## Léame de Microsoft Repository 2.0

(c) 1998 Microsoft Corporation. Reservados todos los derechos.

El Léame de Repository incluye información actualizada acerca de la documentación que se proporciona con el sistema de desarrollo para Windows e Internet Microsoft(r) Visual Studio. La información de este documento está más actualizada que la información del sistema de Ayuda. Muchos de los problemas que se tratan en este documento se corregirán en versiones posteriores.

Para *cuestiones de información general* sobre el conjunto de productos Visual Studio 6.0, incluida la instalación conjunta de productos, consulte el archivo Léame de [Notas de instalación](file:///C:\Users\jcond\Downloads\Microsoft%20Visual%20Basic%206.0\install.htm) (install.htm).

Para las demás cuestiones sobre el sistema de Ayuda del conjunto de productos Visual Studio, consulte el [archivo Léame de MSDN™ Microsoft Developer Network.](file:///C:\Users\jcond\Downloads\Microsoft%20Visual%20Basic%206.0\readmeDN.htm)

### Contenido

##### Introducción a Repository

[¿Qué es Microsoft Repository?](" \l "Topic_10)

**NUEVO** [Lo nuevo en Microsoft Repository 2.0](#Topic_11)

[Información importante para los clientes de la versión 2.0](#Topic_12)

##### Instalación

**Importante:**   Esta sección contiene información vital acerca de la instalación de Repository 2.0. Lea atentamente todos los apartados antes de comenzar la instalación.

[Notas de instalación de Repository](" \l "Topic_20)

[Archivos de Repository en el disco duro](#Topic_21)

##### Características de Repository 2.0

[Diferencias entre Repository 1.0 y Repository 2.0](" \l "Topic_30)

[Funcionamiento asíncrono](" \l "Topic_31)

[Tipos de SQL y API utilizados en las definiciones de propiedades](" \l "Topic_32)

[Características nuevas](" \l "Topic_33)

##### Limitaciones y más información

[Limitaciones conocidas de Microsoft Repository 2.0](" \l "Topic_40)

[Para obtener más información](#Topic_41)

### Introducción a Repository

#### ¿Qué es Microsoft Repository?

Microsoft Repository es una tecnología que permite definir y llenar modelos de información. La utilizan las herramientas de desarrollo de ingeniería y software para compartir información sobre artefactos de ingeniería. Los artefactos de ingeniería pueden ser componentes de software, componentes industriales, documentos, mapas, páginas Web (cualquier información con una estructura compleja que se desee compartir). Microsoft Repository almacena los datos de depósito en una base de datos relacional. Es compatible con dos sistemas de administración de bases de datos: Microsoft*(r)* SQL Server versión 6.5 o posterior y el motor de bases de datos Microsoft Jet versión 3.0 o posterior.

#### Lo nuevo en Microsoft Repository 2.0

Microsoft Repository 2.0 contiene mejoras significativas en las áreas de administración de versiones y desarrollo en equipo, mientras que se mantiene la compatibilidad con las anteriores interfaces de Microsoft Repository 1.0.

Las principales áreas de nueva funcionalidad son:

* **Versiones de objetos**: permiten al usuario del depósito mantener varias versiones de los objetos del depósito.
* **Versiones de relaciones**: permiten al usuario del depósito mantener varias versiones de las relaciones entre objetos.
* **Administración del área de trabajo**: proporciona un depósito virtual para usuarios en el que pueden tratar versiones privadas de objetos y relaciones. Un modelo de Protección/Desprotección permite a los usuarios controlar la información que es visible en sus propias áreas de trabajo y qué cambios se hacen visibles para los demás usuarios. Esta nueva funcionalidad se describe detalladamente en la documentación del usuario y en las especificaciones que acompañan al software.

#### Información importante para los clientes de la versión 2.0

Los usuarios deberían tener en cuenta lo siguiente:

* Microsoft Repository 2.0 es compatible con las anteriores interfaces y funcionalidad de la versión 1.0. Si todas las aplicaciones que usan una base de datos de depósito sólo utilizan la funcionalidad de la versión 1.0, no aparecerán incompatibilidades. De igual manera, si todas las aplicaciones que utilizan una base de datos de depósito están preparadas para ambas versiones y áreas de trabajo, deberían funcionar bien. No es recomendable la combinación de aplicaciones preparadas para ambas versiones con aplicaciones que no están preparadas para versiones y áreas de trabajo en la misma base de datos del depósito.
* El esquema SQL de Microsoft Repository 2.0 es una extensión del esquema de Microsoft Repository 1.0 debido a la introducción de las versiones. Puede que las aplicaciones que pasan por alto las interfaces del depósito, al tener acceso directamente a tablas SQL tengan que modificar consultas o proporcionar vistas para mantener la compatibilidad con lo anterior.
* El asistente para migración (MigRepV2.exe) convierte la información de la base de datos subyacente de Microsoft Repository de la versión 1.0 a la versión 2.0 (de SQL a SQL, y de Jet a Jet). Debe tener registrado el motor de Microsoft Repository versión 2.0 (motor V2) antes de ejecutar el asistente para migración. Tenga en cuenta que la migración es sólo en un sentido, de la versión 1.0 a la versión 2.0.
* Microsoft Repository 2.0 no es compatible con el modo WRITETHROUGH de ejecución de transacciones.
* Microsoft Repository 2.0 es compatible con plataformas Intel x86 y Alpha.

### Instalación

#### Notas de instalación de Repository

Microsoft Repository se instala durante una instalación de Visual Basic.

Microsoft Repository necesita que se instale el componente de acceso a datos. De manera predeterminada, el componente Data Access se instala durante una instalación de Visual Basic. Sin embargo, si elige no instalar el componente de acceso a datos y, después, instala una herramienta de terceros que utiliza Microsoft Repository, se producirá el error siguiente al iniciar Visual Basic 98:

"Error al abrir la base de datos de Microsoft Repository. El controlador especificado no se ha podido cargar debido a un error 126 del sistema (controlador de Microsoft Access (\*.MDB)). El complemento de Microsoft Repository para Visual Basic se cierra."

Para corregir este problema, instale el componente Data Access.

#### Archivos de Repository en el disco duro

Los archivos siguientes se agregarán al directorio SYSTEM (o SYSTEM32) de Windows:

* REPUTIL.DLL - Herramientas para el motor de Repository

Los siguientes archivos se agregarán al directorio "Archivos de programa\Archivos comunes\Microsoft Shared\Repostry":

* REPODBC.DLL - El motor del depósito
* REPRC.DLL - Recursos del motor del depósito
* REPBROWS.EXE - Un explorador básico del depósito
* REPBRRC.DLL - Recursos del explorador del depósito
* MIGREPV2.EXE - Interfaz de usuario del Asistente para migración
* MIGV2RC.DLL - Recursos del Asistente para migración
* MIGV2.DLL - Algoritmo del Asistente para migración
* REPCDLG.DLL - Cuadro de diálogo común de Repository
* REPCDLG.OCX - Controles ActiveX para el cuadro de diálogo común de Repository
* REPCDRC.DLL - Recursos del cuadro de diálogo común de Repository

Los siguientes archivos se agregarán al directorio "Archivos de programa\Archivos comunes\Microsoft Shared\Repostry\infoMdl":

* REPCDE.DLL - Modelo de información de descripción de componentes
* REPCOM.DLL - Modelo de información COM
* REPDTM.DLL - Modelo de información de tipos de datos
* REPGEN.DLL - Modelo de información genérico
* UML.DLL - Modelo de información UML (Lenguaje de modelo unificado)
* REPUMX.DLL - Modelo de información de extensión de UML
* REPVCM.DLL - Modelo administrador de componentes visuales

Los archivos siguientes se agregarán al directorio de Visual Basic:

* REPVB.DLL - Complemento Repository para Visual Basic
* REPVBRC.DLL - Archivo de recursos para el complemento Repository
* REPVBTIM.DLL - Definiciones de tipo para el modelo MDO

### 

### Características de Repository 2.0

#### Diferencias entre Repository 1.0 y Repository 2.0

##### Uso de Repository 1.0

Si la base de datos del depósito se ha creado mediante el motor de Repository 1.0 y se ha utilizado para la creación, el llenado y la recuperación del modelo de información sólo mediante este motor, habrá algunas diferencias cuando ejecute la misma aplicación con el motor de Repository 2.0. Las diferencias surgen del tratamiento de las relaciones, como se enumera a continuación:

