47

# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES.

CAMPUS ARAGÓN

## DISEÑO E IMPLEMENTACIÓN DE ESQUEMAS DE REPLICACIÓN DE DATOS UTILIZANDO REPLICATION SERVER.

20016

# T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

JAIME MOISÉS MORALES VALENTINO.

ASESOR:
ING. GLADIS E. FUENTES CHÁVEZ.

MÉXICO 2001

**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**
**CAMPUS ARAGÓN**
**ÁREA DE INGENIERÍA EN COMPUTACIÓN**

M EN R.I. CARLOS EDUARDO LEVY VAZQUEZ
Director del Campus Aragón UNAM.
P r e s e n t e .

Por este medio me permito comunicar a usted que revisé la tesis titulada:

"DISEÑO E IMPLEMENTACION DE ESQUEMAS DE REPLICACION DE DATOS UTILIZANDO REPLICATION SERVER"

Que presenta el pasante: JAIME MOISES MORALES VALENTINO

Con número de cuenta: 8814776-1

Para obtener el título de: **INGENIERO EN COMPUTACIÓN.**

Considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el **EXAMEN PROFESIONAL** correspondiente, otorgo mi **VOTO APROBATORIO.**

A t e n t a m e n t e
**"POR MI RAZA HABLARÁ EL ESPÍRITU"**

San Juan de Aragón, Edo., de México, a 15 de JUNIO del 2001.

Revisor de Tesis
ING. GLADIS EMILIA FUENTES CHAVEZ

M. en C. Jesús Díaz Barriga Arceo
Jefe de Carrera

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
CAMPUS ARAGÓN
ÁREA DE INGENIERÍA EN COMPUTACIÓN

M EN R.I. CARLOS EDUARDO LEVY VAZQUEZ
Director del Campus Aragón UNAM.
P r e s e n t e .

Por este medio me permito comunicar a usted que revisé la tesis titulada:

"DISEÑO E IMPLEMENTACION DE ESQUEMAS DE REPLICACION DE DATOS UTILIZANDO REPLICATION SERVER"

Que presenta el pasante: JAIME MOISES MORALES VALENTINO

Con número de cuenta: 8814776-1

Para obtener el título de: **INGENIERO EN COMPUTACIÓN.**

Considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el **EXAMEN PROFESIONAL** correspondiente, otorgo mi **VOTO APROBATORIO.**

A t e n t a m e n t e
**"POR MI RAZA HABLARÁ EL ESPÍRITU"**

San Juan de Aragón, Edo., de México, a 15 de JUNIO del 2001.

Revisor de Tesis
ING. ALEJANDRO RENE GONZALEZ PONCE

M. en C. Jesús Díaz Barriga Arceo
Jefe de Carrera

**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**
**CAMPUS ARAGÓN**
**ÁREA DE INGENIERÍA EN COMPUTACIÓN**

M EN R.I. CARLOS EDUARDO LEVY VAZQUEZ
Director del Campus Aragón UNAM.
P r e s e n t e .

Por este medio me permito comunicar a usted que revisé la tesis titulada:

"DISEÑO E IMPLEMENTACION DE ESQUEMAS DE REPLICACION DE DATOS UTILIZANDO REPLICATION SERVER"

Que presenta el pasante: JAIME MOISES MORALES VALENTINO

Con número de cuenta: 8814776-1

Para obtener el título de: **INGENIERO EN COMPUTACIÓN.**

Considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el **EXAMEN PROFESIONAL** correspondiente, otorgo mi **VOTO APROBATORIO.**

A t e n t a m e n t e
**"POR MI RAZA HABLARÁ EL ESPÍRITU"**

San Juan de Aragón, Edo., de México, a 15 de JUNIO del 2001.

Revisor de Tesis
ING. ERNESTO PEÑALOZA ROMERO

M. en C. Jesús Díaz Barriga Arceo
Jefe de Carrera

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
CAMPUS ARAGÓN
ÁREA DE INGENIERÍA EN COMPUTACIÓN

M EN R.I. CARLOS EDUARDO LEVY VAZQUEZ
Director del Campus Aragón UNAM.
P r e s e n t e .

Por este medio me permito comunicar a usted que revisé la tesis titulada:

"DISEÑO E IMPLEMENTACION DE ESQUEMAS DE REPLICACION DE DATOS UTILIZANDO REPLICATION SERVER"

Que presenta el pasante: JAIME MOISES MORALES VALENTINO

Con número de cuenta: 8814776-1

Para obtener el título de: **INGENIERO EN COMPUTACIÓN.**

Considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el **EXAMEN PROFESIONAL** correspondiente, otorgo mi **VOTO APROBATORIO.**

A t e n t a m e n t e
**"POR MI RAZA HABLARÁ EL ESPÍRITU"**

San Juan de Aragón, Edo., de México, a 15 de JUNIO del 2001.

Revisor de Tesis
ING. RICARDO GUTIERREZ OROZCO

M. en C. Jesús Díaz Barriga Arceo
Jefe de Carrera

**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**
**CAMPUS ARAGÓN**
**ÁREA DE INGENIERÍA EN COMPUTACIÓN**

M EN R.I. CARLOS EDUARDO LEVY VAZQUEZ
Director del Campus Aragón UNAM.
P r e s e n t e .

Por este medio me permito comunicar a usted que revisé la tesis titulada:

"DISEÑO E IMPLEMENTACION DE ESQUEMAS DE REPLICACION DE DATOS UTILIZANDO REPLICATION SERVER"

Que presenta el pasante: JAIME MOISES MORALES VALENTINO

Con número de cuenta: 8814776-1

Para obtener el título de: **INGENIERO EN COMPUTACIÓN.**

Considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el **EXAMEN PROFESIONAL** correspondiente, otorgo mi **VOTO APROBATORIO.**

A t e n t a m e n t e
**"POR MI RAZA HABLARÁ EL ESPÍRITU"**

San Juan de Aragón, Edo., de México, a 15 de JUNIO del 2001.

Revisor de Tesis
ING. DAVID MOISES TERAN PEREZ

M. en C. Jesús Díaz Barriga Arceo
Jefe de Carrera

**ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGÓN**

**JEFATURA DE INGENIERÍA EN COMPUTACIÓN**
**OFICIO: ENAR/JACO/182/2001.**

**ASUNTO:** Asignación de Jurado.

**LIC. ALBERTO IBARRA ROSAS**
**Secretario Académico**
**P r e s e n t e .**

JUN 28 7 15 PM 2001

Por este conducto me permito presentar a usted, nombre de los profesores que sugiero integren el Sínodo del Examen Profesional del alumno **JAIME MOISÉS MORALES VALENTINO** presenta el tema de tesis: **"DISEÑO E IMPLEMENTACIÓN DE ESQUEMAS DE REPLICACIÓN DE DATOS UTILIZANDO REPLICATION SERVER ".**

| | |
|---|---|
| **PRESIDENTE** | **ING. DAVID MOISÉS TERAN PÉREZ** |
| **VOCAL:** | **ING. ERNESTO PEÑALOZA ROMERO** |
| **SECRETARIO:** | **ING. GLADIS EMILIA FUENTES CHAVEZ** |
| **SUPLENTE:** | **ING. RICARDO GUTIERREZ OROZCO** |
| **SUPLENTE:** | **ING. ALEJANDRO RENÉ GONZÁLEZ PONCE** |

Quiero subrayar que la directora de tesis es la **Ing. Gladis Emilia Fuentes Chavez**, la cual está incluido con base en lo que reza el reglamento de Exámenes Profesionales de ésta Escuela.

Sin otro en particular, me es grato enviarle un cordial saludo.

**A T E N T A M E N T E**
**"POR MI RAZA HABLARÁ EL ESPÍRITU"**
**San Juan de Aragón, Edo. de México, marzo 12 del 2001.**
**EL JEFE DE CARRERA**

**ING. JESÚS DÍAZ BARRIGA ARCEO**

**c.c.p. Lic. Ma. Teresa Luna Sánchez.- Jefa del Departamento de Servicios Escolares.**
**Ing. Gladis Emilia Fuentes Chavez- Directora de la tesis.**

**JDA/mav.**

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
CAMPUS ARAGÓN

SECRETARÍA ACADÉMICA

M. en C. JESÚS DÍAZ BARRIGA ARCEO
Jefe de la Carrera de Ingeniería en Computación,
Presente.

En atención a la solicitud de fecha 20 de junio del año en curso, por la que se comunica que el alumno JAIME MOISÉS MORALES VALENTINO, de la carrera de Ingeniero en Computación, ha concluido su trabajo de investigación intitulado "DISEÑO E IMPLEMENTACIÓN DE ESQUEMAS DE REPLICACIÓN DE DATOS UTILIZANDO REPLICATION SERVER", y como el mismo ha sido revisado y aprobado por usted, se autoriza su impresión; así como la iniciación de los trámites correspondientes para la celebración del Examen Profesional.

Sin otro particular, reitero a usted las seguridades de mi atenta consideración.

A t e n t a m e n t e
"POR MI RAZA HABLARÁ EL ESPÍRITU"
San Juan de Aragón, México, 21 de junio del 2001
EL SECRETARIO

Lic. ALBERTO IBARRA ROSAS

C p Asesor de Tesis.
C p Interesado.

AIR/RCC/vr

Recibí
28/6/2001

Ing. en Computación
28 JUN 2001
Gladis E Fuentes Chávez

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

San Juan de Aragón a 6 de abril de 2001

ABR 6 11 35 AM 2001

SECRETARIA ACADEMICA

LIC. ALBERTO IBARRA ROSAS
JEFE DE LA UNIDAD ACADEMICA
P R E S E N T E.

Por medio de la presente se hace constar que el alumno **MORALES VALENTINO JAIME MOISÉS** con número de cuenta 08814776-1, ha concluido satisfactoriamente su trabajo de tesis denominado "**DISEÑO E IMPLEMENTACIÓN DE ESQUEMAS DE REPLICACIÓN DE DATOS UTILIZANDO REPLICATION SERVER**".

Se extiende la presente para que el alumno continúe con sus trámites de titulación.

Vo. Bo.

Ing. Gladis Fuentes Chávez
Asesor

M. en C. Jesús Díaz Barriga Arceo
Jefe de la Carrera

VNIVERSIDAD NACIONAL AVTÓNOMA DE MEXICO

*UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO*
*CAMPUS ARAGON*
*ÁREA DE INGENIERÍA EN COMPUTACIÓN*

*San Juan de Aragón, Edo. de Méx., a* 28 *de* MARZO *del 2000.*

Tesis que desarrollará el (la) C.:

Jaime Moisés Morales Valentino

De la Carrera de Ingeniería en Computación

***Título de la Tesis.***

DISEÑO E IMPLEMENTACION DE ESQUEMAS DE REPLICACION DE DATOS UTILIZANDO REPLICATION SERVER

**Capitulos:**

CAPITULO I. INTRODUCCION

CAPITULO II. ELEMENTOS QUE INTERVIENEN EN LA REPLICACION DE DATOS

CAPITULO III. DISEÑO DE UN ESQUEMA DE REPLICACION

CAPITULO IV. PROGRAMA GENERADOR DE SCRIPTS DE REPLICATION SERVER

CAPITULO V. CONCLUSIONES

ING. GLADIS F. FUENTES CHAVEZ
Director de Tesis

M. en C. Jesús Díaz Barriga Arceo
Jefe de Carrera

Domicilio: Primera Cerrada de Alberto Salinas No. 40 Col Aviación Civil

CP. 15740

Teléfono: 57857494

Promedio: 9.22

San Juan de Aragón, Edo. de México, a 27 de Marzo del 2000.

**Lic. Carlos Eduardo Levy Vázquez**
**Director de la Escuela Nacional de Estudios**
**Profesionales Aragón, UNAM.**
**PRESENTE:**

Con motivo de haber acreditado la totalidad de materias del Plan de Estudios vigente en la Carrera de Ingeniería en Computación, solicito tenga bien autorizar al:

**C. Ing. Gladis E. Fuentes Chávez**

Para dirigirme la Tesis de la cual se anexa al presente Titulo y Capitulado.

Esta designación la hago en virtud del **AMPLIO CONOCIMIENTO QUE EL INGENIERO GLADIS FUENTES POSEE EN RELACION AL TEMA DE LAS BASES DE DATOS, MISMO QUE SERA BASICO PARA EL DESARROLLO DE LA TESIS.**

No dudando que me veré favorecido en mi solicitud, agradezco la atención que se sirva presentar al presente.

ATENTAMENTE

| | |
|---|---|
| Firma: | |
| Nombre: | Jaime Moises Morales Valentino |
| Número de Cuenta: | 8814776-1 |
| 1ra. Inscripción: | 91-1 |
| Ultima Incripción: | 95-2 |
| Fecha en que acredita la última asignatura: | 95-2 |

# AGRADECIMIENTOS

Mamá:

Muchas Gracias por haberme dado la oportunidad de vivir y por haberme mostrado el camino hacia una vida honesta y responsable. ¡Te quiero mucho!. Y quiero que sepas que Dios no me pudo haber puesto en mejores manos que en las tuyas ... ¡Gracias!.

Dulce:

¡Mi amor!. Te agradezco infinitamente por el apoyo y comprensión que me has dado durante los momentos difíciles que hemos vivido. Y agradecerte por los miles de momentos felices que hemos pasado juntos.

Beatriz:

Mi Bety, quiero que sepas lo feliz y orgulloso que me hacer ser tu hermano, agradezco el enorme sacrificio que has hecho para ayudarme a lograr esto.

Elena:

Gracias por la ayuda y apoyo que me has dado a lo largo de toda mi vida. Gracias por que siempre he contado contigo y ¿sabes una cosa? ... siempre contarás conmigo.

Mariana:

Hija, ¡gracias por hacerme tan feliz!, y haber llegado a mi vida para enseñarme lo bello que es ser papá.

Andrea:

Aun estás muy pequeña, pero agradezco la felicidad que has traído a nuestra familia ... ¡Muchas Gracias chiquita!

Lic. Guillermo Rico Díaz

Por creer en mí y brindarme su apoyo incondicional, ¡Muchas Gracias padrino!.

†Mamá Ofe:
En dondequiera que te encuentres ... ¡Muchas Gracias!.

# INDICE

# INDICE DE ILUSTRACIONES

# INDICE DE TABLAS

# CAPITULO I. INTRODUCCION

Si se analiza la evolución de los distintos sectores económicos en los últimos años, se puede comprobar la excepcional expansión que la información ha tenido en relación con otros sectores, llegándose a calificar esta expansión, y los profundos cambios a los que ha dado lugar, como una **segunda revolución industrial**, marcando el comienzo de una nueva era en el desarrollo de la humanidad.

En la era industrial lo más importante era el uso del capital, dinero y recursos tangibles, para generar nuevos productos. En el presente los recursos básicos son las ideas y el uso de la información. En estos momentos, la sociedad y la economía tienen una naturaleza global, las actividades en Londres y Tokio, por ejemplo, tienen influencia sobre las operaciones comerciales en Nueva York, Chicago, Atlanta y Los Angeles. Las influencias globales han reemplazado a las economías nacionalistas de la era industrial.

Además, la centralización y descentralización han sido una oposición en curso desde el origen del comercio. Después de la Segunda Guerra Mundial, las corporaciones fueron ordenadas en una estructura jerárquica y altamente centralizada. Desde los años 80, muchas organizaciones se condujeron hacia la descentralización, así los niveles corporativos se convirtieron en especies puestas en peligro, entonces se convirtió en un desafío mantener el control eficaz de las unidades de negocio lejanas a las oficinas corporativas.

Hoy, las compañías desean ambas maneras. Desean las ventajas del control prometidas por la centralización, junto con la flexibilidad de la respuesta a los clientes traídos por la descentralización. Quisieran los encargados locales tomar la iniciativa para conseguir resultados, pero siguiendo la estrategia corporativa.

Así, ante el inminente proceso de globalización que estamos viviendo, en el cual las compañías se fusionan para volverse más competitivas y en las cuales por razones estratégicas crean unidades del negocio alrededor del mundo, se vuelve cada vez más necesario contar con información distribuida en las diversas unidades que forman parte de la organización sin importar su ubicación física y sobre todo respetando las políticas establecidas a nivel central.

Los sistemas adecuados ayudan a las compañías a encontrar el equilibrio entre la centralización y la descentralización, así, los datos mismos que son un activo corporativo, pueden ser almacenados en forma central o distribuidos a lo largo de las unidades que conforman la empresa. La distribución de los datos, hecha correctamente, es una herramienta que ayuda a las compañías a poner datos necesarios en las manos de responsables locales, manteniendo un control central firme de estos datos.

Con todo lo anterior y derivado del avance tecnológico que han sufrido las comunicaciones al grado de que en la actualidad es posible realizar llamadas telefónicas desde la cima de un volcán o bien desde la mitad del océano Pacífico. Se ha vuelto muy común, que las grandes organizaciones

posean un sin fin de redes locales en las áreas que conforman la empresa, y estas a su vez se enlacen por medio una red de área amplia o inclusive utilizando la red redes, es decir, Internet; la cual día con día disminuye su costo de acceso hasta llegar a ser totalmente gratis como se ha visto con los diversos servicios que han aparecido tales como tutopia o terra aquí en México.

Aunado a lo anterior, se ha alcanzado un nivel de automatización el cual permite hoy día realizar consultas de saldos vía telefónica, comprar boletos vía Internet o realizar pagos de servicios sin la necesidad de salir de casa. Todo esto se debe en parte al desarrollo que han sufrido las bases de datos, mismas que permiten almacenar y mantener cientos o miles de registros perfectamente organizados y los que son utilizados o creados por los sistemas de información que se adoptan en las compañías.

Esto nos lleva a pensar que la información de aquellas áreas lejanas físicamente del lugar o sitio en dónde se requiere dicha información, pueda ser transmitida por medio de una red de datos al sitio en dónde se necesite, generalmente con la finalidad de conocer una situación particular de la empresa en su conjunto para apoyar la toma de decisiones.

De lo antes mencionado, surge un término que nombraremos en primera instancia **Transmisión de Información**, mismo que por sí solo puede expresar un si número de significados ya que la palabra información posee un amplio sentido en el mundo de la informática. Así, es necesario establecer un sentido más concreto al término información el cual, de acuerdo al propósito de este trabajo se considerará como registros en una base de datos por ser estas últimas un elemento fundamental en un esquema de replicación.

Junto a la evolución de las bases de datos, ha surgido un lenguaje especialmente diseñado para el manejo, actualización y consulta de las bases de datos. A ese lenguaje se le ha denominado SQL (Structured Query Language conocido como sequel). SQL se ha vuelto un lenguaje estándar en el ambiente de bases de datos relacionales, sobre todo en lo que se refiere a las sentencias de inserción, actualización y/o borrado de registros en una tabla, mismos que son prácticamente iguales en cualquier sistema de base de datos actual.

Al hablar de inserciones, actualizaciones y/o borrado de registros, es necesario mencionar el concepto transacción, el cual (para no entrar en detalle) se basa en el principio "todo o nada", es decir, un conjunto de inserciones, actualizaciones y/o borrado de registros, se realizan todas o no se realiza ninguna.

Ahora bien, el concepto replicación podemos considerarlo como sinónimo de duplicar o repetir, es decir, la replica de información puede considerarse como una duplicidad o repetición de información y dado que se estableció previamente que información se entendería por registros en una base de datos, entonces replica de información sería lo mismo a replica de registros, es decir, duplicidad o repetición de registros. Considerando que en las bases de datos actuales, los registros se insertan mediante instrucciones SQL pertenecientes a una transacción, entonces el hablar de replica de transacciones es equivalente a hablar de replica de registros. Así pues llegamos a que se está dando una duplicidad o repetición de transacciones. La ventaja que se presenta al manejar esta duplicidad, consiste en que es posible repetir la transacción original en una base de datos

diferente en a dónde ésta se llevó a cabo originalmente, así de esta forma, tenemos que una transacción puede enviarse a otro sito en dónde pueda realizarse sobre un base de datos diferente. Por lo que se ha duplicado un registro en un sitio diferente al original.

Replication Server es un producto cuyo propósito fundamental es el de replicar y administrar transacciones entre diferentes sitios conectados a él. Replication Server toma las transacciones de una base de datos fuentes, mismas que son enviadas o transmitidas mediante una red (sea LAN o WAN) a otro punto para ser aplicada en una o varias bases de datos destino. De tal forma, sí se inserta un registro en un lugar, el Replication Server detecta esta transacción misma que es transmitida a otro punto para que se lleve a cabo su replica en un lugar remoto, con lo cual obtenemos que el registro que se insertó en un lugar origen se inserta también en un lugar destino obteniendo así la **Transmisión de Información**.

Por lo tanto, la finalidad de esta tesis es diseñar e implementar esquemas de replicación de bases de datos usando como herramienta a Replication Server, para administrar eficazmente la información entre las diversas unidades que forman parte de una organización. Todo esto con la intención de disponer de la información en el lugar requerido a través de una red de datos para así conocer situaciones particulares o generales de la empresa u organización en su conjunto y así favorecer la toma de decisiones.

Para ello, en el segundo capítulo de este trabajo se establecerá el marco teórico de los elementos que participan en un esquema de replicación, abordando principalmente las bases de datos a través de una reseña histórica, una descripción de los componentes básicos de una base de datos, resaltando la importancia de las tablas como objetos fundamentales en un esquema de replicación. Se mostrará además el lenguaje SQL y sus comandos básicos que nos servirán como pauta para profundizar sobre el concepto transacción al ser esta la unidad básica de replicación. Será necesario el mencionar las funciones de un manejador de bases de datos al ser este una interfaz entre el concepto de bases de datos y la realidad, no sin olvidar describir el funcionamiento de lo que se conoce como log de transacciones al ser este la fuente directa para la replicación.

Necesariamente se tendrá que dedicar un espacio para tratar más ampliamente el concepto de replicación y por último se describirá el funcionamiento y arquitectura del Replication Server para poder así comprender las ventajas y limitaciones que como cualquier producto este posee y de esta forma continuar con el diseño de esquemas de replicación.

En el tercer capítulo se proporcionarán parámetros y técnicas fundamentales para la concepción y diseño de un esquema de replicación. Para ello se mencionarán consideraciones que se deben tener en cuenta para poder realizar un diseño óptimo de un esquema de replicación. De igual forma se proporcionará una metodología para diseñar un esquema de replicación, que como en cualquier otra actividad es la base de un buen diseño con el fin de lograr una implementación exitosa (vease la Figura 1.1.). Y por último se darán algunas recomendaciones que ha surgido de la experiencia propia en el diseño de un esquema de replicación.

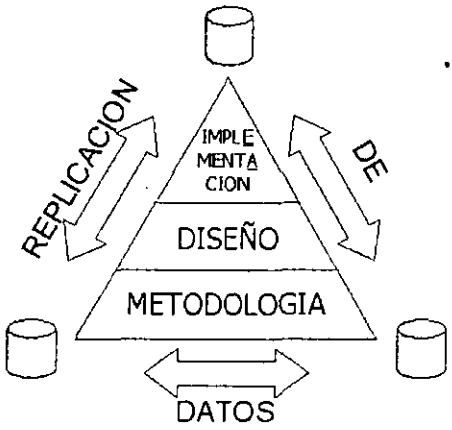



**Figura 1.1.**

En el cuarto capítulo se describe brevemente un proceso de implementación típico que a pesar de no ser complejo en escencia, da la oportunidad para desarrollar el programa llamado GSRS el cual permite crear rápidamente los scripts de replicación basándose en el diseño del esquema. Estos scripts, como se verá a lo largo de este trabajo, es una labor dura y tediosa en el proceso de implementación de un esquema de replicación.

# CAPITULO II. ELEMENTOS QUE INTERVIENEN EN LA REPLICACION DE DATOS

OBJETIVO: Establecer el marco teórico de los elementos que participan en un esquema de replicación de datos basado en Replication Server.

## Información.

Las cualidades que debe poseer la información, y que hacen de ella un recurso fundamental, son básicamente: precisión, oportunidad, compleción[1], significado e integridad. Todas ellas en el grado que exija cada sistema concreto.

La **precisión** se puede definir como el porcentaje de información correcta, sobre la información total del sistema, mientras que la **oportunidad,** se refiere al tiempo transcurrido desde el momento en que se produjo el hecho que originó el dato, hasta el momento en que la información se pone a disposición del usuario.

La **compleción** significa que ha de ser completa para poder cumplir sus fines. La compleción absoluta es imposible de conseguir, y lo que se suele pretender en los sistemas de información es alcanzar un nivel que se considere suficiente, el cual dependerá de dos factores: de los datos existentes en el sistema de información y de los que el sistema es capaz de localizar ante una consulta concreta. En este segundo factor, influirá la flexibilidad e idoneidad del lenguaje de recuperación y el acierto en la formulación de la consulta.

La información que se suministra al usuario debe ser también **significativa**; es decir, ha de poseer el máximo contenido semántico posible, ya que sin él, no constituirá verdadera información.

Asimismo, toda la información contenida en el sistema debe ser **coherente** en sí misma, además consistente con las reglas semánticas propias del mundo real al que ha de representar lo más fielmente posible. Esto en las bases de datos se conoce con el nombre de **integridad**.

Otra característica que debe poseer la información es la **seguridad**, ya que esta ha de ser protegida tanto frente a su deterioro (por causas físicas o lógicas) como frente a accesos no autorizados.

Dado que en el presente trabajo se ve involucrada la transmisión de información, será necesario tener presente las características anteriores durante el diseño e implementación de un Esquema de Replicación de Datos.

---

[1] El término compleción, no se encuentra dentro del diccionario, sin embargo se respeta el uso de este término ya que la interpretación del mismo se presenta en esta misma sección conforme a lo expuesto por James A. Senn en su libro titulado Análisis y Diseño de Sistemas de Información.

## Sistema de Información

En el sentido más amplio, un sistema es un conjunto de componentes que interactuan entre sí para lograr un objetivo común.

Una organización es un sistema. Sus componentes (mercadotecnia, manufactura, ventas, investigación, embarques, contabilidad y personal) trabajan juntos para crear utilidades que beneficien tanto a los empleados como a los accionistas de la compañía. Cada uno de estos componentes es a su vez un sistema. El departamento de contabilidad, por ejemplo, quizá esté formado por cuentas por pagar, cuentas por cobrar, facturación y auditoría entre otras.

Todo sistema organizacional depende, en mayor o menor medida, de una entidad abstracta denominada sistema de información. Este sistema es el medio por el cual los datos fluyen de una persona o departamento hacia otros y puede ser cualquier cosa, desde la comunicación interna entre los diferentes componentes de la organización hasta sistemas de cómputo que generan reportes para los usuarios. Los sistemas de información proporcionan servicio a todos los demás sistemas de una organización, y enlazan todos los componentes en forma tal que estos trabajen con eficacia para alcanzar el mismo objetivo.

De lo anterior, es importante destacar que "un sistema de información enlaza los componentes de la organización" y así es como la replicación de datos se vuelve parte del sistema de información dentro de una organización cualquiera ya que por medio de este se pone a disposición de sistemas mayores, la información que físicamente se encuentre lejana. Lo anterior se intenta ejemplificar mediante la Figura II.1. Replicación de Datos dentro de un Sistema de Información.

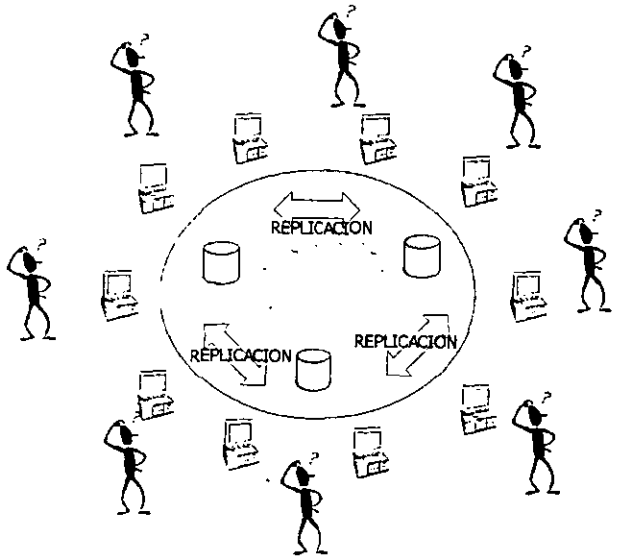



**Figura II.1. Replicación de Datos dentro de un Sistema de Información**

## Bases de Datos, nodos en un esquema de replicación.

Una base de datos es una de las partes fundamentales en un esquema de replicación. Es necesario subrayar que la replicación de datos es resultado de la necesidad de contar con información oportuna y confiable, obviamente, esta información es almacenada en alguna base de datos. Así, las bases de datos forman parte de un esquema de replicación, y podemos considerarlas como nodos dentro del mismo ya que serán fuentes o destinos de los datos replicados.

Debido a la importancia que una base de datos cobra dentro de un esquema de replicación y que además será necesario conocer algunos términos o conceptos propios de las bases de datos, se procede a dar un panorama general que nos ayude a comprender una base de datos en forma general, comenzando en primera instancia por el concepto mismo de base de datos.

### Bases de Datos

El concepto de base de datos ha sido definido por muchos autores y a continuación se presentan algunas de estas definiciones:

> "En esencia, un sistema de base de datos no es más que un sistema para archivar en una computadora." (C.J. Date)
>
> "Colección de datos interrelacionados almacenados en conjunto sin redundancias perjudiciales o innecesarias; su finalidad es servir a una aplicación o más de la mejor manera posible; los datos se almacenan de modo que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir nuevos datos y para modificar o extraer los datos almacenados" (Martin,1975)
>
> "Colección o depósito de datos donde los datos están lógicamente relacionados entre sí, tienen una definición y descripción comunes y están estructurados de una forma particular. Una base de datos es también un modelo del mundo real y, como tal, debe poder servir para toda una gama de usos y aplicaciones." (Conference des Statisticiens Europeens, 1977)
>
> "Conjunto de datos de la empresa memorizado por una computadora que es utilizado por numerosas personas y cuya organización es regida por un modelo de datos" (Flory, 1982)
>
> "Conjunto estructurado de datos registrados que son accesados por una computadora para satisfacer simultáneamente a varios usuarios de forma selectiva y en tiempo oportuno." (Delobel, 1982)
>
> "Colección no redundante de datos que son compartidos por diferentes sistemas de aplicación." (Howe, 1983)
>
> "Colección integrada y generalizada de datos, estructurada atendiendo a las relaciones naturales de modo que suministre todos los caminos de acceso necesarios a cada

unidad de datos con objeto de poder atender todas las necesidades de los diferentes usuarios." (Deen, 1985)

"Conjunto de ficheros maestros, organizados y administrados de una manera flexible de modo que los ficheros pueden ser fácilmente adaptados a nuevas tareas imprevisibles." (Frank, 1988)

"Colección de datos interrelacionados." (Elsmari y Navathe, 1989)

"Colección o depósito de datos integrados, almacenados en soporte secundario (no volátil) y con redundancia controlada. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de ellos, y su definición (estructura de la base de datos) única y almacenada junto con los datos, se ha de apoyar en un modelo de datos, el cual ha de permitir captar las interrelaciones y restricciones existentes en el mundo real. Los procedimientos de actualización y recuperación, comunes y bien determinados, facilitarán la seguridad del conjunto de datos." (Adoración de Miguel y Mario Piattini)

Ahora bien, debido a que el propósito de este trabajo no es propiamente el de bases de datos, sólo nos limitaremos a resaltar que la mayoría de las definiciones anteriores mencionan una "colección o conjunto de datos", y esta es la parte que nos interesa, ya que estos datos son la materia prima en un esquema de replicación. Sin olvidar todo aquello que interactua en el ambiente mismo de la base de datos.

## Modelos de Datos

El modelo sobre el cual este trabajo se basa, es el relacional. Lo anterior, como consecuencia directa de la popularidad del mismo ya que la gran mayoría de los productos actuales se orientan a este modelo. Sin embargo, dentro del ambiente de bases de datos existen diferentes modelos que pueden ser utilizados, mismos que se describen a continuación.

### Modelo Relacional

Una base de datos relacional es aquella cuyos usuarios la perciben como un conjunto de tablas.

El modelo relacional, es en la actualidad el más popular en los sistemas de manejo de una base de datos, puesto que conceptualmente es sencillo y comprensible por los profesionales de los sistemas de información y muchos otros usuarios finales.

Las bases de datos relacionales utilizan un modelo para mostrar como se relacionan lógicamente los datos en un registro.

El modelo relacional de datos, desarrollado en 1970 por E.F. Codd, se basa en una relación: una tabla bidimensional. Los renglones de la tabla representan los registros y las columnas muestran los atributos de la entidad.

El orden de los datos en la tabla no es significativo y tampoco implica un orden cuando los registros están incluidos en la relación.

### Modelo Jerárquico

Los primeros sistemas de bases datos, introducidos a mediados de los 60, estaban basados en el modelo jerárquico, el cual presume que todas las interrelaciones entre los datos pueden estructurarse como jerarquías. El modelo jerárquico relaciona entidades por medio de una relación superior-subordinado o padre-hijo.

Un modelo jerárquico de datos permite dos tipos de relación:

- Uno a uno.- Una entidad en un nivel se relaciona con una entidad en el siguiente nivel.
- Uno a muchos.- Una entidad en un nivel se relaciona con una, muchas o ninguna entidad del siguiente nivel.

### Modelo de Red

Las redes constituyen una manera natural de representar las interrelaciones entre los objetos. Se utilizan ampliamente en las matemáticas, la investigación operativa, la química, la física, la sociología, y otros campos. Como los objetos y sus interrelaciones constituyen maneras útiles de modelar muchos de los fenómenos que nos conciernen en los negocios, no es sorprendente que la arquitectura de retículos se aplique también a la organización de las bases de datos.

Generalmente, las redes se pueden representar mediante una estructura matemática llamada grafo orientado. Los grafos orientados son una estructura simple, se construye con puntos o nodos conectados por arcos orientados o aristas. Dentro del marco de los aristas, los nodos pueden considerarse como tipos de registros de los datos y las aristas pueden considerarse como la representación de las interrelaciones uno-uno o uno-muchos. Así, el modelo de datos en red representa los datos en estructuras de retículos, de los tipos de registros conectados por interrelaciones uno-uno o uno-muchos. La estructura del grafo facilita la representación simple de las interrelaciones jerárquicas (como los datos genealógicos), las interrelaciones de pertenencia (como departamento al que se asigna un empleado) y muchas otras. Además, una vez que se ha establecido una interrelación entre dos objetos, la recuperación y la manipulación de los datos asociados puede realizarse eficientemente.

El modelo de red es análogo al modelo jerárquico, excepto que una entidad puede tener más de un padre. Así, como se muestra la Figura II.2. Estructura básica del modelo de datos de red., los miembros pueden pertenecer a más de una relación. Esta capacidad introduce el uso de un tipo adicional de relación entre los datos:

- Muchos a muchos.- Una entidad se puede relacionar con una, muchas o con ninguna entidad en otro nivel.

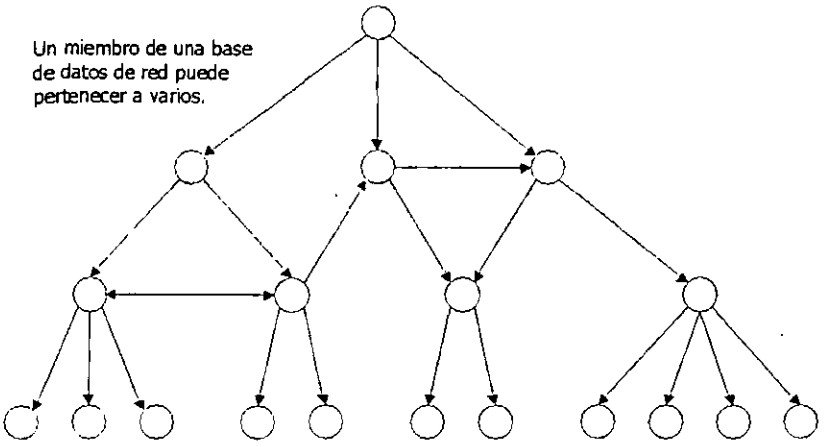



**Figura II.2. Estructura básica del modelo de datos de red.**

En las bases de datos de tipo red, así como en las jerárquicas, se deben establecer las relaciones entre las entidades al mismo tiempo que se establece el modelo de los datos y se crea la base de datos.

Las bases de datos jerárquicas y de red son conceptualmente sencillas y parecen no ser complicadas a primera vista. Sin embargo, en un ambiente de una base de datos grande, puede evolucionar rápidamente hacia una telaraña complicada de interrelaciones que son difíciles de manejar al evolucionar la base de datos durante su uso.

## Componentes de una Base de Datos

Si bien una base de datos se considera como una sola entidad, es necesario recordar que esta es una "colección o conjunto de datos"[2], así que esta colección puede ser compuesta por diferentes elementos que son almacenados dentro de la misma base de datos. Entre los elementos más comunes que se encuentran en la mayoría de las bases de datos relacionales son: tablas, vistas, procedimientos almacenados y triggers entre otros.

### Tabla.

Una tabla base es una tabla "real"; es decir, una tabla con existencia física, en el sentido de que existen registros almacenados físicamente, y quizá también índices físicos, los cuales representan en forma directa esa tabla almacenada.

Una tabla en un sistema relacional se compone de una fila de cabeceras de columnas junto con cero o más filas de valores de datos.

[2] Veáse las definiciones de base de datos en la página 8.

Para una tabla dada:

a) La fila de cabeceras de columnas especifica una o más columnas (dando, entre otras cosas un tipo de datos para cada una)

b) Cada fila de datos contiene un solo valor escalar para cada una de las columnas especificadas en la fila de cabeceras de columna. Además, todos los valores de una columna dada tienen el mismo tipo de datos, a saber el tipo especificado en la fila de cabeceras de columna para esa columna.

### Vista.

Una vista es una tabla virtual, es decir, una tabla que no existe en sí pero que ante el usuario parece existir. Las vistas no se sustentan en sus propios datos almacenados, separados físicamente y distinguibles. En vez de ello, se almacena su definición en el catálogo en términos de otras tablas.

### Procedimientos Almacenados

Los procedimientos almacenados (*stored procedures*) como su nombre lo indica, son procedimientos cuyo código se almacena directamente en la base de datos, de tal forma que operan como funciones o subrutinas a las cuales se les envían parámetros mismos que serán utilizados internamente para cumplir sus objetivos.

### Triggers

Un *trigger*[3] es muy similar a un procedimiento almacenado ya que consiste en una rutina cuyo código se almacena directamente en la base de datos, pero, a diferencia de los procedimientos almacenados, estos no son invocados directamente por usuarios o programador alguno, sino más bien, son ejecutados en respuesta a eventos específicos como lo son la inserción, actualización y borrado de registros en las tablas existentes. Así, generalmente solo existen tres posibles tipos de triggers que pueden relacionarse a una tabla específica, estos son: de inserción, actualización y borrado.

## Structured Query Language (SQL)

Una vez que se han explicado los componentes de una base de datos, es necesario mostrar el lenguaje que se utiliza para crear, actualizar, consultar e inclusive eliminar estos objetos. Este lenguaje se conoce como SQL (Structured Query Language), el cual es un lenguaje relacional que al mismo tiempo es un lenguaje de consulta interactiva y un lenguaje de programación de bases de datos

[3] Para este término se ha prefendo seguirlo utilizando en inglés por ser conocido de esa forma en los ambientes de bases de datos.

El lenguaje SQL surge originariamente con el nombre de SEQUEL (Structured English Query Language) implementado en un prototipo de IBM, el SEQUEL-XRM, durante los años 1974-1975. Este prototipo evolucionó durante los años 1976-1977, pasándose a denominar su lenguaje SEQUEL/2, y cambiando posteriormente este nombre por SQL, debido a motivos legales.

En 1979 aparece el primer RDBMS (Relational Database Management System) comercial basado en SQL (ORACLE), posteriormente van surgiendo otros productos basados en SQL como son el SQL/DS, DB2, DG/SQL, Sybase, Interbase, Informix, Unify, etc. E incluso otros productos que no poseían SQL como lenguaje base (Ingres, Datacom, Adabas, Supra, IDMS/R) ofrecen interfaces SQL; por lo que este lenguaje se convierte en un estándar de facto, aunque con múltiples variantes según los distintos fabricantes.

En 1982 el Comité de bases de datos X3H2 de ANSI presenta un lenguaje relacional estándar basado principalmente en el SQL propio de los productos IBM; en 1986 este organismo aprueba el lenguaje como norma pasando a denominarse SQL/ANS, que también es aprobado, al año siguiente, como norma ISO (ISO-1987). En esta norma se especifican dos niveles (I y II) a cumplir, siendo el nivel I un subconjunto de las funcionalidades proporcionadas por el nivel II.

En 1989 se revisa la versión 1 del estándar (ISO-1989), revisión conocida como Addendum, que añade cierta integridad referencial, que se denomina integridad referencial básica, ya que sólo permite definir la opción de modificación y borrado restringidos y no proporciona cambios en cascada. Por otra parte, ya que la norma ISO de 1989 no estandariza las definiciones para SQL embedido en lenguajes de programación, ANSI define, este mismo año, un estándar para el lenguaje SQL embedido (ANSI-1989).

En junio de 1990, IBM anuncia su estándar DRDA (Distributed Relational Database Access) como parte de su arquitectura SAA (System Application Architecture).

En abril de 1991 el SAG (SQL Access Group) completa la Fase I de especificaciones técnicas, que define un estándar para intercambiar mensajes SQL sobre una red OSI, basado en la especialización SQL del RDA de ISO. En junio de 1991 este grupo realizó una demostración con más de veinte RDBMS que intercambiaban datos y consultas. En el mes de este mismo año Microsoft anunció ODBC (Open Database Connectivity) basado en el estándar del SAG.

En 1992 este grupo completó su segunda fase, que especificaba un IPA (Interfaz para la Programación de Aplicaciones) CLI (Call-Level Interface) y que ampliaba el estándar a más instalaciones cliente/servidor, en las que además de las especificaciones OSI se incluye otros protocolos de red como, por ejemplo, TCP/IP. En noviembre de ese año Borland relanzó el estándar ODAPI (Open Database Application Programming Interface) como IDAPI (Integrated Database Application Programming Interface), con el patrocinio de IBM, Novell y WordPerfect.

En 1992 también se aprueba como norma internacional una nueva versión del SQL, conocida como SQL2 o SQL-92 (ISO-1992), en la que se incrementa substancialmente la capacidad semántica del esquema relacional, se añaden nuevos operadores, se mejora el tratamiento de errores y se incluyen normas para el SQL embedido. En esta nueva norma se definen tres niveles de conformidad distintos: Entry SQL, Intermediate SQL y Full SQL, siendo este último el que ofrece

mayores funcionalidades, mientras que el primer nivel es un subconjunto del segundo y éste, a su vez, del tercero.

Este estándar ha sido complementado recientemente con dos nuevas partes que abordan la interfaz de nivel de llamadas (Call-Level Interface), ISO (1995) y la definición de módulos almacenados persistentes (Persistent Stored Modules) ISO-1996. Esta norma internacional convierte al SQL en un lenguaje computacionalmente completo añadiéndole estructuras de control, manejo de excepciones, etc.

En la actualidad se acaban de aprobar nuevas propuestas para extender el SQL (el llamado SQL3), dotándolo, además de una mayor capacidad semántica, de ciertos principios del paradigma de la orientación al objeto. También se está trabajando sobre una extensión del SQL3 para soportar bases de datos multimedia, conocida como SQL/MM (SQL Multimedia). Debido al tamaño que posee el SQL3 ya se ha empezado a hablar del SQL4, en el que se incluyen aquellas característica del lenguaje que no se encuentran todavía definidas completamente o que necesitan mayor profundidad.

A continuación se presentan algunas sentencias de este lenguaje relacional.

### CREATE TABLE

Este comando de SQL permite como su nombre lo indica crear tablas dentro de una base de datos, la sintaxis genérica es la siguiente:

```
CREATE TABLE nombre_tabla
(
  nombre_columna tipo_de_datos {null|not null},
  nombre_columna tipo_de_datos {null|not null},
  ...
  PRIMARY KEY( llave_primaria )
)
```

Por ejemplo, el siguiente comando crea la tabla llamada clientes con una columna llamada id del tipo integer y una columna llamada nombre del tipo varchar(30), siendo la llave primaria de la tabla la columna id.

```
CREATE TABLE clientes
(
  id       integer          not null,
  nombre   varchar(30)      null,
  primary key (id)
)
```

Como se puede observar en el comando anterior, se describe totalmente la tabla que esta siendo creada, de tal forma que la estructura de la tabla necesariamente contendrá todas y cada una de las características que en este comando se presentan. Cuando se trabaja con bases de datos reales, generalmente se utilizan varias decenas de tablas, lo cual implica contar con los comandos necesarios para crear todas y cada una de estas tablas. Se resalta esto ya que para realizar la

implementación de un esquema de replicación por medio del Replication Server, es necesario contar con la descripción, tipo de dato y logitud de cada una de las columnas que conforman una tabla. Por lo que, se aprovechará el hecho de que dicha estructura se almacena en la propia base de datos para de esta forma reducir el trabajo a realizar durante la implementación de un esquema de replicación[4].

### CREATE PROCEDURE

Este comando tiene como función el crear un procedimiento almacenado, el cual queda a disposición de los usuarios para ser utilizado. La sintaxis general es:

```
create procedure nombre_procedimiento [[(]@parameter_name datatype
[( length) | ( precision [, scale])][= default][output]
[, @ parameter_name
datatype [( length) | ( precision [, scale])]
[= default][output]]...[)]]
[with recompile]
as { SQL_statements | external name dll_name}
```

### SELECT

Esta instrucción es la base del SQL, y es por medio de la cual se realizan consultas a las tablas de una base de datos. La sintaxis general de esta sentencia es:

```
SELECT {*,lista_columnas}
  FROM tabla1, tabla2, ...
 WHERE condicion1 {and/or} condicion2 ...
```

Por ejemplo, para consultar la información de la tabla que se creo con la instrucción CREATE TABLE anteriormente, con la siguiente instrucción observaríamos dicha información:

```
SELECT id, nombre FROM clientes
```

O también podemos utilizar la siguiente:

```
SELECT * FROM clientes
```

En la cual el * representa todas las columnas de una tabla. O bien si fuera necesario presentar los datos de una sola fila podríamos utilizar:

```
SELECT * FROM clientes WHERE id=1
```

### INSERT

La instrucción INSERT permite insertar registros en una tabla especifica. La sintaxis general de esta instrucción es:

---

[4] Véase Bases del Generador de Scripts de Replication Server (GSRS). en la página 81.

```
INSERT INTO nombre_tabla [(columna1, columna2, ...)]
                    VALUES (valor1,valor2,...)
```

De tal forma que la inserción a la tabla clientes se realizaría con una sentencia como la siguiente:

```
INSERT INTO clientes (id,nombre) VALUES (1,'José Pérez')
```

O también mediante la siguiente:

```
INSERT INTO clientes VALUES (1,'José Pérez')
```

### UPDATE

UPDATE es la instrucción por medio de la cual se realizan cambios dentro de los campos de una fila o grupo de filas específicas. La sintaxis para esta sentencia se presenta a continuación:

```
UPDATE nombre_tabla
SET campo1=valor_nuevo[,.
    campo2=valor_nuevo,...]
[WHERE condicion1 [and|or] condicion2 ...]
```

Así, es posible actualizar datos de la columna de una fila específica mediante una sentencia similar a la siguiente:

```
UPDATE clientes
SET nombre='Juan Pérez'
WHERE id=1
```

O si fuera necesario actualizar todas las filas de una tabla, entonces con omitir la cláusula WHERE para que la actualización no tenga restricción alguna.

### DELETE

Esta instrucción como su nombre lo indica tiene la finalidad de eliminar registros de una tabla. La sintaxis para esta sentencia.es:

```
DELETE FROM nombre_tabla
[WHERE condicion1 [and|or] condicion2 ...]
```

De tal formá, para eliminar todos los registros de una tabla (clientes para este caso), será necesario ejecutar la siguiente sentencia:

```
DELETE FROM clientes
```

O podemos utilizar:

```
DELETE FROM clientes WHERE id=1
```

Si fuera necesario borrar solo unos cuantas filas de la tabla.

Las sentencias INSERT, UPDATE y DELETE son las sentencias básicas para el mantenimiento de las tablas, es decir, ALTAS, CAMBIOS y BAJAS respectivamente, sin embargo cada una de estás sentencias por se consideran como la unidad básica de trabajo, para ello existen lo que se conoce como transacción siendo está el elemento mínimo que puede manipular Replication Server, y la cual se describe a continuación.

## Transacción

Los tres últimos comandos se ha sido descrito anteriormente (INSERT, UPDATE y DELETE), son los con los cuales se inserta, actualiza o elimina información dentro de las tablas de una base de datos, pero a pesar de ser estos las instrucciones más elementales que existen para el mantenimiento de los datos, no son estos la unidad básica de trabajo dentro de una base de datos. La unidad básica y elemental dentro de las bases de datos es la **transacción**.

Una transacción es una unidad lógica de trabajo que puede definirse como "secuencias de operaciones que han de ejecutarse de forma atómica", es decir, o bien se realizan todas las operaciones que comprende la transacción globalmente o no se realiza ninguna. Puede considerarse a una transacción cualquier suceso o actividad que afecta a una organización. Una de las transacciones más comunes son: facturación, entrega de mercancía, pago a empleados y depósito de cheques. Los tipos de transacciones cambian en cada una de las diferentes organizaciones. Sin embargo, la mayor parte de las compañías procesan dichas transacciones como una mayor parte de sus actividades cotidianas. Véase Figura II.3 Sistema de Procesamiento de Transacciones, en dónde se muestra que las transacciones son la base de la operación de cualquier organización.

Como ejemplo de transacción, se puede mencionar la teoría de la partida doble dentro contabilidad, ya que esta puede identificarse perfectamente con el término transacción, ya que a todo cargo corresponde un abono y por lo mismo solo se considerará como correcta si se concluyen ambos movimientos.


**Figura II.3 Sistema de Procesamiento de Transacciones**

El procesamiento de transacciones garantizará que si la transacción ejecuta algunas modificaciones y después se presenta una falla antes de que llegue al término normal de la transacción, se anularán esas modificaciones. De esta manera puede lograrse que una secuencia de operaciones, la cual en esencia no es atómica, aparente serlo desde un punto de vista externo.

El componente del sistema encargado de lograr esta atomicidad (o apariencia de atomicidad) se conoce como *manejador de transacciones*, y las operaciones COMMIT (comprometer) y ROLLBACK (retroceder) son la clave de su funcionamiento:

- La operación COMMIT señala el término exitoso de la transacción: Le dice al manejador de transacciones que se ha finalizado con éxito una unidad lógica de trabajo, que la base de datos está (o debería estar) de nuevo en un estado consistente, y que se pueden "comprometer", o hacer permanentes, todas las modificaciones efectuadas por esa unidad de trabajo.

- La operación ROLLBACK, en cambio, señala el término no exitoso de la transacción: dice al manejador de transacciones que algo salió mal, que la base de datos podría estar en un estado inconsistente y que todas las modificaciones efectuadas hasta el momento por la unidad lógica de trabajo deben "retroceder" o anularse.

Es posible anular las modificaciones que se vayan realizando en una transacción dado que el sistema mantiene una *bitácora* o *diario* en algún almacenamiento secundario, en el cual se registran los detalles de todas las operaciones de actualización, en particular, los valores inicial y final del objeto modificado. Por tanto, si resulta necesario anular alguna modificación específica, el sistema puede utilizar la entrada correspondiente de la bitácora para restaurar el valor original del objeto modificado.

Así, podemos percatarnos que las transacciones no sólo son la unidad lógica de trabajo sino también la unidad de *recuperación*. Si una transacción se compromete con éxito, el sistema deberá

garantizar el establecimiento permanente de sus modificaciones en la base de datos, aún si el sistema se cayera en el instante siguiente. Si esto sucediera, el procedimiento de reinicio del sistema instalará de todas maneras esas modificaciones en la base de datos; podrá descubrir los valores que se deben grabar examinando las entradas pertinentes de la bitácora. Así el procedimiento de reinicio recuperará todas las unidades de trabajo (transacciones) completadas con éxito pero cuyas modificaciones no lograron grabarse físicamente antes de la caída; por tanto, como ya se mencionó anteriormente, as transacciones son también una unidad de recuperación.

La mayor parte de los sistemas de bases de datos son sistemas para múltiples usuarios; es decir, son sistemas en los cuales se permite a cualquier cantidad de transacciones tener acceso a la misma base de datos al mismo tiempo. En sistemas como estos, se necesita algún tipo de *mecanismo de control de concurrencia* a fin de asegurar que ninguna transacción concurrente interfiera con las transacciones de las demás.

En esencia, son tres los errores que pueden presentarse, es decir, tres situaciones en las cuales una transacción aunque correcta en sí, puede producir de todos modos un resultado incorrecto, debido a una interferencia por parte de alguna otra transacción. Estos problemas son:

1. El problema de la modificación pérdida.
2. El problema de la dependencia no comprometida.
3. El problema del análisis inconsistente.

Por todo lo anterior y considerando que por definición una base de datos se encuentra en un estado consistente antes de que empiece a ejecutar una transacción, y también lo deberá estar cuando la transacción termine de ejecutarse. Podemos decir que las propiedades principales que debe poseer una transacción son las siguientes:

- **Atomicidad**: Se ejecutan todas las sentencias de una transacción o ninguna.
- **Preservación de la consistencia**: la ejecución de una transacción debe dejar a la base de datos en un estado consistente.
- **Aislamiento**: Una transacción no muestra los cambios que produce hasta que finaliza.
- **Persistencia**: Una vez que la transacción finaliza con éxito, sus efectos perduran en la base de datos.

## Log de Transacciones

Para conseguir anular y recuperar transacciones, el método más extendido suele ser la utilización de un archivo denominado diario (*log, journal*), más comúnmente conocido como *log de transacciones*, en el que se va guardando toda la información necesaria para deshacer (en caso de alguna falla) o rehacer (si hay que recuperar) las transacciones.

Un registro del log de transacciones suele constar de:

- Identificación de la transacción.
- Hora de la modificación.
- Identificador del registro afectado.
- Tipo de la acción.
- Valor anterior del registro.
- Nuevo valor del registro.
- Información adicional (por ejemplo, un puntero al registro previo del diario que concierne a la misma transacción).

Al irse almacenando todos los cambios en esta bitácora puede surgir un problema en caso de que se realice un cambio en la base de datos y no en el log mismo, debido a alguna falla del equipo; por ello, normalmente se obliga a que los registros que se modifican y se encuentran en memorias de área intermedia o memoria principal se escriban antes en el log de transacciones que en la misma base de datos, para poder anular así las transacciones si llega a ser necesario. Esto se conoce como *log write-ahead protocol* (protocolo de bitácora de escritura adelantada).

El archivo log puede ser un archivo circular, es decir, que una vez lleno va eliminando registros según van entrando otros nuevos, aunque lo normal es que conste de dos partes; la primera *en línea* (en disco), que almacena las actualizaciones que se llevan a cabo hasta que se llena, momento en el que se pasa el contenido a la segunda parte (por ejemplo, en cinta).

Para evitar tener que recorrer todo el archivo diario, lo cual consumiría mucho tiempo, se introduce el concepto de *punto de verificación* o *punto de recuperación* (*checkpoint*), que se ejecuta periódicamente y que implica:

- Pasar el contenido de las memorias de área intermedia al archivo diario (al igual que para la base de datos, para el archivo diario existen unas áreas intermedias dónde se guardan registros de este fichero).
- Escribir un registro de punto de recuperación en el diario.
- Pasar el contenido de las memorias de área intermedia de la base de datos a soporte secundario.
- Escribir la dirección del registro de recuperación en un archivo de arranque.

Recientemente se han propuesto nuevas técnicas, entre las que destaca la del *archivo diario efímero* (*ephemeral logging*), que no requiere puntos de verificación ni aborta transacciones muy largas (como sucede con el fichero diario clásico). En esta técnica se gestiona el archivo diario

como una cadena de colas a las que se va añadiendo registros, llevándose a cabo, de forma automática, la *recolección de basura* (*garbage collection*) y la compresión del archivo.

También existen otras formas de garantizar la recuperación ante fallos sin emplear un archivo diario, como es la *técnica de páginas ocultas* (*shadow paging*), que consiste en mantener dos tablas o directorios de páginas durante la vida de una transacción. Al empezar la transacción ambas tablas son iguales, reflejándose todos los cambios en una sola de ellas (la primaria) y manteniendo la otra (secundaria) sin cambios. En caso de que la transacción se grabe, se deshecha la página secundaria y la primaria se convierte en la actual; si la transacción aborta, se deshecha la primaria y se restablece la secundaria. Esta técnica presenta ventajas respecto al archivo diario, ya que la recuperación es más rápida, pero necesita *recolección de basura* para reclamar bloques inaccesibles.

La importancia del log de transacciones como se verá más adelante radica en el hecho que Replication Server recorre este archivo para obtener o leer las transacciones que deba replicar.

## Manejadores de Bases de Datos

Si bien ya se han descrito los componentes de una base de datos, el lenguaje que se utiliza comúnmente en estas y sobre todo el concepto mismo de lo que es una base de datos, es necesario aclarar que todo lo anterior se lleva a la práctica por medio de una pieza de software a la cual identificaremos como Manejador de Base de Datos o DBMS por sus siglas en inglés derivadas de Data Base Managment System. Un Manejador de Base de Datos se puede definir de las siguientes formas:

"Un conjunto coordinado de programas, procedimientos, lenguajes, etc., que suministra a los distintos tipos de usuarios los medios necesarios para describir y manipular los datos almacenados en la base, garantizando su seguridad"[5]

El DBMS es un nivel de programas entre la base de datos física (es decir, los datos tal y como están almacenados en realidad) y los usuarios del sistema."

Una de las funciones generales del DBMS es "distanciar a los usuarios de la base de datos de detalles del nivel de equipo".

En general, pues, la información en la base de datos por lo menos en sistemas grandes, estará integrada y además será compartida.

Un DBMS proporciona la flexibilidad en el almacenamiento y recuperación de datos, así como en la producción de información. Este permite la independencia de los datos lo cual significa que las aplicaciones pueden cambiar sin afectar datos almacenados.

El DBMS es un software, parecido a un sistema operativo o a un compilador, que brinda un conjunto de servicios a los usuarios finales, programadores y otros. El DBMS existe para facilitar la

[5] Fuente: Fundamentos y Modelos de Bases de Datos, Adoración de Miguel y Mario Piattini.

gestión de una base de datos. Con este fin, un DBMS típicamente brinda la mayoría de los servicios:

- Herramienta para la definición y el control centralizados de los datos, conocida como diccionario de datos/directorio (DD/D) o catálogo.
- Mecanismos de seguridad e integridad de los datos.
- Acceso concurrente a los datos para varios usuarios.
- Utilidades para la consulta, manipulación y la elaboración de informes orientados al usuario.
- Utilidades para el desarrollo de sistemas de aplicación orientados al programador.

En la Figura II.4. Componentes de un DBMS., se muestran los elementos de un DBMS y su interrelación con los diferentes usuarios.

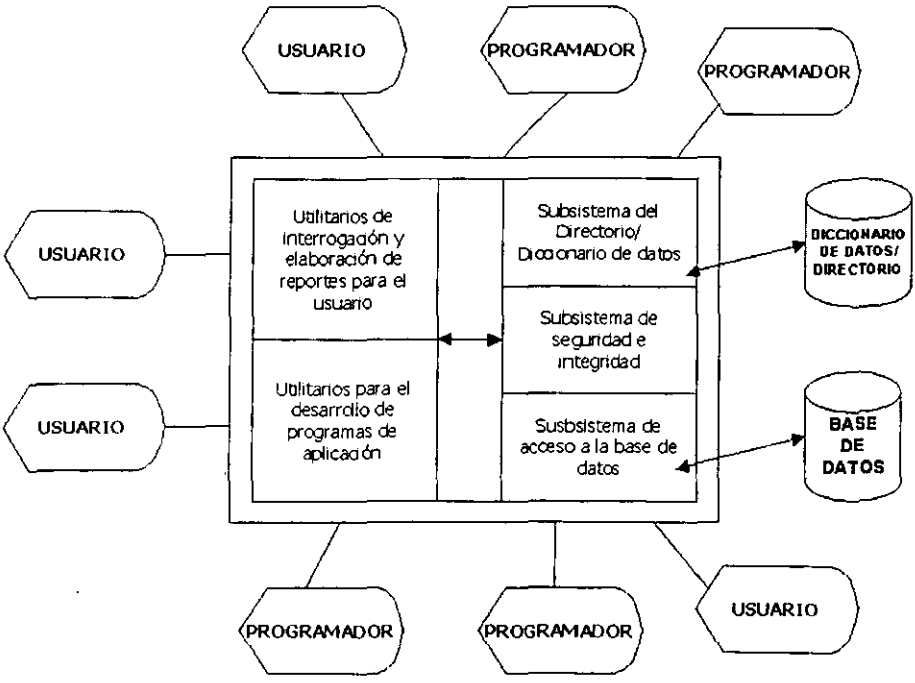



**Figura II.4. Componentes de un DBMS.**

La importancia de un DBMS radica en que realmente nunca se accesa a la base de datos directamente sino más bien se trabaja vía el DBMS, mismo que recibe las instrucciones y las aplica.

Para el ambiente bajo el cual se basa el presente es utilizado Adaptive Server Enterprise (comúnmente conocido como Sybase) como DBMS.

## Bases de datos distribuidas

Con la amplia prevalencia de los sistemas de bases de datos centralizados, los usuarios y los programas aplicación acceden a la base de datos desde sitios locales, así como desde localidades remotas. En contraste, una base de datos distribuida (distributed database) no se almacena completamente en una localidad central, sino que se distribuye en una red de localidades que pueden estar geográficamente separadas y conectadas por enlaces de comunicaciones. Cada localidad tendrá su propia base datos y está capacitada para acceder a los datos de otras localidades.

Hay varias razones para desarrollar y usar un sistema de bases datos distribuida entre las que se incluyen las siguientes:

1. A menudo, las organizaciones tienen ramales o divisiones en diferentes localidades. Para una localidad dada, $L$, pudiera haber un conjunto de datos que use frecuentemente y quizás exclusivamente. Además $L$ puede usar datos que se usen con más frecuencia en otra localidad $L'$.

   En un sistema de base de datos centralizado cada sitio tendría que estar usando enlaces de comunicación con la base datos para ambos tipos de información por lo que las comunicaciones pueden convertirse en un cuello de botella importante.

2. Permitir que cada sitio almacene y mantenga su propia base de datos facilita el acceso inmediato y eficaz a los datos que se usan con más frecuencia. Tales datos podrían usarse en otros sitios, pero con menos frecuencia. Datos similares almacenados en otras localidades también podrían accederse si fuera necesario.

3. Las bases de datos distribuidas pueden mejorar la fiabilidad. Si las computadoras de un sitio fallan, o queda fuera de servicio algún enlace de comunicaciones, el resto de la red puede seguir funcionando. Es más, si los datos están duplicados en más de un sitio, los datos necesarios pueden estar disponibles en otro lugar que aún se mantiene operando.

4. Permitir el control local de los datos que se usan con más frecuencia en un sitio puede mejorar el grado de satisfacción de los usuarios con relación al sistema de base de datos. Esto es las bases de datos locales pueden reflejar mejor la estructura administrativa local y, por lo tanto, brindar un mejor servicio a las necesidades de la gestión.

Estas son algunas de las ventajas que se obtienen comúnmente de los sistemas de bases de datos distribuidas. Sin embargo, hay circunstancias en las que un sistema distribuido puede no presentar ventajas. En situaciones donde tenga lugar una gran cantidad de comunicación entre los sitios, el costo adicional en que se puede incurrir por las coordinaciones y las tareas de control puede degradar severamente el rendimiento. Esto puede ser especialmente cierto cuando se mantienen

duplicados de los datos en sitios diferentes y, por consiguiente, se necesitan recursos extras para asegurar que las actualizaciones que se hagan de modo concurrente sean consistentes.

Lo anterior se basa en el concepto 2PC, mismo que será descrito más adelante, sin embargo, la ventaja de replicar los datos, es la velocidad que se gana en tiempo de procesamiento en los sitios en donde se mantengan dichos duplicados, así como la posibilidad de disponer copias de seguridad por si ocurre alguna falla en alguna localidad.

El diseño de un sistema de base de datos distribuida puede resultar una tarea compleja. Se deben hacer consideraciones muy cuidadosas sobre los objetivos y las estrategias que deben servir al diseño y se deben, en paralelo, tomar decisiones sobre cómo hay que distribuir los datos entre los sitios de la red.

Algunas estrategias y objetivos que podemos considerar en la mayoría de las implementaciones de los sistemas de bases datos distribuidas, son las siguientes:

**Transparencia de la ubicación.-** La transparencia de la ubicación permite a un usuario acceder a los datos sin conocer, o tener que ver con los sitios en los que residen los mismos. Las ubicaciones de los datos están ocultas al usuario.

**Transparencia de duplicación.-** La transparencia de duplicación de los datos significa que si existe más de una copia de los datos, una sola copia se debe escoger cuando se van a recuperar datos y todas las copias se deben a actualizar cuando se hagan cambios.

**Independencia de la configuración.-** La independencia de la configuración permitir a la organización añadir o reemplazar hardware sin tener que estar cambiando los componentes de software existentes en el DBMS. La independencia de la configuración significa tener un sistema que puede expandirse cuando el hardware se sature.

**DBMSs no homogéneos.-** A veces es deseable integrar bases de datos mantenidas por diferentes DBMSs sobre computadoras diferentes. A menudo los DBMSs son suministrados por diferentes fabricantes y pueden soportar diversos modelos de datos. Un enfoque para integrar estas bases datos es proporcionar una única interfaz de usuario que pueda ser usada para acceder a los datos mantenidos por los DBMSs no homogéneos. Los diferentes modelos de datos soportados por los DBMSs no homogéneos se le ocultan al usuario mediante esta única y amplia interfaz.

**Duplicación de datos.-** La duplicación de datos ocurre si el sistema mantiene en varias copias idénticas de una relación, *R*, con cada copia almacenada en un sitio diferente. Típicamente la duplicación se introduce para aumentar la disponibilidad del sistema: cuando una copia no esté disponible debido a una falla de un sitio, sería posible tener acceso a otra copia.

La duplicación también puede mejorar el rendimiento, puesto que las transacciones tienen mayor posibilidad de encontrar una copia localmente. El inconveniente está en el costo extra del almacenamiento adicional y en el mantenimiento de la consistencia mutua entre las copias. La

actualización de una copia local impone el sobrecosto de transmitir la actualización a todas las copias. Y es en esta transmisión y actualización en dónde Replication Server puede ayudarnos.

Formalmente las ventajas de la duplicación están en:

1. Si una de las estaciones que contiene una relación *R* falla, la relación puede recuperarse de otro sitio y el sistema puede continuar cualquier procesamiento que involucre a *R*. Por tanto, se mejora la disponibilidad de la base de datos.

2. Si la mayoría de los accesos a *R* sólo involucran una lectura de la relación, entonces es posible que varios sitios que estén procesado estas consultas concurrentemente, así mientras más copias haya de *R* a lo largo de la red, mayor es la posibilidad de que una consulta puede ejecutarse sin requerir de transmisión de datos entre las estaciones. Lo que significa la ganancia en los costos y en tiempo.

La desventaja principal es que el sistema debe asegurar que son idénticas todas las copias de *R*. Por tanto, cuando ocurra una actualización de *R*, tal actualización deberá propagarse a todas las localidades con el correspondiente aumento del costo.

Para las bases de datos con duplicación hay al menos dos alternativas posibles de implementación. En una de éstas, se mantiene una base de datos centralizada de la cual se extraen copias de porciones para su uso local. Esta redundancia se ve compensada por la reducción de los costos de comunicaciones debido a que los datos están almacenados localmente. La fiabilidad también se mejora, puesto que la pérdida de los datos en una localidad puede restaurarse por la copia de los datos que se mantiene que en otra localidad.

La segunda alternativa es omitir la base de datos centralizada y duplicar segmentos de la base de datos en los sitios que tienen los usuarios más frecuentes. Con esto se evitan los costos de mantener la base de datos centralizada, pero pueden crecer los costos en comunicaciones.

### La replicación de datos permite bases de datos distribuidas

De lo anterior, se infiere que la replicación permite implementar una base de datos distribuida, pero es necesario tener presente que esta distribución se basa en la duplicación asíncrona, por lo menos para los fines del presente. Aunque es necesario resaltar que por medio del 2PC es posible lograr una duplicidad síncrona de estos datos tal y como se explicará más adelante.

Además es necesario resaltar la diferencia entre datos distribuidos y datos replicados.

Lo datos distribuidos son datos que están particionados en fragmentos, de los cuales cada fragmento se localiza en un sitio en particular dentro del sistema distribuido y cada fragmento existe en un solo sitio, por lo tanto, no existen copias de los datos.

Por otro lado, los datos replicados implican la copia de los mismos a través de todo el sistema, por lo cual, existen múltiples copias del mismo dato dentro del entorno del sistema.

## Replicación de datos

El término replicación no existe en el diccionario de la lengua española con un significado que vaya acorde al contexto del presente, sin embargo, este término hace referencia al hecho o resultado de duplicar. En pocas palabras, el término puede ser utilizado como sinónimo de copiar, de tal forma, que la replicación de datos podemos identificarla también como la copia de los mismos en un sentido amplio.

Además, este término se identifica dentro de la literatura técnica como un proceso automático mismo que replica tablas relacionales entre diversas bases de datos, de tal forma que la información entre las tablas permanezca consistente de manera síncrona, aunque, en los últimos años se ha considerado este término de manera más amplia para identificar inclusive la propagación asíncrona de estos datos.

### Tipos de Replicación

Anteriormente se han mencionado implícitamente los tipos de replicación existentes, estos tipos corresponden a la replicación síncrona y asíncrona, mismos que se describen a continuación.

#### Síncrona

La replicación síncrona de datos permite implementar lo que se puede llamar "alta consistencia" entre los diferentes almacenes de datos. Esto significa que la latencia[6] existente entre la consistencia de los datos es prácticamente cero. Los datos a lo largo de todos los sitios replicados son siempre los mismos, no importa de que sitio se origine la actualización. Lo anterior se puede lograr mediante la aplicación misma o mediante triggers[7] en las tablas aunque lo más común para implementar este tipo de replicación es mediante el protocolo de compromiso de dos fases, conodico como *two-phase commit* (2PC). En la Figura II.5. Protocolo de compromiso de dos fases (two phase commit – 2PC) se muestra gráficamente el protocolo 2PC.

Este tipo de replicación ofrece los siguientes beneficios:

- Considera múltiples operaciones como una sola operación lógica.
- Garantiza la absoluta consistencia entre los servidores de aquellos datos que se ven afectados en la transacción.
- Proporciona total transparencia a las operaciones que soporta, dependiendo del diseño de los programas.
- Se pueden replicar transacciones entre múltiples bases de datos.

[6] Considerar al término latencia como el tiempo transcurrido desde que se realizó el cambió en base de datos origen y hasta que dicha actualización es aplicada en las bases de datos remotas.
[7] Veáse Triggers en la página 11.

Sin embargo, existen también inconvenientes que deben ser considerados, estos pueden ser:

- Todas las transacciones fallarán si cualquier componente del sistema no está disponible.
- Se presentan una carga adicional por los mensajes para preparar y comprometer la transacción.
- La carga adicional ocasiona que este tipo replicación sean la parte más lenta dentro del sistema.
- Cada nodo que se agregue incrementa la posibilidad de falla.

La implementación del 2PC debe ser codificada dentro de la aplicación.

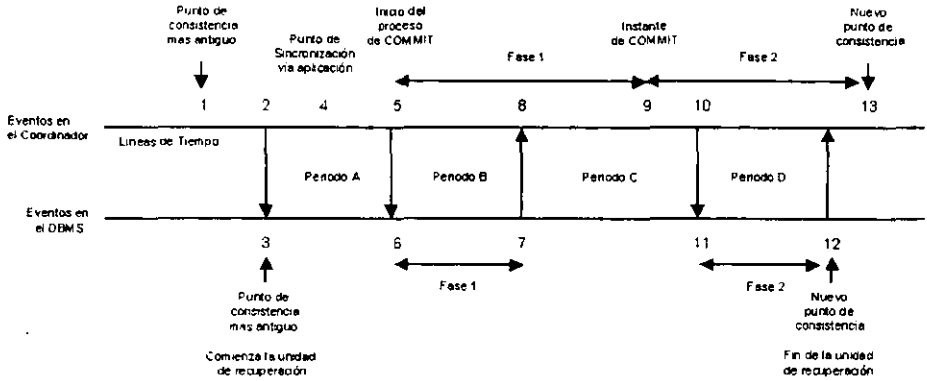



**Figura II.5. Protocolo de compromiso de dos fases (two phase commit – 2PC)**

Sin embargo, la replicación a la cual este trabajo se orienta no es la replicación síncrona sino más bien la replicación asíncrona que se describirá a continuación.

### Asíncrona

La replicación asíncrona de datos provee "baja consistencia" entre los diferentes sitios de almacenamiento. Esto significa que la latencia existente entre la consistencia de los datos siempre será mayor a cero; el proceso de replicación se presenta de manera asíncrona desde el lugar en el cual se origina la transacción. En otras palabras, siempre existirá un periodo de diferencia (de la transacción comprometida) entre el lugar dónde se origina la transacción y los lugares en dónde debe verse reflejada la misma. A este tiempo de diferencia lo identificaremos como **latencia** conforme a la documentación técnica del Replication Server. Si se cuenta con infraestructura apropiada y suficientes recursos, la latencia existente es usualmente cero.

Así, al haber explicado el concepto de replicación asíncrona, podemos comenzar a describir la funciones del Replication Server, mismas que a continuación se describen.

## Replication Server, herramienta de replicación de datos.

Replication Server mantiene datos replicados en múltiples bases de datos de manera asíncrona por medio de una red (llámese LAN o WAN) asegurando la integridad y consistencia de los datos a todos los clientes que utilizan las bases de datos de una red, reduciendo el tráfico sobre la red y sobre las computadoras en sistemas centralizados.

La arquitectura de Replication Server soporta servidores de datos heterogéneos, es decir, puede conectarse a diferentes ambientes de bases de datos. Es posible construir sistemas de replicación sobre aplicaciones y bases de datos existentes.

Los sistemas distribuidos de bases de datos, permiten accesar a los datos a través de centros distribuidos de procesamiento de datos. Como se muestra en la Figura II.6. Diagrama de Replicación, estos centros son muy frecuentemente compuestos de varias redes locales(LAN) y redes de área amplia (WAN).

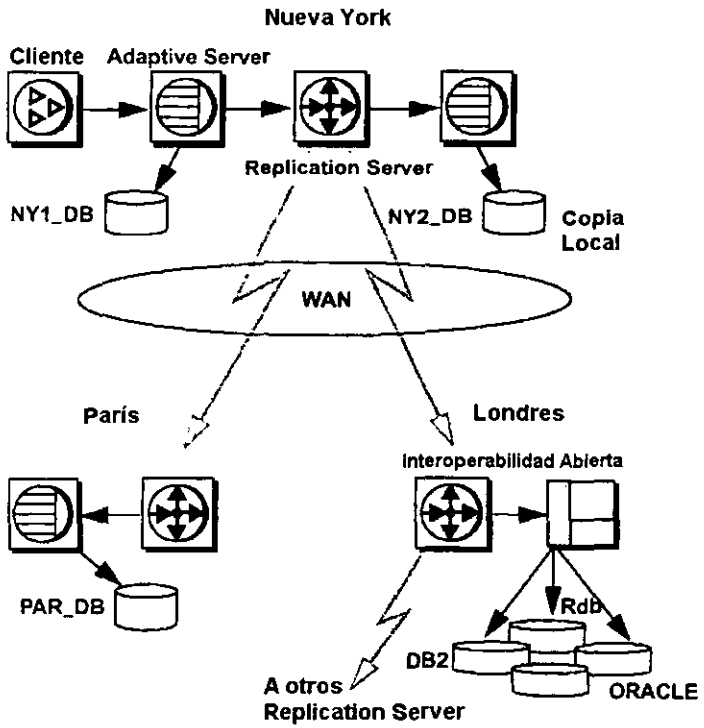



**Figura II.6. Diagrama de Replicación**

## Ventajas de la Replicación de Datos

La réplica de tablas en servidores locales da a los clientes las ventajas de una acceso local, lo cual significa un mejor desempeño y una mayor disponibilidad de los datos.

### Mejor Desempeño

La peticiones de datos son completadas en el servidor de datos local sin tener la necesidad de accesar a la WAN. Así pues, los clientes locales obtienen un mejor rendimiento porque:

- Las tasas de transferencia de información son más rápidas sobre una LAN que sobre una WAN.
- Los clientes locales comparten los recursos del servidor local en lugar de compartir con toda la comunidad de usuarios en un servidor central. Un ejemplo, de lo anterior sería el hecho de que la respuesta de una red local se degrada en proporción a como el tráfico en la WAN aumenta.
- El tráfico y la contención por bloqueos son reducidos considerablemente mediante una separación de las aplicaciones locales y las de soporte a decisiones (DSS) usando servidor central.

### Mayor Disponibilidad de Datos

Los datos son replicados en bases de datos locales y remotas. Por lo tanto, los clientes pueden continuar el trabajo en un ambiente con tolerancia a fallas:

- Cuando ocurra una falla en una base de datos remota, los clientes pueden utilizar la copia local de los datos replicados.
- Cuando suceda una falla en la WAN, los clientes locales pueden hacer uso de los datos locales.
- Cuando el servidor local falla, los clientes pueden utilizar datos replicados en otro sitio.

Una falla en la red o en la base de datos de algún otro sitio, no detienen el trabajo de la base de datos local. Cuando existe una falla de comunicaciones en la WAN, Replication Server almacena las operaciones que se realizan sobre tablas replicadas para que al momento de restablecer las comunicaciones puedan actualizarse los datos en las tablas que se encontraban inaccesibles. Si el servidor de datos local, falla, los clientes pueden continuar su trabajo, operando temporalmente con una copia replicada de los datos.

## Conceptos de Replication Server

Replication Server utiliza el modelo de base de datos relacional para representar datos en tablas que tienen un número fijo de columnas y un número variable de filas. Cada **tabla replicada** necesariamente debe tener una o varias columnas que puedan ser usadas como llave primaria para así identificar cada fila individualmente.

Replication Server permite definir los datos y procedimientos almacenados (store procedures) que se quieren replicar a bases de datos remotas. Como parte del diseño y planeación de un esquema de replicación, es necesario designar bases de datos fuente y bases de datos destino, para así crear rutas que los datos replicados puedan seguir de un Replication Server a otro.

En general, una base de datos fuente, contiene datos primarios y pueden por lo tanto llamarse **bases de datos primarias**, mientras que una base de datos destino contiene datos replicados y puede entonces llamarse **base de datos replicada**. Transacciones o ejecuciones de procedimientos almacenados son replicados a través del sistema de replicación desde la base de datos primaria hasta la replicada. Las ejecuciones de procedimientos almacenados pueden también ser replicados desde la base replicada a la primaria. Sin embargo, solo los datos primarios son directamente modificados.

Cuando se configura un esquema de replicación, será necesario en algún momento identificar aquellas tablas que serán primarias y aquellas tablas que serán replicadas, así para describir a cada tabla a replicar (tabla primaria) se crea una **replication definition** (definición de replicación). Una replication definition lista las columnas y tipos de datos de la tabla a replicar, así como las columnas que conforman la llave primaria para dicha tabla, junto con la especificación del lugar en dónde se localiza la versión primaria de la tabla. Además se listan las columnas que pueden ser usadas para delimitar los datos a los que se desea subscribir.

Posterior a la replication definition se deben crear **subscripciones** (subscriptions) a estas para especificar cuales serán los sitios, bases y/o tablas que recibirán las actualizaciones que afecten a la tabla primaria, basándose en las filas que se quieren replicar. Una subscripción indica al Replication Server que replique una tabla o parte de ella. Copias replicadas de datos no necesariamente deben ser copias completas de la tabla primaria. Estas pueden limitarse a las filas o columnas requeridas.

Por defecto, el crear una subscripción ocasionará que Replication Server copie los datos requeridos de una base de datos primaria a una replicada. Este proceso se llama **materialización de la subscripción**. Una vez que la creación y materialización de una subscripción son concluidas, Replication Server comienza a distribuir las transacciones de los datos primarios tan pronto como ocurren.

Para describir un **procedimiento almacenado a replicar**, se crea una **function replication definition** (definición de replicación de función). Esta incluye los parámetros y tipos de datos, la ubicación de los datos primarios que el procedimiento puede modificar, la lista de parámetros que pueden ser usados durante la subscripción a ejecuciones de estos procedimientos, y además indica el nombre del procedimiento en la base de datos destino.

Se crean subscripciones para una function replication definition con el propósito de replicar procedimientos almacenados de una base de datos primaria a una base de datos replicada. Aunque es posible replicar procedimientos de un sitio replicado a un sitio primario sin utilizar subscripciones.

## Componentes de un Sistema de Replicación

Un sistema de replicación basado en Replication Server presenta una arquitectura como la que se muestra en la Figura II.7. Arquitectura de un Sistema de Replicación cuyos componentes se describen a continuación.

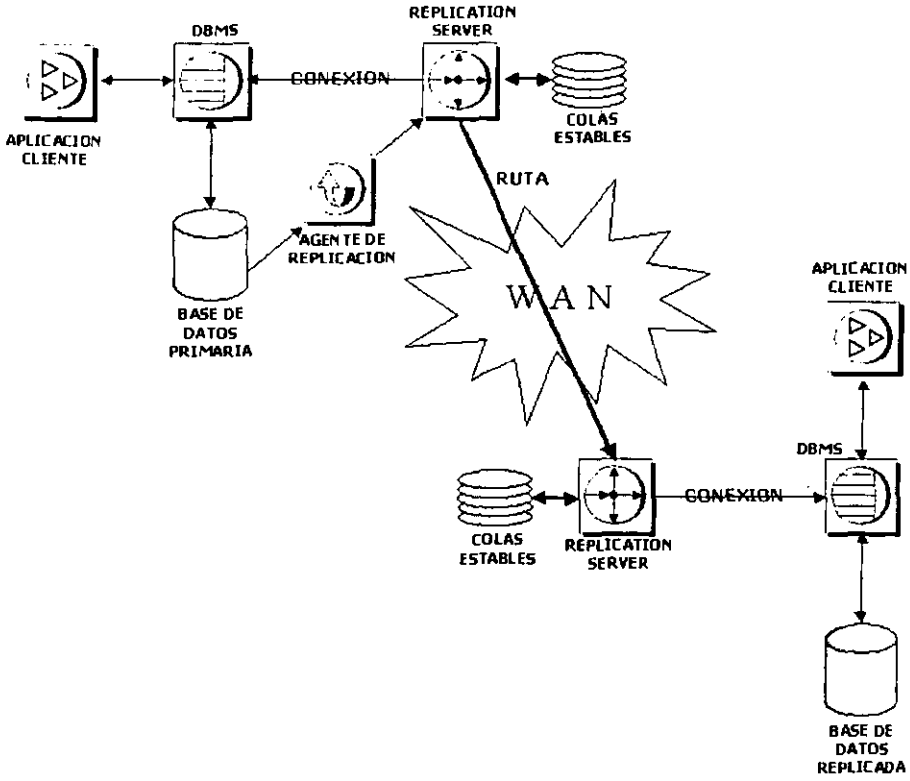



**Figura II.7. Arquitectura de un Sistema de Replicación**

### Replication Server

Un Replication Server en cada sitio se encarga de coordinar las actividades de replicación de datos para los servidores de datos locales y se encarga de intercambiar datos con Replication Servers de otros sitios.

Un Replication Server:

- Recibe transacciones de datos primarias desde las bases de datos por medio de los agentes de replicación y los distribuye a aquellos sitios que se hayan subscrito a esos datos.

- Recibe transacciones de otros Replication Servers y las aplica a bases de datos locales.

Las tablas de sistema del Replication Server almacenan información necesaria para realizar las actividades anteriores. Estas tablas de sistema incluyen descripciones de los datos replicados y de los objetos replicados tales como definiciones de replicación (replication definitions) y subscripciones (subscriptions), registros de seguridad sobre los usuarios propios del Replication Server, información sobre rutas a otros sitios, métodos de acceso a bases de datos locales e información administrativa adicional.

Las tablas de sistema del Replication Server se almacenan en una base de datos de Adaptive Server (comúnmente Sybase) que se conoce como Replication Server System Database (Base de Datos de Sistema del Replication Server) o simplemente RSSD. Una RSSD es necesaria por cada Replication Server y un Adaptive Server que contenga una RSSD puede también contener bases de datos que participen en la propia replicación u otras aplicaciones.

Replication Server posee su propio lenguaje que se conoce como RCL (Replication Command Language) mismo que utilizaremos más adelante.

### Particiones y colas estables

Replication Server almacena mensajes en disco para asegurarse de que estos puedan ser entregados después de una falla. Cuando se instala un Replication Server se asigna una partición inicial que Replication Server utiliza para este almacenamiento en disco. Es posible agregar o eliminar particiones posteriores a la instalación del Replication Server.

Dentro de estas particiones en disco, Replication Server asigna y distribuye datos entre las stable queues (colas estables) correspondientes a las rutas y conexiones que este administra. Los mensajes son almacenados en estas colas estables por lo menos hasta que se confirma que estos mensajes han sido recibidos por los sitios destino. Estas colas estables actúan como buffers para los datos que fluyen a través del sistema de replicación. Si un sitio remoto de Replication Server queda incomunicado por problemas en la red, el Replication Server primario almacena las transacciones (mensajes) en una cola hasta que la comunicación se restablezca. Con lo anterior, es obvio que entre más espacio sea asignado a las particiones en disco, Replication Server puede encolar datos sin necesidad de interrumpir el servicio en la base de datos primaria[8].

[8] El término base de datos primaria se define más adelante.

### Servidores de datos

Los servidores de datos o DBMS's controlan las bases de datos que contienen datos primarios o datos replicados. Las aplicaciones cliente los utilizan para almacenar y recuperar datos así como para procesar consultas o transacciones. Replication Server mantiene los datos replicados en los servidores de datos conectándose a ellos como un usuario normal.

Replication Server depende de los DBMSs para proporcionar los servicios del procesamiento de transacciones necesarios para proteger los datos que estos almacenan. Para garantizar la integridad de estos datos distribuidos, los servidores de datos deben cumplir con las propiedades de las transacciones como atomicidad y consistencia.

### Bases de datos primarias y replicadas

Como ya se ha mencionado dentro de este capítulo, una base de datos es una parte fundamental dentro de un esquema de replicación, por lo tanto, es indiscutible el hecho de mencionar a las bases de datos como un elemento en un Sistema de Replicación.

Es necesario aclarar que dentro de este sistema, podemos clasificar a las bases de datos de dos formas diferentes: bases de datos primarias y bases de datos replicadas.

El primer tipo de bases de datos, las primarias, se refieren a aquellas bases de datos que serán fuente de información, es decir, la información que en ellas exista será distribuida hacia otros puntos. Mientras que las bases de datos replicadas, son aquellas que recibirán datos provenientes de otras bases primarias.

Una base de datos puede comportarse de ambas formas, primaria ya que su información será replicada a otros sitios, o bien replicada ya que recibe información proveniente de alguna otra base de datos primaria.

### Agentes de replicación

Un agente de replicación se requiere para cada base de datos que almacena datos primarios. Este, lee el log de transacciones de la base de datos para detectar si existen cambios (transacciones comprometidas) en los datos primarios. De ser así, este agente de replicación lee la transacción para enviarla al Replication Server y que este se encargue de procesarla ya sea transmitiéndola a otro punto o aplicándola a una base de datos directamente.

### Aplicaciones cliente

Las aplicaciones cliente que se incluyen en la arquitectura, representan aplicaciones que permiten interactuar a la base de datos con los usuarios finales.

### Rutas y Conexiones

Las rutas permiten a los Replication Server intercambiar mensajes entre ellos, mientras que las conexiones le permiten enviar comandos directamente a las bases de datos. Una ruta, es un canal que permite el flujo de mensajes en una sola dirección de un Replication Server a otro. Una conexión es un canal de mensajes entre un Replication Server y una base de datos. En un ambiente, Warm Standby[9], Replication Server utiliza una conexión lógica para representar la base de datos activa y la base de datos standby.

Para replicar datos de una base a otra, es necesario establecer primero las rutas y conexiones que permitan al Replication Server mover los datos del origen al destino. Replication Server cuenta con herramientas que al momento de agregar una base de datos al ambiente de replicación, crean la conexión automáticamente, no siendo así para las rutas mismas que deben ser creadas manualmente.

Por lo anterior, si se cuenta con más de un Replication Server en la arquitectura, será necesario crear las rutas entre ellos. En cambio, si se cuenta con un solo Replication Server, no es necesario y de hecho no podría existir ruta alguna.

Cuando se crean rutas entre un servidor primario y un servidor replicado, las transacciones fluirán desde el primario hasta el replicado, obviamente si se contará con una replicación bidireccional, será necesario crear las rutas en ambos sentidos.

### LAN y WAN

Un ambiente de replicación de datos no tendría sentido sino exisitieran las redes como medio de comunicación entre los diferentes servidores que se pretende replicar información, así tanto las LAN como las WAN, forman una parte indispensable para la replicación de datos. El presente trabajo no tiene la finalidad de ampliar conceptos y conocimientos básicos de redes, así que se asume que aquellos que lean el presente tendrán por lo menos el conocimiento para diferenciar entre LAN y WAN.

## Modelos en un esquema de replicación

Para poder diseñar un esquema de replicación, existen configuraciones básicas que el Replication Server soporta y a partir de las cuales se pueden crear variaciones para obtener un funcionamiento adecuado a las necesidades que se requieran. A estas configuraciones se les da el nombre de modelos y cada uno de estos se explican a continuación.

Los modelos básicos que se consideran en un esquema de replicación son:

- Copia de datos primarios.

[9] Véase Modelo WARM-Standby en la página 41.

- Fragmentos primarios distribuidos.
- Consolidado corporativo.
- Consolidado corporativo redistribuible.
- WARM Standby.

### Modelo básico de copia de datos primarios

Este modelo permite replicar datos de una base de datos primaria[10] a diferentes bases de datos destino. Es adecuado para aplicaciones de soporte a decisiones, aunque aplicaciones con un bajo volumen de transacciones pueden actualizar datos primarios en forma remota ya sea directamente por medio la WAN o mediante request functions (procedimientos almacenados replicados). De esta forma los datos que son actualizados desde lugares remotos, pueden ser actualizados a aquellos otros sitios que estén suscritos a los datos primarios. En la Figura II.8. Modelo Básico de Copia de Datos Primarios, se muestra gráficamente este modelo de replicación.

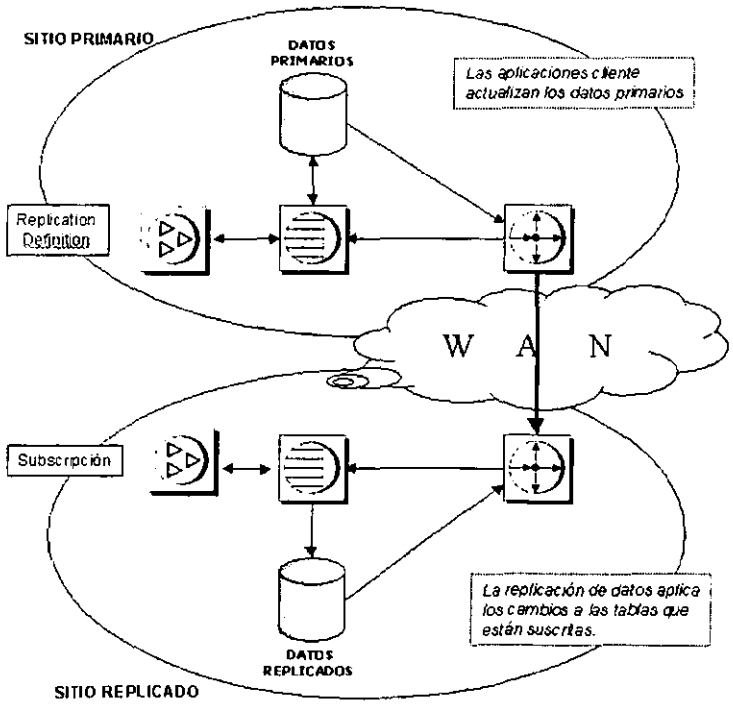



**Figura II.8. Modelo Básico de Copia de Datos Primarios**

[10] Base de datos primaria véase Conceptos de Replication Server en la página 29.

Sin embargo, como ya se mencionó anteriormente, este modelo puede tener una variación que implica que los sitios replicados actualicen información del sitio primario con el propósito de distribuir esos cambios a todos los sitios, inclusive a ellos mismos. Así, podemos considerar que existen dos formas en las que puede replicarse información bajo este modelo:

- Replicando los cambios de la tabla primaria[11].
- Replicando la ejecución de procedimientos almacenados[12].

El primer caso es el que se ejemplifica en la Figura II.9. Modelo Básico de Copia de Datos Primarios con actualización vía red, mientras que en la Figura II.10. Modelo Básico de Copia de Datos Primarios con actualización vía Request Functions se muestra el segundo caso, mismo que puede ser de gran utilidad por citar un ejemplo, para actualizar saldos de clientes que tienen operaciones en varios sitios.

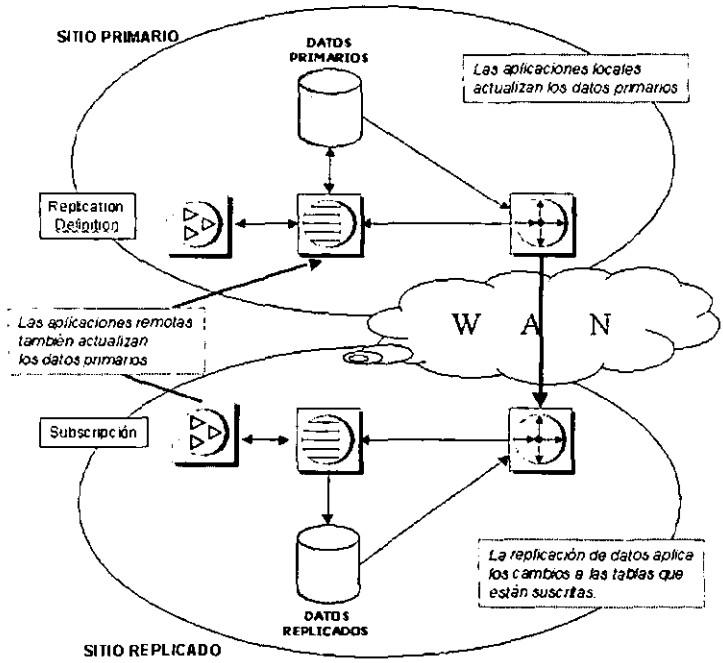



**Figura II.9. Modelo Básico de Copia de Datos Primarios con actualización vía red**

[11] Tablas a replicar véase Conceptos de Replication Server en la página 29.
[12] Procedimientos almacenados a replicar véase Conceptos de Replication Server en la página 29.

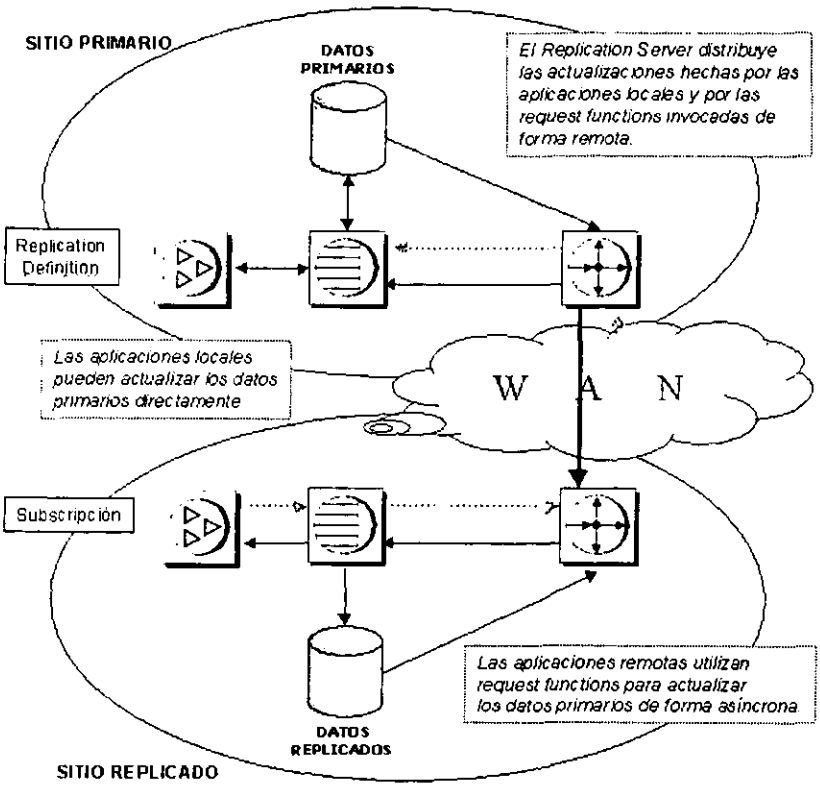



**Figura II.10. Modelo Básico de Copia de Datos Primarios con actualización vía Request Functions**

Modelo de fragmentos primarios distribuidos

En este modelo, existen tablas en cada sitio que contienen tanto datos primarios como datos replicados. Sin embargo, cada sitio funciona como un sitio primario para un subconjunto de renglones al cual se le llama fragmento. Las actualizaciones a el fragmento primario son distribuidas a los otros sitios. Las actualizaciones a los datos que no son primarios, son recibidas de otros sitios primarios para otros fragmentos.

Las aplicaciones que utilizan este modelo de replicación tienen tablas distribuidas que contienen datos primarios y replicados. El Replication Server de cada sitio distribuye las modificaciones realizadas a datos locales primarios hacia otros sitios y aplica las modificaciones que se reciben de otros sitios. Este modelo se muestra en la Figura II.11. Modelo de fragmentos primarios distribuidos.

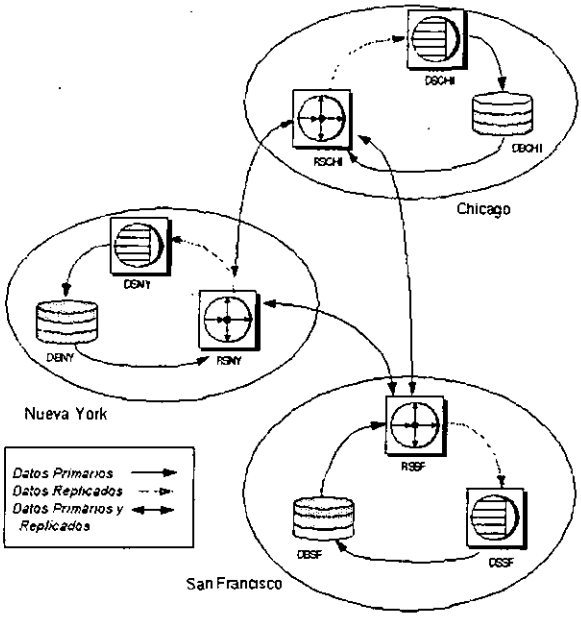



**Figura II.11. Modelo de fragmentos primarios distribuidos**

A continuación se presenta un ejemplo de algunas tablas configuradas bajo el esquema anterior.

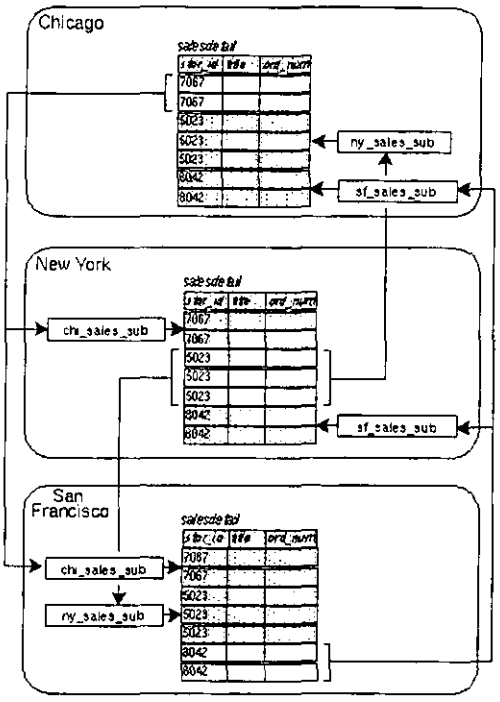



**Figura II.12. Tabla distribuida con fragmentos primarios.**

### Modelo consolidado corporativo

En este modelo, múltiples fragmentos primarios los cuales se encuentran en lugares remotos son consolidados en una tabla replicada que se ubica en el sitio central.

Este modelo tiene fragmentos primarios distribuidos y una tabla consolidada a nivel central. Cada una de las tablas en los sitios primarios contienen solo registros que son primarios en ese lugar y no se replica información a estas tablas. La tabla central es una tabla que consolida la información a nivel corporativo de cada una de las tablas primarias.

Este modelo requiere definiciones de replicación en cada uno de los sitios primarios y por cada uno de estos sitios debe crearse una subscripción en el sitio central.

Agentes de replicación son indispensables en los sitios primarios, sin serlo en el lugar en dónde se consolida la información.

La Figura II.13. Modelo consolidado corporativo presenta gráficamente este modelo de replicación.

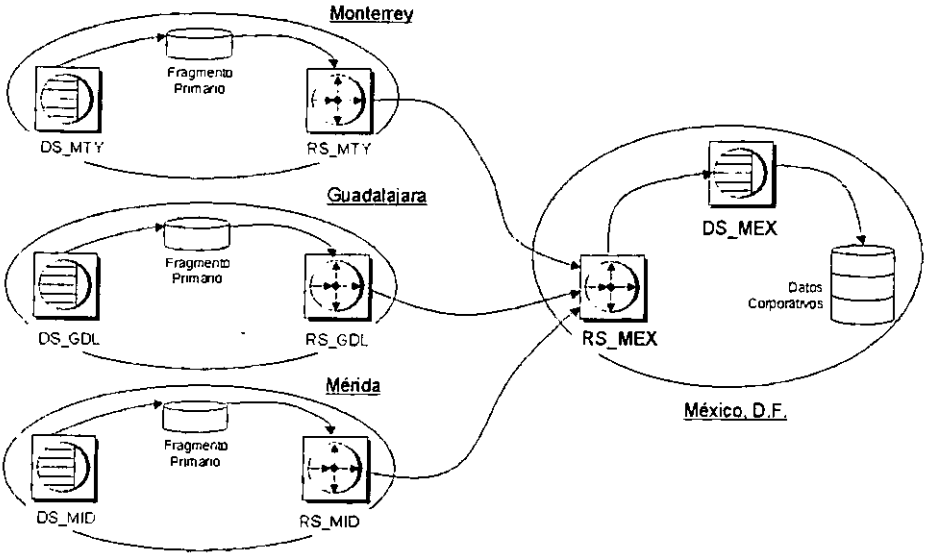



**Figura II.13. Modelo consolidado corporativo**

En la Figura II.14. Tabla distribuida con el modelo consolidado corporativo, se presenta un ejemplo de una tabla que se configure bajo este esquema:

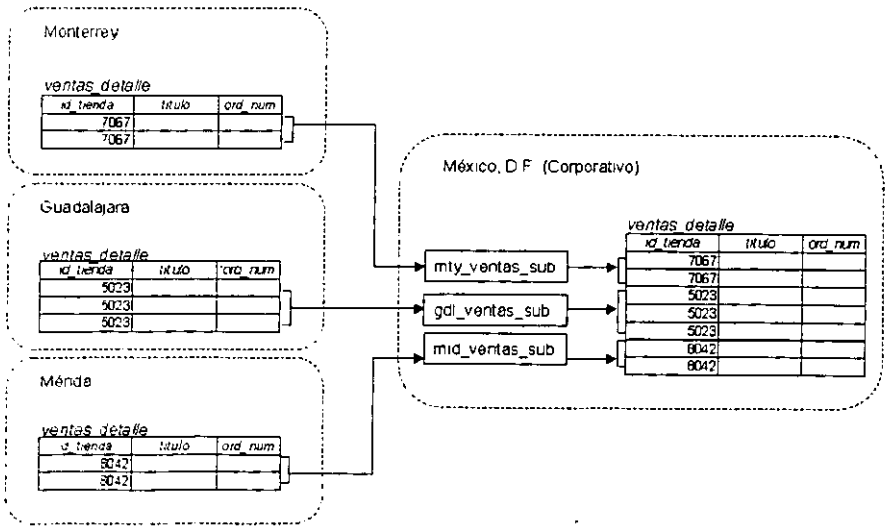



**Figura II.14. Tabla distribuida con el modelo consolidado corporativo**

### Modelo consolidado corporativo redistribuible

La Figura II.15. Modelo consolidado corporativo redistribuible presenta el modelo consolidado corporativo redistribuible, siendo este el mismo que el modelo corporativo consolidado con excepción que la tabla central se distribuye hacia sitios remotos.

Los fragmentos primarios que se encuentran distribuidos en los sitios remotos, mismos que son enviados hacia la tabla consolidada central. En el lugar en dónde se consolida la información, un agente de replicación procesará la tabla consolidada como si fueran datos primarios.

La tabla consolidada es descrita mediante una definición de replicación para que otros sitios puedan subscribirse a esta tabla.

Normalmente, el agente de replicación filtra las actualizaciones hechas por el usuario de mantenimiento para que estas no sean replicadas nuevamente. Esto asegura que los datos replicados no se redistribuyan como si fueran datos primarios.

El agente de replicación posee la opción *send_maint_xacts_to_replicate*, misma que permite habilitar este modelo. Si el agente de replicación se inicia con valor *true* en esta opción, se envían todas las transacciones al Replication Server como si estas fueran realizadas por cualquier aplicación.

Si se llega a utilizar este modelo:

- No se debe permitir a sitios primarios resubscribirse a sus datos primarios. Si se llegara a realizar, las transacciones podrían quedarse en un ciclo infinito en el sistema de replicación.
- No se debe permitir a las aplicaciones actualizar la tabla corporativa. Todas las actualizaciones deberán realizarse en los sitios primarios.

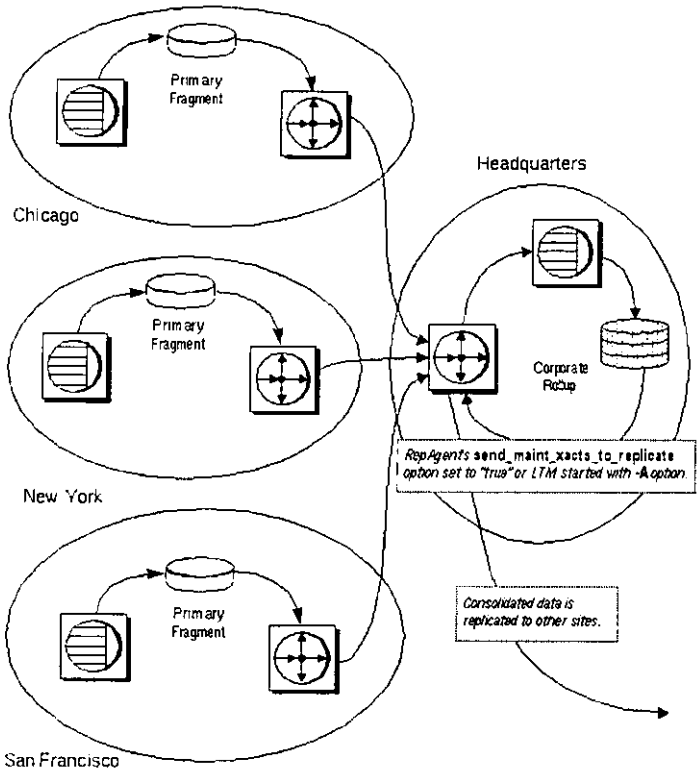



**Figura II.15. Modelo consolidado corporativo redistribuible**

## Modelo WARM-Standby

En una aplicación WARM Standby, el Replication Server mantiene dos bases de datos en la cual una actúa como respaldo de la otra.

Comúnmente los clientes se conectan a la base de datos primaria y en ella realizan sus actualizaciones mientras que Replication Server mantiene a la base de datos de respaldo como una copia de la original. Si la base de datos activa falla, o si se requieren realizar acciones de mantenimiento en el equipo o en la base de datos, puede cambiarse la operación a la base de datos de respaldo con una pequeña interrupción para los clientes.

Con esto se concluye el marco teórico de todos aquellos elementos que intervienen dentro de un esquema de replicación, es decir, sabemos que es lo que conforma a un esquema de replicación, sin embargo, es necesario contar con las herramientas que nos permitan diseñar un esquema de

replicación acorde a las necesidades que se presente. Ese es el propósito del siguiente capítulo en el cual se muestra una metodología que permitirá planear el esquema de replicación conforme a lo requerido.

# CAPITULO III. DISEÑO DE UN ESQUEMA DE REPLICACION

*OBJETIVO: Proporcionar parámetros y técnicas para la concepción y diseño de un esquema de replicación con Replication Server.*

## Diseño de un esquema de replicación

El diseño de un ambiente de base de datos distribuida es un proceso en el que se deben ajustar las necesidades de los usuarios y de la propia empresa a la infraestructura física y a la tecnología disponible en la misma. Esta base de datos distribuida deberá ajustarse a los datos y procesos que se requieran en un sitio específico, es decir, debemos de tener la información en dónde la necesitemos.

Para implementar una base de datos distribuida, se utilizará el Replication Server de Sybase, Inc como la herramienta que se encargará de distribuir la base de datos, de tal forma que los conceptos que en este capítulo se manejen serán orientados a dicha herramienta.

Un esquema de replicación en primera instancia aparenta ser sumamente sencillo y en efecto así puede ser, sin embargo, un gran problema que se presenta al comenzar el diseño de un esquema de replicación aparece en el momento en que al usuario, el cual quiere replicar su información, se le hace una pregunta básica para comenzar con este diseño: "¿Cómo?", es decir, cuales tablas quieres que se actualicen en el servidor primario para que esos cambios se reflejen en los servidores replicados, o cuales tablas podrán modificarse en los sitios remotos para que esos cambios se concentren en el servidor principal o peor aún, cuales tablas se distribuirán parcialmente a cada sitio.

En base a la experiencia propia, esto es lo que habitualmente sucede cuando se inician los trabajos para diseñar un esquema de replicación, de tal forma que para establecer un orden que nos ayude a planear adecuadamente un esquema de replicación, será necesario seguir una metodología misma que se describirá a continuación.

## Metodología

El libro titulado Data Replication de Marie Buretta presenta una metodología compuesta por nueve pasos a seguir para el diseño de la distribución de los datos. Esta metodología se adopta derivada del trabajo que esta autora realiza específicamente sobre la replicación de datos. La metodología se compone de los siguientes pasos:

3. Preparar un perfil para la distribución de la aplicación.
4. Preparar un perfil de la infraestructura física.
5. Crear grupos de datos con integridad referencial.
6. Preparar el perfil de uso de procesos – datos.

7. Integrar el uso global de los datos con el perfil de procesos - datos.

8. Crear un esquema de colocación de los datos

9. Validar el esquema de colocación contra las restricciones existentes así como con las capacidades y tecnologías.

10. Validar el esquema de colocación contra los requerimientos de nivel de servicio.

11. Implementar el esquema de distribución.

A continuación se describirán cada uno de los nueve pasos que conforman esta metodología y se aplicarán al caso que se maneja en este trabajo.

1. Perfil para la distribución de la aplicación.

Este paso tiene como objetivo identificar los procesos en los que las aplicaciones de la empresa pueden ser divididas, de tal forma que se pueda identificar que procesos son ejecutados en los diversos niveles de la empresa, por ejemplo, podemos tomar una empresa *X* la cual este conformada por oficinas centrales, regionales y por último cada región conste de unidades más pequeñas. Con lo anterior, es obvio que las actividades que desempeñan las oficinas centrales, difieren de las actividades que desarrollan las oficinas regionales y/o locales.

Al concluir con las actividades de este punto de la metodología, contaremos con la siguiente información:

- La naturaleza distribuida de la aplicación.
- Una estructuración inicial de los requerimientos de red.

Para poder cumplir con los puntos anteriores, podemos considerar de gran ayuda lo siguiente:

- El conocimiento de los requerimientos de la aplicación, así como los diversos niveles que forman parte de la organización.
- Los diagramas funcionales de descomposición.- Un diagrama funcional de descomposición es una plantilla estructurada en dónde las funciones de alto nivel del negocio son descompuestas jerárquicamente en diferentes aplicaciones o procesos. La descomposición se especifica generalmente hasta un nivel que representa un proceso manual o un módulo de un software.

En este paso deben realizarse dos tipos de tareas. En la primera, nos permitirá identificar las categorías lógicas entre los diversos puntos que forman parte de la organización, así como de los usuarios y procesos, mientras que en la segunda debemos asignar valores cuantitativos a cada categoría. Así, podemos determinar los siguientes elementos:

- Los diferentes tipos de sitios que forman parte de la organización.

- Los tipos de usuarios para la aplicación.
- Los tipos de transacciones y/o los procesos más importantes que son ejecutados por los usuarios.
- Los valores cuantitativos a cada tipo de sitio.

De acuerdo a lo que se expuso anteriormente y derivado del conocimiento de los sistemas administrativos y/o informáticos dentro de Aeropuertos y Servicios Auxiliares, se puede establecer que para el caso de ASA existen los siguientes niveles:

Oficinas Generales.- Corresponden a las oficinas centrales de ASA y representan la parte normativa del organismo.

Aeropuertos.- En este tipo de sitios ASA administra el aeropuerto como la estación de combustible correspondiente.

Estaciones de combustible.- Así se identificarán aquellos puntos en dónde el aeropuerto no es administrado por ASA sino solo la estación de combustible. Esto se debe a un proceso de privatización por medio del cual la administración del aeropuerto se concesionó a la iniciativa privada más no así la estación de combustible que sigue siendo parte de la institución.

Como se mencionó al inicio, cada nivel tiene perfectamente identificados los procesos y/o aplicaciones que cada uno tiene como responsabilidad, es decir:

Entre otras actividades, oficinas generales se encarga de controlar las solicitudes y contratos de crédito, actualización y mantenimiento de tarifas, expedición de facturas de crédito y captura de información de aquellos aeropuertos que no están en línea; esto se realiza básicamente por personal de Contratación y Facturación. Se manifiestan solo estas áreas ya que corresponden al esquema de replicación que se está mostrando en este trabajo.

Un aeropuerto se encarga de capturar itinerarios de vuelos, capturar manifiestos de llegada y salida, capturar la información de abordadores, registrar las remisiones de combustible así como sus movimientos en el inventario, verificar la situación crediticia del cliente, emitir y cobrar facturas de contado; estas actividades son realizadas por las áreas de operaciones, finanzas y combustibles (esta última especificamente por la estación de combustible).

Una estación de combustible se encarga de registrar las remisiones de combustible así como los movimientos que se generan en el inventario, emitir y cobrar las facturas de contado; esto es realizado por personal propio de la estación de combustible.

Todo esto se resume en la Tabla III.1. Perfil para la distribución de la aplicación, en la cual podemos identificar los elementos que se buscaban al inicio de este paso.

**Tabla III.1. Perfil para la distribución de la aplicación**

| Tipo de Sitio | Ubicación física | Tipos de Usuarios | Tipos de Transacciones y procesos |
|---|---|---|---|
| Oficinas Generales | México, D.F. | Contratación | *Mantenimiento a Contratos |
| | | | *Expedición de Autorizaciones de Crédito. |
| | | | *Mantenimiento a Matrículas, Países y Aviones (Catálogos) |
| | | Facturación | Captura de información de aeropuertos que no están en línea. |
| | | | *Mantenimiento a Tarifas de Servicios (Catálogos) |
| | | | *Expedición de Facturas |
| | | Cobranza | Aplicación de pagos |
| | | | Emisión de estados de cuentas |
| Aeropuertos | Toluca, Edo. Méx. | Operaciones | *Mantenimiento de itinerarios |
| | Puebla, Pue. | | *Planeración de Vuelos |
| | ... | | *Registro del movimiento operacional |
| | | | *Registro de Manifiestos |
| | | | *Registro de Abordadores |
| | | Finanzas | Consultas sobre situación crediticia de clientes |
| | | | Expedición de Facturas (Contado) |
| | | | Cobranza |
| | | Combustibles | *Registro de Remisiones |
| | | | *Registro de Succiones |
| | | | Consulta de tarifas |
| | | | *Expedición de facturas (Contado) |
| | | | Cobranza |
| Estaciones de Combustible | México, D.F. | Combustibles | *Registro de Remisiones |
| | Monterrey, N.L. | | *Registro de Succiones |
| | Guadalajara, Jal. | | Consulta de tarifas |
| | | | *Expedición de facturas (Contado) |
| | | | Cobranza |

*** Estos procesos son los que se utilizarán para el presente trabajo.**

## 2. Perfil de la infraestructura física.

En este segundo punto de la metodología se realizarán actividades que nos permitan documentar e identificar lo siguiente:

- Los recursos físicos disponibles en los puntos potenciales de la organización.
- Las restricciones existentes en estos posibles puntos de la distribución.

Para determinar la viabilidad de la distribución de datos y/o procesos a un punto determinado, es muy importante tener presente todos los recursos y restricciones de dicho punto antes de tomar la decisión de incluirlo o no en el esquema. Para realizar esta actividad debemos tener presente lo siguiente:

- La infraestructura existente en cada punto potencial de la organización.
- Cualquier posible restricción que pudiera existir en la configuración.

Además, en el desarrollo de este segundo punto de la metodología se puede incluir el consultar al personal de soporte e infraestructura acerca de cualquier documentación electrónica o física existente sobre los siguientes puntos:

- Conectividad de la red (WAN y LAN).
- Recursos de hardware y su capacidad disponible.
- Recursos de software y su capacidad disponible.
- Niveles de seguridad soportados.
- Requerimientos de disponibilidad del sistema actualmente soportados.
- Soporte administrativo y sus diferentes niveles asociados.
- Cualquier restricción que pudiera influir en el uso de cualquier punto de la organización.

Tomando como base las cantidades anticipadas de datos y/o número de usuarios concurrentes, se debe identificar cualquier restricción de la plataforma, esto es, debemos determinar si la capacidad del medio físico será suficiente para soportar el volumen de información que será transmitido por el mismo. Adicionalmente, se debe identificar cualquier restricción relacionada con la disponibilidad o exactitud de una plataforma específica, por ejemplo, una restricción puede ser el hecho que una aplicación requiera una disponibilidad de 7 x 24, aunque la plataforma pudiera actualmente estar disponible sólo por 12 horas consecutivas.

La Tabla III.2. Perfil de la infraestructura física nos permite concentrar la información requerida en este segundo punto.

**Tabla III.2. Perfil de la infraestructura física**

| Sitio/Tipo de Sitio | Conectividad de la Red (WAN y LAN) | Recursos de Hardware | Recursos de Software | Nivel de Seguridad | Nivel de Disponibilidad | Soporte Administrativo y Restricciones | Restricciones Adicionales |
|---|---|---|---|---|---|---|---|
| México, D.F.<br>Oficinas Generales | DC0 – A todos los Aeropuertos (64KB). | Dell<br>Dell Cluster<br>HP 900 T500 | Windows NT Server 4.0 Service Pack 4. Español.<br>Sybase Adaptive Server Enterprise 11.5.1 | Bajo | Lunes a Viernes<br>8:00 – 22:00 | | Excepto Cancún y AICM.<br>Existen DC0's en todos los aeropuertos que se interconectan por medio de la red pública de datos de RED UNO. |
| | 56Kb – Estación de Combustible de Cancún | Compaq Proliant 1500 | Windows NT Server 4.0 Service Pack 4. Español<br>Sybase Adaptive Server Enterprise 11.5.1 | Bajo | Sábado<br>8:00 a 16:00 | | Es vía modem. |
| | 2MB – AICM | Compaq Proliant 2500 | Windows NT Server 4.0 Service Pack 4. Español<br>Sybase Adaptive Server Enterprise 11.5.1 | Bajo | 7x24 | | |
| Estados de la República<br>Aeropuertos | DC0 – Oficinas Generales México | Compaq Proliant 1500 | Windows NT Server 4.0 Service Pack 4. Español<br>Sybase Adaptive Server Enterprise 11.5.1 | Bajo | 7 x 24 | | |
| Estados de la República<br>Estación de Combustible | 2 MB – Aeropuerto | Compaq Proliant 1500 | Windows NT Server 4.0 Service Pack 4 Español | Bajo | 7 x 24 | | La conexión del aeropuerto a la estación de combustibles es por microondas a una distancia aproximada de 1.5 KM, es por este motivo que la velocidad disminuye a 2MB.<br>En el caso de los aeropuertos que no son de ASA se tiene convenios con los grupos aeroportuarios. |
| México, D.F.<br>Estación de Combustible | DC0 – Aeropuerto Internacional de la Ciudad de México | | | | | | La conexión del aeropuerto a la estación de combustibles es por medio de la red pública de datos pero debido a restricciones de tipo presupuestal es necesario que se conecte primero al AICM. |

## 3. Conjuntos de recuperación de datos con integridad referencial.

El asegurar la integridad de los datos es de gran importancia en un esquema de replicación, esta debe ser preservada no solo durante las actividades normales, sino también en las tareas de administración como lo puede ser la recuperación de los datos en caso de alguna falla.

Al concluir este punto, podemos identificar lo siguiente:

- Los conjuntos de recuperación de datos.- Estos grupos son entidades de datos cuyas reglas de integridad referencial se relacionan entre sí de tal forma que se mantiene la consistencia de los datos, por ejemplo, una relación uno a muchos.

- Las oportunidades para una división física de los datos en los diversos sitios de la organización.

Para obtener satisfactoriamente la información buscada en este punto, podemos hacer uso de:

- Diagrama entidad relación.- Este modelo nos dará una vista lógica de los datos requeridos para soportar la funcionalidad deseada. Este debe ser producto del desarrollo de cualquier aplicación.

- Conocimiento profundo de las relaciones existentes entre las diferentes entidades de datos por medio de la integridad referencial. Es decir, la experiencia de aquellos individuos involucrados directamente con el diseño y/o explotación de la base de datos.

- Comprender cualquier integridad existente entre los diferentes sitios. Esta puede ser implementada por cualquier herramienta middleware[13] que sea utilizada.

- Un cuarto punto, que no es contemplado dentro de la metodología de Marie Buretta , pero el cual derivado de la experiencia diaria resulta ser de gran ayuda, consiste en considerar en este punto los diagramas funcionales de descomposición, mismos que se describieron en la página 44.

Cuando se está realizando el modelado lógico para los sistemas distribuidos, es de gran ayuda añadir datos adicionales (como características relacionadas a las entidades, relaciones y atributos) e incorporarlos en el diccionario de datos. Estas características adicionales son de gran utilidad cuando se diseña el esquema de distribución.

Durante esta actividad será necesario dividir el diagrama entidad relación en subconjuntos de entidades los cuales no incluyan relaciones con otros subconjuntos y además necesitaremos minimizar el número de subconjuntos en que se particione el modelo global.

---

[13] El término middleware no tiene una traducción al español, sin embargo, se considera como un software que actúa como una capa de conversión o traducción. Lo anterior según atomica.com (para mayor información sobre Gurunet consultar la bibliografía).

Se inicia agrupando las entidades lógicamente y listando los grupos y las entidades en una matriz. Por ejemplo, podemos asociar todas las entidades relacionadas con la información de los clientes, ordenes de compra, facturación, etc. Cada entidad o grupo lógico de entidades debe aparecer solo una vez en la matriz. Conforme se vayan identificando grupos, se recomienda determinar la asignación en la matriz bajo las siguientes consideraciones:

- Entidades con alta integridad referencial deben ser listadas en el mismo grupo dentro de la matriz, esto es, ellos deberían almacenarse y recuperarse de manera conjunta. Un ejemplo de este tipo de entidades incluye la tabla maestra de las ordenes de compra y el detalle de las mismas.

- Entidades con integridad referencial media que son altamente volátiles deben ser de igual forma listadas en el mismo grupo dentro de la matriz, para almacenarse y recuperarse conjuntamente. Alta volatilidad significa actividad de actualización frecuente. La relación media existe entre grupos lógicos. Un ejemplo de entidades de este tipo incluyen las referencias de llaves foráneas de productos con la de detalles de ordenes de compra.

- Entidades con integridad referencial media que no son altamente volátiles y no necesariamente se almacenan y recuperan conjuntamente. No ser altamente volátiles significa que la actividad de actualización se presenta con muy poca frecuencia. Estas entidades pueden ser denormalizadas o replicadas. Un ejemplo de este tipo de entidades incluyen la relación entre la tabla maestra de ordenes de compra y la tabla de clientes.

- Entidades con una relación débil son buenas candidatas para ser replicadas al lugar en dónde son utilizadas. Un ejemplo de este tipo de entidades son las relaciones entre las entidades transaccionales y las entidades de solo lectura que son utilizadas para validar los procesos de inserción o actualización, estas entidades generalmente se les da el nombre de catálogos. Este tipo de entidades puede incluir la tabla de códigos de estado, la tabla de códigos de moneda y otro tipo de tablas que sirvan para la validación.

Se continua la afinación de la lista hasta que todas las entidades hayan sido asignadas a un grupo. Generalmente los grupos representan datos estrechamente relacionados transaccionalmente de tal forma que deben ser recuperados como un conjunto, los datos con una débil relación referencial pueden ser replicados a los sitios de proceso, y un pequeño grupo de entidades con una media integridad se relacionan con otros grupos con una mayor relación. Para estos grupos con una relación media en dónde las llaves foráneas no son volátiles, pueden ser consideradas para la denormalización o replicación a los sitios en donde sean utilizadas. Adicionalmente, se debe identificar columnas que puedan ser utilizadas para particionar los datos a lo largo de los diferentes puntos de la organización, por ejemplo, un número de sucursal pudiera ser de gran utilidad para particionar los datos de un banco.

La denormalización es un proceso en el cual deliberadamente se violan las técnicas de normalización. Cuando se presenta la denormalización o la redundancia, anomalías de actualización

o borrado pueden presentarse cuando los datos son modificados en uno pero no en todos los lugares en dónde se almacena. El resultado es que el usuario observa inconsistencia de los datos en las consultas en donde se utilizan las tablas que presentan redundancia. Cuando se aplican técnicas para la denormalización, es necesario incorporar procesos de actualización y borrado que apliquen los cambios a los datos redundantes en dónde estos se encuentren almacenados.

Cuando se considera la denormalización de datos, los siguientes factores deben tomarse en cuenta:

- La actividad que se realizará para modificar los datos redundantes.
- Complejidad de los procesos a utilizar para prever la actualización y borrado.
- Complejidad de los procedimientos de reconciliación si las estructuras denormalizadas son almacenadas en diferentes lugares.
- Alternativas de replicación utilizadas a lo largo de los lugares distribuidos.

Siguiendo con el ejemplo que se ha expuesto durante este trabajo, tenemos que existen los siguientes grupos lógicos de entidades:

- Catálogos
- Clientes
- Contratos
- Movimiento Operacional
- Abordadores
- Manifiestos
- Combustibles
- Facturación

Y de los diagramas de entidad relación existentes del modelo de la base de datos, tenemos las siguientes entidades que se listan a continuación:

a) asa_c_avion

b) asa_c_cliente

c) asa_c_matricula

d) asa_c_pais

e) asa_d_abordador

f) asa_d_clidom

g) asa_d_conxmat

h) asa_d_excxreg

i) asa_m_abordador

j) asa_m_contrato

k) asa_m_manifa

l) asa_m_manifs

m) asa_m_registro

n) cco_d_itinerario

o) cco_d_planeacion

p) cco_m_itinerario

q) cmb_m_remision

r) cmb_m_succion

s) fac_d_facconc

t) fac_d_factura

u) fac_h_cierre

v) fac_m_factura

w) scf_d_autorizacion

x) scf_d_lugxaut

y) scf_m_autorizacion

z) fac_c_tarifa

aa) fac_c_tarcom

bb) fac_c_tartua

En la Tabla III.3. Agrupación Lógica de Entidades se muestra un resumen de la información obtenida en este punto.

**Tabla III.3. Agrupación Lógica de Entidades**

| Entidad / Subconjunto | Catálogos | Clientes | Contratos | Autorizaciones De Crédito | Movimiento Operacional | Abordadores | Manifiestos | Combustibles | Facturación |
|---|---|---|---|---|---|---|---|---|---|
| a) asa_c_avion | ✓ | | | | | | | | |
| b) asa_c_cliente | | ✓ | | | | | | | |
| c) asa_c_matricula | ✓ | | | | | | | | |
| d) asa_c_pais | ✓ | | | | | | | | |
| e) asa_d_abordador | | | | | | ✓ | | | |
| f) asa_d_didom | | ✓ | | | | | | | |
| g) asa_d_conxmat | | | ✓ | | | | | | |
| h) asa_d_excxreg | | | | | ✓ | | | | |
| i) asa_m_abordador | | | | | | ✓ | | | |
| j) asa_m_contrato | | | ✓ | | | | | | |
| k) asa_m_manifa | | | | | | | ✓ | | |
| l) asa_m_manifs | | | | | | | ✓ | | |
| m) asa_m_registro | | | | | ✓ | | | | |
| n) cco_d_itinerario | | | | | ✓ | | | | |
| o) cco_d_planeacion | | | | | ✓ | | | | |
| p) cco_m_itinerario | | | | | ✓ | | | | |
| q) cmb_m_remision | | | | | | | | ✓ | |
| r) cmb_m_succion | | | | | | | | ✓ | |
| s) fac_d_facconc | | | | | | | | | ✓ |
| t) fac_d_factura | | | | | | | | | ✓ |
| u) fac_h_cierre | | | | | | | | | ✓ |
| v) fac_m_factura | | | | | | | | | ✓ |
| w) scf_d_autorizacion | | | | ✓ | | | | | |
| x) scf_d_lugxaut | | | | ✓ | | | | | |
| y) scf_m_autorizacion | | | | ✓ | | | | | |
| z) fac_c_tarifa | ✓ | | | | | | | | |
| aa) fac_c_tarcom | ✓ | | | | | | | | |
| bb) fac_c_tartua | ✓ | | | | | | | | |

En la matriz CRUD, se debe identificar con un asterisco cualquier proceso que debiera considerarse "crítico".

Las entidades ya se han listado en el paso número tres, así que se utilizarán estas mismas dentro de la matriz CRUD.

a) asa_c_avion

b) asa_c_cliente

c) asa_c_matricula

d) asa_c_pais

e) asa_d_abordador

f) asa_d_clidom

g) asa_d_conxmat

h) asa_d_excxreg

i) asa_m_abordador

j) asa_m_contrato

k) asa_m_manifa

l) asa_m_manifs

m) asa_m_registro

n) cco_d_itinerario

o) cco_d_planeacion

p) cco_m_itinerario

q) cmb_m_remision

r) cmb_m_succion

s) fac_d_facconc

t) fac_d_factura

u) fac_h_cierre

v) fac_m_factura

w) scf_d_autorizacion

x) scf_d_lugxaut

y) scf_m_autorizacion

z) fac_c_tarifa

aa) fac_c_tarcom

bb) fac_c_tartua

Ahora bien, los procesos que podemos identificar dentro de las aplicaciones que se integrarán al esquema de replicación, son los siguientes:

- Mantenimiento de Catálogos
- Mantenimiento a Contratos
- Autorizaciones de Crédito
- Mantenimiento a Itinerarios
- Planeación Diaria
- Registro del Movimiento Operacional
- Registro de Abordadores
- Registro de Manifiestos
- Registro de Remisiones
- Registro de Succiones
- Emisión de Facturas

La Tabla III.4. Matriz CRUD, ilustra el perfil de procesos–datos que se obtiene después de relacionar los procesos y entidades anteriores.

**Tabla III.4. Matriz CRUD**

| | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | aa | bb |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Mantenimiento de Catálogos | CRUD | | CRUD | CRUD | | | | | | | | | | | | | | | | | | | | | | CRUD | CRUD | CRUD |
| Mantenimiento a Contratos | | CRUD | | | | CRUD | CRUD | | | CRUD | | | | | | | | | | | | | | | | | | |
| Autorizaciones de Crédito | R | R | R | | | | | | | R | | | | | | | | | | | | | CRUD | CRUD | CRUD | | | |
| Mantenimiento a Itinerarios | R | | | | | | | | | | | | | CRUD | | CRUD | | | | | | | | | | | | |
| Planeación de Vuelos | R | | | | | | | | | | | | | R | CRUD | R | | | | | | | | | | | | |
| Registro del Movimiento Operacional (*) | R | R | R | R | | | R | CRUD | | R | | | CRUD | R | CRUD | | | | | | | | R | R | R | | | |
| Registro de Abordadores | | | | | CRUD | | R | | CRUD | R | | | | | | | | | | | | | R | R | R | | | |
| Registro de Manifiestos (*) | | | | | | | R | | | R | CRUD | CRUD | | | | | | | | | | | R | R | R | | | |
| Registro de Remisiones (*) | R | R | R | | | | R | | | R | | | | | | | CRUD | | | | | | R | R | R | | | |
| Registro de Succiones | R | R | R | | | | R | | | R | | | | | | | | CRUD | | | | | R | R | R | | | |
| Emisión de Facturas (*) | | R | | | | R | | R | RU | R | RU | RU | RU | | | | RU | RU | CRUD | CRUD | CRUD | CRUD | | | | R | R | R |

5. Integrar el uso global de los datos con el perfil de procesos - datos.

La integración de los datos con la organización es muy importante ya que en un ambiente distribuido, los datos son compartidos y fluyen a través de las líneas de comunicación del negocio. En otras palabras los datos tienen un uso global. Durante esta actividad los flujos de datos desde y hacia la aplicación son identificados y se negocian aquí los niveles de servicio. Al concluir este punto, se tiene que haber integrado el uso global de los datos, con el perfil de uso de procesos – datos de las aplicaciones y además haber negociado tanto con los desarrolladores así como con los administradores de datos los niveles de servicios requeridos.

Para este punto podemos tener presentes además del perfil de uso de procesos – datos los siguientes elementos:

- El conocimiento de las demandas externas de datos desde y hacia las aplicaciones.
- Reuniones con los equipos de desarrollo y con los administradores de datos, los cuales pueden ser dueños de los datos a los cuales la aplicación necesite accesar o bien puede necesitar acceso a los datos que su aplicación cree o modifique.

Con los grupos de administradores, desarrollo y arquitectos; son con los que debe negociarse todo aquello relacionado a los datos necesarios para la entrada, entre estos datos podemos listar:

- Identificación de los datos requeridos.
- Identificación de la fuente primaria para cada dato.
- Identificación de los niveles de cambio para cada dato.
- Acuerdos sobre los requerimientos en los niveles de servicio.

Con relación a los datos requeridos como salida de las aplicaciones:

- Identificación de los datos requeridos.
- Identificación de el número de procesos y/o bases de datos que requieren algún evento de notificación.
- Identificación de los niveles de cambio típicos para cada elemento de datos aplicados. Si los niveles de cambio son muy altos entonces deben contemplarse mecanismos de monitoreo para alertar sobre las caídas de estos nodos de replicación.
- Acordar los niveles de servicio requeridos para todas las aplicaciones involucradas.

De acuerdo a lo mencionado anteriormente y como parte del ejemplo que se ha desarrollado a lo largo de este trabajo, podemos decir que esta integración dio como resultado lo siguiente:

Se determinó que para Oficinas Generales de ASA es suficiente contar con un servicio de lunes a viernes en un horario de 8:00AM a 7:00PM y los sábados de 8:00 AM a 3:00 PM. Para el caso de los aeropuertos y de las estaciones de combustible, se determinó que se necesita replicar información solo a aquellos sitios que por el volumen de información que generan necesitan contar con un servicio de 7x24x365. Así se determinó que las siguientes estaciones de combustible, que se encuentran en aeropuertos que no son administrados por ASA y las cuales requieren de este servicio son:

- Tijuana
- Cancún
- México
- Guadalajara
- Monterrey
- Acapulco
- Mérida
- , Puerto Vallarta
- Los Cabos
- Mazatlán

Mientras que de los aeropuertos que son administrados por ASA los que requieren un servicio similar son:

- Toluca
- Puebla
- Cuernavaca

El resto de los aeropuertos puede operar directamente conectado a los servidores centrales ya que el volumen de información que concentran es mínimo y sus horarios se pueden ajustar al de Oficinas Generales. Así, el esquema de replicación que se diseñe obviamente será solo para las estaciones de combustible y aeropuertos mencionados previamente.

Adicionalmente se determinó en base a la Tabla III.1. Perfil para la distribución de la aplicación y la Tabla III.4. Matriz CRUD lo siguiente:

Las Oficinas Generales de ASA serán los sitios primarios para las siguientes tablas:

a) asa_c_avion

c) asa_c_matricula

d) asa_c_pais

z) fac_c_tarifa

aa) fac_c_tarcom

bb) fac_c_tartua

Lo anterior en virtud de que Oficinas Generales es la única que realiza el proceso mantenimiento a catálogos, lo cual implica la realización de transacciones según se observa en la Matriz CRUD, en dónde estas tablas están marcadas como CRUD para este proceso. Sin embargo, estas tablas son requeridas en otros procesos (pero solo para lectura), mismos que no se realizan en Oficinas Generales, tal es el caso de la Planeación de Vuelos, Registro del Movimiento Operacional, Registro de Remisiones, Registro de Succiones y Emisión de Facturas[14]. Por lo que la información de estas tablas se requiere en los aeropuertos y estaciones de combustible, de esta forma dichas tablas deberán ser replicadas a estos sitios. Cabe destacar que no se menciona el proceso Autorizaciónes de Crédito, en la lista de aquellos que requieren la información, debido a que el proceso es realizado exclusivamente por Oficinas Generales, por lo cual los datos se encuentran en el mismo sitio en dónde se realiza este proceso.

Situación similar se observa con las tablas relacionadas a Mantenimiento de Contratos, Autorizaciones de Crédito.

Esto se puede resumir en la siguiente tabla:

| TABLA | Oficinas Generales | Aeropuertos | Estaciones de Combustible |
|---|---|---|---|
| a) asa_c_avion | Primaria | Replicada | Replicada |
| b) asa_c_cliente | P | R | R |
| c) asa_c_matricula | P | R | R |
| d) asa_c_pais | P | R | R |
| f) asa_d_clidom | P | R | R |
| g) asa_d_conxmat | P | R | R |
| j) asa_m_contrato | P | R | R |
| w) scf_d_autorizacion | P | R | R |
| x) scf_d_lugxaut | P | R | R |
| y) scf_m_autorizacion | P | R | R |
| z) fac_c_tarifa | P | R | R |

[14] Veáse que en la Tabla III.1. