pesimista

Consiste en impedir la modificación de un registro durante todo el tiempo en que se encuentre impuesto un bloqueo por otro usuario.

El bloqueo pesimista se implementa por fila. Para este propósito se utiliza el campo **sys\_lock** implementado como clave foránea (tipo INT) que admite valores nulos pero no duplicados.

El mecanismo se apoya en una tabla adicional que contiene la información de estado del bloqueo, dicha tabla con el nombre **sys\_lockinfo** deberá tener los siguientes campos:

* **sys\_table**. Clave foránea de tipo entero que apunta hacia un catálogo de tablas para identificar la tabla a la que aplica el bloqueo.
* **sys\_row**. Entero que contiene el valor de la sys\_pk de la fila bloqueada sin establecerse como clave foránea para evitar problemas por restricciones.
* **sys\_token**. Clave foránea que contiene el valor de sys\_pk de la tabla que controla las sesiones.
* **sys\_active**. Un valor boolean que indica si el bloqueo está activo o no.

De lo anterior se desprende la necesidad de que se implementen al menos las siguientes tablas de sistema:

* **sys\_catalog**. Que contendrá un catálogo de las entidades (tablas que se ajustan al patrón de diseño).
* **sys\_session**. Que contendrá un registro de control de sesión.

Los procesos de bloqueo se implementan como sigue:

INT function Solicitud\_de\_bloqueo(tabla, fila, sesión)

Si el valor del campo sys\_lock es nulo

Insertar un registro de bloqueo en sys\_lockinfo

Establecer el valor de sys\_lock como la clave primaria del registro de bloqueo

Return sys\_lockinfo.sys\_pk //El bloqueo se ha establecido correctamente

Si no

Verificar si el registro de bloqueo en sys\_lockinfo ha excedido el timeout

Si timeout está excedido

Cambiar valor de sys\_lockinfo.sys\_active=false

Insertar un nuevo registro de bloqueo en sys\_lockinfo

Establecer el valor de sys\_lock como la clave primaria del registro de bloqueo

Return sys\_lockinfo.sys\_pk //El bloqueo se ha establecido correctamente

Si no

Return 0 //El registro está bloqueado por otro usuario

Bool function Quitar\_Bloqueo(IDBloqueo)

Cambiar valor de sys\_lockinfo.sys\_active=false del registro de bloqueo que corresponde

Poner en nulo el valor del campo sys\_lock de la tabla cuyo registro estaba bloqueado.

Observe que existe la necesidad de un valor de timeout para el máximo tiempo permitido para que un registro permanezca bloqueado.

Por otro lado, la implementación de la actualización en la base de datos desde un objeto persistente deberá considerar además la implementación de bloqueo pesimista aquí descrita.

### Eliminación de entidades

Eliminar además de peligroso puede ser complejo y engorroso, por eso se implementan los siguientes conceptos:

* **Eliminación lógica:** Consiste en marcar una fila como eliminada para que no sea tomada en cuenta por las consultas y simule no existir aunque sigue estando en la base de datos y permite conservar la integridad y datos relacionados. Esto se consigue usando el campo **sys\_deleted** para establecer y consultar la marca.
* **Eliminación definitiva:** Esta implica el borrado de la información de la base de datos (sentencia DELETE) y puede ser limitada a los registros de una tabla o en cascada, lo cual supone eliminar registros relacionados a un padre o principal.

## Funciones

Además de las funciones para el acceso a datos relacionales, las bases de datos y tablas que sigan modelo de datos de Devkron pueden usar las siguientes funciones:

|  |  |
| --- | --- |
| dbr.login  Devuelve una referencia hacia una conexión de bases de datos abierta y crea un registro de sesión si el usuario y contraseña son válidos.  dbr.login(qn, usuario, pwd) | qn – Cadena con el nombre cualificado de una conexión  usuario – Cadena con el Id de usuario  pwd – Cadena con la contraseña |
| dbr.logout  Cierra la sesión y la conexión a la base de datos.  dbr.logout(db) | db – Referencia a una conexión de bases de datos |
| dbr.reconnect  Devuelve una referencia hacia una conexión de bases de datos abierta si el id de sesión presentado es válido.  dbr.reconnect(qn, sesid) | qn – Cadena con el nombre cualificado de una conexión  sesid – Cadena con el Id de una sesión |
| dbr.setHashUnicode  Establece que la comprobación de la firma md5 se realice considerando la cadena de contraseña codificada con UNICODE.  dbr.setHashUnicode(v) | v – Número (booleano) que establece el tipo de codificación para la firma de comprobación de la contraseña |
| dbr.getSID  Devuelve una cadena con el Id de la sesión  dbr.getSID(db) | db – Referencia a una conexión de bases de datos |
| dbr.get  Devuelve una referencia a un registro con los datos de fila de una tabla con base en su clave primaria (sys\_pk)  dbr.get(db, tabla, pk) | db – Referencia a una conexión de bases de datos  tabla – Cadena con el nombre de la tabla  pk – Número (entero) que corresponde a la clave primaria de la fila a recuperar |
| dbr.getg  Devuelve una referencia a un registro con los datos de fila de una tabla con base en su GUID (sys\_guid)  dbr.getg(db, tabla, guid) | db – Referencia a una conexión de bases de datos  tabla – Cadena con el nombre de la tabla  guid – Cadena que corresponde a la GUID de la fila a recuperar |
| dbr.getw  Devuelve una referencia a un registro con los datos de fila de una tabla con base en una expresión.  dbr.getw(db, tabla, expresión, params) | db – Referencia a una conexión de bases de datos  tabla – Cadena con el nombre de la tabla  expresión – Cadena de expresión de filtro (SQL)  params – Referencia a un registro con los parámetros para la expresión de filtro |
| dbr.save  Actualiza o inserta una fila en una tabla de la base de datos con base en su sys\_pk considerando la versión de instancia de registro (sys\_recver)  dbr.save(db, tabla, registro) | db – Referencia a una conexión de bases de datos  tabla – Cadena con el nombre de la tabla  registro - Referencia a un registro que contiene los datos |
| dbr.chklock  Devuelve @true si el Id de bloqueo presentado es válido.  dbr.chklock(db, tabla, id) | db – Referencia a una conexión de bases de datos  tabla – Cadena con el nombre de la tabla  id – Número que corresponde a un Id de bloqueo |
| dbr.lock  Impone un bloqueo pesimista sobre la fila de la tabla indicada y devuelve el Id de bloqueo.  dbr.lock(db, tabla, pk) | db – Referencia a una conexión de bases de datos  tabla – Cadena con el nombre de la tabla  pk – Número que corresponde con la clave primaria (sys\_pk) de la fila a bloquear |
| dbr.unlock  Retira el bloqueo pesimista especificado por su id.  dbr.unlock(db, tabla, id) | db – Referencia a una conexión de bases de datos  tabla – Cadena con el nombre de la tabla  id – Número que corresponde a un Id de bloqueo |
| dbr.erase  Realiza la eliminación “lógica” de una fila de una tabla.  dbr.erase(db, tabla, pk) | db – Referencia a una conexión de bases de datos  tabla – Cadena con el nombre de la tabla  pk – Número que corresponde con la clave primaria (sys\_pk) de la fila a borrar |