* En Repository 1.0, si una aplicación inserta una relación de nombre único con un nombre ya existente, la infracción de la unicidad de nombres se detecta de manera inmediata. En Repository 2.0, la transgresión sólo se detecta de manera inmediata si la relación ya existente está en la caché (por ejemplo, creada como parte de la misma transacción o ya recuperada en la memoria caché). En cualquier otro caso, la validación de unicidad de nombre ocurre en el momento de la confirmación.
* Los enumeradores del motor de Repository 1.0, excepto los de las colecciones de relaciones, tienen contenido estático, por lo que los elementos agregados no aparecen hasta que se solicita un nuevo enumerador a la colección. En el motor de Repository 2.0, los enumeradores son dinámicos, por lo que los elementos agregados aparecen inmediatamente en el enumerador.
* En Repository 1.0, si la posición de inserción en **IRelationshipCol::Insert()** es mayor que la cuenta actual de la colección, se trata la llamada como **IRelationshipCol::Add()**. En Repository 2.0, si el punto de inserción especificado es una unidad mayor que la cuenta actual, la llamada se reduce a **IRelationshipCol::Add()**. Para posiciones mayores de inserción, la llamada devuelve un error puesto que el usuario probablemente se equivocó (sería un error de programación en cualquier caso).
* La invocación a **IRelationshipCol::Insert()** en una colección no ordenada o de destino produce un error en Repository 2.0. En Repository 1.0, la llamada realiza **IRelationshipCol::Add()** en el origen si la colección está ordenada.

**Utilización en modo mixto de Repository 1.0 y Repository 2.0**

Es concebible que una aplicación de Repository 1.0 se pueda ejecutar en el motor de Repository 2.0, en la que las operaciones de versiones se realizan mediante otras aplicaciones sobre el mismo depósito. El motor del curso tiene que respetar la semántica de las versiones, y algunas reglas semánticas serán necesariamente visibles para la aplicación que no tiene control de versiones. Una aplicación de Repository 1.0 solamente puede ver estos efectos al funcionar sobre una base de datos de depósito en la cual las aplicaciones de Repository 2.0 realizan operaciones de versiones y áreas de trabajo. Estos efectos no surgen cuando todas las aplicaciones que actúan sobre la base de datos del depósito siguen la semántica de Repository 1.0. Además de los efectos mencionados anteriormente, pueden surgir los siguientes:

* Una aplicación de Repository 1.0 puede recuperar un objeto mediante **get\_Object()** (obtiene una versión resuelta), eliminarlo (la versión se elimina) y utilizar **get\_Object()** una segunda vez (la llamada se resuelve con una versión diferente).
* Puesto que todos los objetos están desbloqueados en Repository 1.0, las actualizaciones nunca fallan en Repository 1.0. Sin embargo, en Repository 2.0, todas las operaciones de actualización (actualizaciones de propiedades y de la colección origen) sobre una versión bloqueada o desprotegida producirán un error.
* En Repository 2.0, una operación de eliminación sobre una versión desprotegida o sobre una versión con un sucesor producirá un error. Ninguna de las situaciones se podían dar en Repository 1.0.
* En Repository 1.0, **IRelationship::Delete()** elimina la relación completamente por lo que la relación deja de existir en los objetos de origen y de destino. En Repository 2.0, si hemos explorado desde el destino, la relación desde la versión de origen a las demás versiones de destino sigue existiendo tras la eliminación. Si hemos explorado desde el origen, se elimina la relación desde esa versión a todas las versiones de destino. El mismo comportamiento se da para **IRelationshipCol::Remove()** y **ITargetObjectCol::Remove()**.
* En Repository 1.0, la propagación de eliminaciones detiene en un objeto si es el destino de alguna otra relación del mismo tipo. En Repository 2.0, además de esta condición, la propagación de eliminación se detiene en una versión no modificable (bloqueada o desprotegida en un área de trabajo diferente) y en las versiones que tienen relaciones entrantes (de cualquier tipo) desde un origen bloqueado (es decir, sin error).

**Tablas SQL**

* Las tablas SQL de Repository 2.0 están revisadas de manera exhaustiva para el control de versiones. Las vistas SQL que coinciden con consultas de SQL de Repository 1.0 que se ejecutan sobre una base de datos del depósito en la cual todos los objetos tienen una versión deben funcionar sin ningún cambio.
* Es probable que las consultas SQL de Repository 1.0 ejecutadas sobre un depósito en el que hay varias versiones de objetos se ejecuten hasta el final, pero su semántica se debería revisar a la luz del control de versiones. En ciertas situaciones, las consultas pueden devolver varias versiones del mismo objeto; una aplicación de Repository 1.0 no espera esto.
* La tabla **RTblSites** no tiene la columna **NextLocalID** en Repository 2.0.
* La tabla **RTblClassDefs** tiene una nueva columna llamada **VerPropDescs** donde se almacena la definición de clase en Repository 2.0. La columna **PropDescs** sólo contiene valores nulos y ya no se utiliza. Sin embargo, se mantiene por compatibilidad.

**Varios**

La tabla siguiente muestra otras diferencias entre Repository 1.0 y Repository 2.0.

|  |  |  |
| --- | --- | --- |
| **Descripción** | **Motor V1** | **Motor V2** |
| IRepositoryItem::get\_Item() | Un índice entero fuera de intervalo devuelve EREP\_BADPARAMS. | Un índice entero fuera de intervalo devuelve DISP\_E\_BADINDEX. |
| IRelationshipCol::Insert()  ITargetObjectCol::Insert() | La inserción se anexa a la colección si la colección no está ordenada. | Error en la inserción con las colecciones no ordenadas (EREP\_COL\_NOT-SEQUENCED) y las colecciones de destino (EREP\_RELSHIP\_ORGONLY). |
| IRelationshipCol::Move()  ITargetObjectCol::Move() | Error EREP\_RELSHIP\_ ORGONLY para colecciones no ordenadas y de destino. | Error EREP\_COL\_NOT-SEQUENCED para colecciones no ordenadas, y EREP\_RELSHIP\_ ORGONLY para colecciones de destino. |
| IRepositoryObjectVersion::put\_Name() | El error EREP\_OBJ\_NO-NAMING-RELSHIP no se agrega a la cola de errores. | El error REP\_OBJ\_NONAMING-RELSHIP se agrega a la cola de errores. |
| Colecciones de nombres únicos: elemento B con nombre "B" se agrega varias veces a una colección de nombres únicos | El código de error es EREP\_RELSHIP\_DUPENAME. | El código de error es EREP\_RELSHIP\_EXISTS. |
| IRepositoryItem::get\_Name() | Se completa con éxito sobre una relación sin nombre (devuelve "0"). | Produce un error. |
| IRelationshipCol::get\_Item()  ITargetObjectCol:: get\_Item() | El argumento puede ser INTID, OBJID, Index o Name. | El argumento puede ser OBJID, Index o Name. |
| ExecuteQuery() sobre SQL Server | Comprueba la columna INTID en el conjunto de resultados. | Acepta más de un INTID siempre que sea posible una conversión implícita. |
| Actualizar sobre una colección de instancias de objetos | Actualiza las propiedades y después los objetos. | Actualiza los objetos y después las propiedades. |

#### Funcionamiento asíncrono

La carga de colecciones de objetos en respuesta a **IRepositoryODBC::ExecuteQuery()** puede ser asíncrona. El subproceso que llama debe comprobar si la carga está completa. Si el subproceso que llama intenta leer datos, actualiza la colección o construye un enumerador mientras la carga está en curso, se bloqueará hasta que la carga esté completa.

##### Indicadores

* Para establecer o borrar la opción "async" (asíncrona):

RODBC\_RESET\_OPTIONS = 1  // Restablecer todas las opciones

RODBC\_ASYNCH = 2         // Ejecución de consulta asíncrona

* Para el estado de carga del objeto:

READY = 1                // Carga completa

INPROGRESS = 2           // Carga en curso

CANCELLED = 3            // Carga cancelada (por el solicitante)

FAILED = 4               // Error en la carga (motivo desconocido)

##### Interfaces nuevas

* **IObjectCol2**: hereda de **IObjectCol**.

Métodos adicionales:

* **LoadStatus**: obtiene el estado de carga de la colección.

###### Firma

**HRESULT LoadStatus(long** \*piEstado**)**                   // Automatización

**HRESULT get\_LoadStatus(long** \*piEstado**)**           // COM

###### Argumentos

piEstado

[out, retval]

Uno de las siguientes: READY/INPROGRESS/CANCELLED/FAILED.

###### Valor devuelto

S\_OK si se ha realizado con éxito, EREP\_BADPARAMS si no se proporciona argumento de salida.

* **Cancel**: solicita la cancelación de la operación de carga en curso.

###### Firma

**HRESULT** **Cancel()**

###### Valor devuelto

S\_OK.

* **IRepositoryODBC2**: hereda de **IRepositoryODBC**.

Métodos adicionales:

* **GetOption:** obtiene el valor de la opción de carga.

###### Firma

**HRESULT** **GetOption(long** iOpción, **VARIANT** \*psValor**)**

###### Argumentos

iOpción

[in]

RODBC\_ASYNCH.

psValor

[out, retval]

VARIANT\_TRUE o VARIANT\_FALSE, según si se estableció o no la opción RODBC\_ASYNCH.

###### Valor devuelto

S\_OK si se ha realizado con éxito, EREP\_BADPARAMS si se especificó cualquier otra opción.

 **SetOption:** Establece la opción de carga de la colección. El indicador asíncrono se puede establecer si y sólo si el sistema de base de datos subyacente admite el funcionamiento asíncrono.

###### Firma

**HRESULT** **SetOption(long** iOpción, **VARIANT** sValor**)**

###### Argumentos de entrada

iOpción RODBC\_ASYNCH y sValor TRUE establecen el modo asíncrono de la carga.

iOpción RODBC\_ASYNCH y sValor FALSE desactivan el modo asíncrono.

iOpción RODBC\_RESET\_OPTIONS restablece el modo asíncrono de la carga.

###### Valor devuelto

S\_OK si se realiza con éxito, EREP\_TYPE\_COLMISMATCH si se especifica un valor distinto de TRUE o FALSE en sValor cuando iOpción es RODBC\_ASYNCH.

**Otros cambios**

* La clase **ObjectCol** es compatible con **IObjectCol** [predeterminada] y **IObjectCol2**.
* **IObjectCol2::Refresh()** actualiza de manera asíncrona la colección de objetos (vuelve a cargar la colección de objetos y actualiza los objetos del destino) cuando está activo el modo asíncrono. El subproceso que llama debe comprobar si se ha completado la actualización. Si el subproceso que llama intenta leer datos, actualizar la colección o construir un enumerador mientras la actualización está en curso, se bloqueará hasta que se complete la actualización.

#### Tipos de SQL y API que se utilizan en las definiciones de propiedades

Las dos tablas siguientes muestran los tipos de API que reconoce el motor de Repository, así como los tipos de SQL. Estos valores aparecen en las propiedades **APIType** y **SQLType** de un objeto **PropertyDef**. Para obtener información acerca de la conversión entre tipos de SQL y API, consulte la Referencia al programador de ODBC.

|  |  |
| --- | --- |
| **TIPOS DE API RECONOCIDOS POR EL MOTOR** | |
| **TIPO DE API** | **VALOR** |
| SQL\_C\_UTINYINT | -28 |
| SQL\_C\_STINYINT | -26 |
| SQL\_C\_ULONG | -18 |
| SQL\_C\_USHORT | -17 |
| SQL\_C\_SLONG | -16 |
| SQL\_C\_SSHORT | -15 |
| SQL\_C\_BINARY | -2 |
| SQL\_C\_TINYINT | -6 |
| SQL\_C\_BIT | -7 |
| SQL\_C\_CHAR | 1 |
| SQL\_C\_LONG | 4 |
| SQL\_C\_SHORT | 5 |
| SQL\_C\_FLOAT | 7 |
| SQL\_C\_DOUBLE | 8 |
| SQL\_C\_DATE | 9 |
| SQL\_C\_TIME | 10 |
| SQL\_C\_TIMESTAMP | 11 |

|  |  |
| --- | --- |
| **TIPO DE SQL** | **VALOR** |
| SQL\_LONGVARCHAR | -1 |
| SQL\_BINARY | -2 |
| SQL\_VARBINARY | -3 |
| SQL\_LONGVARBINARY | -4 |
| SQL\_BIGINT | -5 |
| SQL\_TINYINT | -6 |
| SQL\_BIT | -7 |
| SQL\_CHAR | 1 |
| SQL\_NUMERIC | 2 |
| SQL\_DECIMAL | 3 |
| SQL\_INTEGER | 4 |
| SQL\_SMALLINT | 5 |
| SQL\_FLOAT | 6 |
| SQL\_REAL | 7 |
| SQL\_DOUBLE | 8 |
| SQL\_DATETIME | 9 |
| SQL\_TIME | 10 |
| SQL\_TIMESTAMP | 11 |
| SQL\_VARCHAR | 12 |

#### Características nuevas

##### Propiedades

Se han agregado dos nuevas propiedades a la interfaz **IRepository2** para indicar la versión del archivo de base de datos.

* **MajorDBVersion**

###### Firma

**HRESULT MajorDBVersion(long** \*piVersiónPralBD**)**             // Automatización

**HRESULT get\_MajorDBVersion(** long \*piVersiónPralBD**)**      // COM

###### *Argumentos*

piVersiónPralBD

[out, retval]

El número de versión principal de la primera versión del motor de Repository que presentó este formato de base de datos.

* **MinorDBVersion**

###### Firma

**HRESULT MinorDBVersion(long** \*piVersiónSecundariaBD**)**             // Automatización

**HRESULT get\_MinorDBVersion(** long \*piVersiónSecundariaBD**)**      // COM

###### *Argumentos*

piVersiónSecundariaBD

[out, retval]

El número de versión secundaria de la primera versión del motor de Repository que presentó este formato de base de datos.

##### Métodos

Se agregó un método nuevo **CreateObjectEx()** a la interfaz de **IRepository2**.Crea la primera versión de un nuevo objeto de Repository del tipo especificado. A la versión recién creada se le asigna el identificador de versión del objeto que se ha pasado como argumento, a diferencia de **IRepository::CreateObject()**, en donde Repository asigna el Id. de versión. Tiene la siguiente sintaxis:

###### Firma

**HRESULT IRepository2::CreateObjectEx(**

**VARIANT** IdTipo,

**VARIANT** IdObjeto,

**VARIANT** IdVersiónExt,

**IRepositoryObjectVersion** \*\*ppVersiónObjetoRep**)**;

###### Argumentos

IdTipo

[in]

Hace lo mismo que **IRepository::CreateObject**

IdObjeto

[in]

Hace lo mismo que **IRepository::CreateObject**

IdVersiónExt

[in]

Éste es el identificador de la versión del objeto (20 bytes) que se va a asignar a la primera versión del objeto

ppVersiónObjetoRep

[out]

Éste es el puntero **IRepositoryObjectVersion** hacia la versión recién creada

### Limitaciones conocidas de Microsoft Repository 2.0

#### 1. Limitaciones relacionadas con las interfaces y métodos asociados

**IRepositoryObjectVersion**

* **MergeVersion()**: Las relaciones se insertan al final de la colección ordenada.

**IRelationshipCol** y **ITargetObjectCol**

* **Remove()**: Quitar un elemento en una colección ordenada no actualiza el orden de la colección.

**IVersionAdminInfo**

* **VersionModifyTime()**: No cambia cuando se modifica una colección de relaciones de origen o de objetos de destino.
* **ModifyByUser()**: No cambia cuando se modifica una colección de relaciones de origen o de objetos de destino.

**IRepository2**

* **CreateObject()**: Sólo se puede llamar desde el depósito compartido pero no desde un área de trabajo. La solución es crear el objeto mediante el depósito central e incluirlo en el área de trabajo.

**IClassDef** e **IInterfaceDef**

* **ObjectInstances()**: No alcanza al área de trabajo.

#### ****2. Constantes de Repository****

En la documentación de Repository, se menciona que la longitud máxima, en bytes, del nombre que una relación asigna a su objeto de destino es 260, como se define en la constante RELSHIPNAMESIZE. Este valor es realmente 249.

#### ****3. Convenciones para nombres****

Los nombres de relaciones y objetos deben seguir estas convenciones:

1. Los nombres no pueden tener más de 249 caracteres.

2. Se puede utilizar cualquier carácter en un nombre.

3. El nombre puede contener espacios en blanco al principio o en medio. También puede ser una cadena vacía.

4. Si el nombre es todo espacios en blanco, se trata como una cadena vacía.

Esto se aplica a los nombres que se utilizan entre comillas o se almacenan en variables. Por ejemplo, la interfaz "ICarpeta" siguiente debe cumplir estas restricciones:

oFolder("ICarpeta").FolderName

        -o bien-

oFolder(sICarpeta).FolderName

       where: sICarpeta = "ICarpeta"

Sin embargo, las normas no son para los nombres que se especifican fuera de comillas (como los nombres de propiedad o los nombres de relación). Por ejemplo, la colección de relaciones "RcCarpetaContiene" below:

oFolder("IFolder").RcFolderContains.Count

se debe ajustar a las normas de nombres de Microsoft Visual Basic, excepto en el límite que es de 127 caracteres en lugar de los 255 caracteres que permite Visual Basic.

#### 4. Procedimientos almacenados

El nombre del procedimiento almacenado de una tabla se genera precediendo al nombre de la tabla con la cadena "R\_i". Puesto que los nombres de tabla son únicos, así se generarán nombres de procedimiento almacenado únicos. Sin embargo, si la longitud del nombre de la tabla es mayor que **MaxIdentifierLength**-3, al algoritmo de generación de nombre de la tabla produce un error. Por esta razón, un usuario no puede proporcionar un nombre de tabla con una longitud mayor que **MaxIdentifierLength**-3. Si se proporciona un nombre más largo se produce el error EREP\_BADNAME.

Cuando el usuario no proporciona ningún nombre de tabla para una interfaz, el motor genera automáticamente el nombre de tabla a partir del nombre de la interfaz. Si el nombre de la interfaz, sin la letra "I" inicial, es menor que **MaxIdentifierLength**-4, el nombre de la interfaz se utilizará como nombre de tabla. En cualquier otro caso, el nombre de la interfaz se trunca a **MaxIdentifierLength**-7, y se agrega al nombre un número de 4 caracteres (denominado "*uniquificador*") para hacerlo único, antes de anexar el prefijo R\_i.

El motor utiliza argumentos con nombre para llamar a los procedimientos almacenados. Un argumento con nombre comienza con el carácter "@" y no puede tener una longitud mayor que **MaxIdentifierLength**. De esta manera, los nombres de propiedad, que también son nombres de columna, no pueden tener una longitud mayor que **MaxIdentifierLength**-1.

Los valores de **MaxIdentifierLength** son:

* 30 caracteres para SQL Server 6.5
* 128 caracteres para SQL Server 7.0

#### 5. Definiciones de tablas

Las tablas siguientes no se incluyen en el archivo de ayuda de Repository:

* **RTblSumInfo**
* **RTblNamedObj**
* **RTblVersionAdminInfo**

Estas tablas se explican junto con los nombres de columna y los tipos de datos.

**RTblSumInfo**

**RTblSumInfo** es una tabla específica de interfaz; sus columnas corresponden a las propiedades expuestas mediante la interfaz **ISummaryInformation**. De manera predeterminada, ninguna clase del Modelo de información de tipos de Repository implementa **ISummaryInformation**. Por lo tanto la base de datos de Repository no incluye de manera predeterminada la tabla **RTblSumInfo**. Repository omite la tabla de la base de datos para ahorrar espacio. Sin embargo, tan pronto como inserte en Repository cualquier clase que implemente **ISummaryInformation**, Repository creará la tabla.

|  |  |  |
| --- | --- | --- |
| **Nombre de columna** | **Tipo de dato** | **Descripción** |
| IntID | RTIntID | Identificador interno de la clase. |
| Z\_BranchID\_Z | RTBrID | Indica la bifurcación del gráfico de versión que contiene el intervalo de elementos a los que se van a aplicar los valores de propiedad de esta fila. |
| Z\_VS\_Z | RTVerID | Identificador de versión en bifurcación que indica el límite inferior del intervalo de elementos a los que se van a aplicar los valores de propiedad de esta fila. |
| Z\_VE\_Z | RTVerID | Identificador de versión en bifurcación que indica el límite superior del intervalo de elementos a los que se van a aplicar los valores de propiedad de esta fila. |
| Comments | RTLongString | Campo que se utiliza para comentarios. |
| ShortDesc | RTLongString | Descripción del objeto. |

**RTblNamedObj**

**RTblNamedObj** es una tabla específica de interfaz; sus columnas corresponden a las propiedades expuestas mediante la interfaz **INamedObject**. De manera predeterminada, ninguna clase del Modelo de información de tipos de Repository implementa **INamedObject**. Por lo tanto la base de datos de Repository no incluye de manera predeterminada la tabla **RTblNamedObj**. Repository omite la tabla de la base de datos para ahorrar espacio. Sin embargo, tan pronto como inserte en Repository cualquier clase que implemente **INamedObject**, Repository creará la tabla.

|  |  |  |
| --- | --- | --- |
| **Nombre de columna** | **Tipo de datos** | **Descripción** |
| IntID | RTIntID | Identificador interno de la clase. |
| Z\_BranchID\_Z | RTBrID | Indica la bifurcación del gráfico de versión que contiene el intervalo de elementos a los que se van a aplicar los valores de propiedad de esta fila. |
| Z\_VS\_Z | RTVerID | Número de versión en bifurcación que indica el límite inferior del intervalo de elementos a los que se van a aplicar los valores de propiedad de esta fila. |
| Z\_VE\_Z | RTVerID | Número de versión en bifurcación que indica el límite superior del intervalo de elementos a los que se van a aplicar los valores de propiedad de esta fila. |
| Name | RTLongString | Nombre del objeto. |

**RTblVersionAdminInfo**

**RTblVersionAdminInfo** es una tabla específica de interfaz; sus columnas corresponden a las propiedades expuestas mediante la interfaz **IVersionAdminInfo**. De manera predeterminada, ninguna clase del Modelo de información de tipos de Repository implementa **IVersionAdminInfo**. Por lo tanto la base de datos de Repository no incluye de manera predeterminada la tabla **RTblVersionAdminInfo**. Repository omite la tabla de la base de datos para ahorrar espacio. Sin embargo, tan pronto como inserte en Repository cualquier clase que implemente **IVersionAdminInfo**, Repository creará la tabla.

|  |  |  |
| --- | --- | --- |
| **Nombre de columna** | **Tipo de dato** | **Descripción** |
| IntID | RTIntID | Identificador interno de la clase. |
| Z\_BranchID\_Z | RTBrID | Indica la bifurcación del gráfico de versión que contiene el intervalo de elementos a los que se van a aplicar los valores de propiedad de esta fila. |
| Z\_VS\_Z | RTVerID | Identificador de versión en bifurcación que indica el límite inferior del intervalo de elementos a los que se van a aplicar los valores de propiedad de esta fila. |
| Z\_VE\_Z | RTVerID | Identificador de versión en bifurcación que indica el límite superior del intervalo de elementos a los que se van a aplicar los valores de propiedad de esta fila. |
| VersionCreateTime | Date/Time | Hora en que se creó la versión. |
| VersionModifyTime | Date/Time | Hora en que se modificó la versión. |
| CreateByUser | RTLongString | Usuario que creó la versión. |
| ModifyByUser | RTLongString | Usuario que modificó la versión. |

#### 6. Otras limitaciones

* El Asistente para migración (MigRepV2.exe), el Explorador de depósito (Repbrows.exe) y el Administrador de componentes visuales (VCM) no son compatibles con archivos DSN (Nombres de orígenes de datos). Estas herramientas sólo funcionan con un DSN de tipo Access o SQL Server.
* En el Asistente para migración, si trabaja con un DSN de tipo SQL Server, se le pedirá que escriba un nombre y una contraseña a menos que el DSN está configurado para utilizar la opción de autenticación de Windows NT (en el Administrador de orígenes de datos ODBC).
* En el tema "Ejemplos de combinación de versiones," hay una imprecisión en el segundo gráfico. Las propiedades de la versión V8 podrían cambiar tras la combinación, de (Grande, rojo, null) a (Medio, rojo, null).
* La documentación de la lista de **CollectionDefFlags** no mencionaba el siguiente indicador:

COLLECTION\_NEWORGVERSIONSDONOTPARTICIPATE = 64

Este indicador está relacionado con otro indicador de la misma enumeración:

COLLECTION\_NEWORGVERSIONSPARTICIPATE = 32

Los dos indicadores tienen significados opuestos. Por lo tanto, no se pueden activar al mismo tiempo. Sin embargo, puesto que la configuración predeterminada de cada uno de ellos es *no establecido*, es posible que ambos indicadores estén no activados al mismo tiempo. Si ambos indicadores están no activados, el depósito funciona como si estuviese activado el indicador COLLECTION\_NEWORGVERSIONSPARTICIPATE. Es decir, durante la creación de una nueva versión de un objeto de depósito, Microsoft Repository no copia colecciones origen nuevas desde una versión creada anteriormente a una versión posterior.

* Si en **SQLSize** se establece un valor mayor que 65535, el motor divide el número entero por 65536 y establece como **SQLSize** el valor del resto de la división, pero no se devuelve ningún error.
* En el tema de ayuda "Algoritmo de migración*"* hay referencias a dos tablas, **RTblRelColPairs** y **RTblTimestamp**. En la base de datos de Repository 2.0, la primera tabla no existe, y la segunda tabla se ha reemplazado con **RTblVersionAdminInfo**.

#### ****Para obtener más información****

Puede dirigirse a los siguientes orígenes para obtener más información sobre Microsoft Repository:

* Compruebe las noticias más recientes del sitio Web de Microsoft Repository (<http://www.microsoft.com/repository>). También puede descargar la documentación de Repository desde este sitio.
* En la documentación de Microsoft Visual Studio 6.0, encontrará *Guía del programador de Repository para Visual Basic*, *Referencia del complemento Repository*, *Explorador de Repository* y *Convertir bases de datos de Repository 1.0*. Para localizar estos documentos comience en la parte superior y siga esta ruta de acceso:

MSDN Library de Visual Studio 6.0    Documentación de Visual Studio        Herramientas de componentes, diseño y anális             Documentos de Microsoft Repository 2.0

* En la documentación de Microsoft Visual Basic 6.0, encontrará la *Guía del programador de Repository para Visual Basic* y *Referencia del complemento Repository*. Para localizar estos documentos comience en la parte superior y siga esta ruta:

MSDN Library de Visual Studio 6.0    Documentación de Visual Basic        Uso de Repository con Visual Basic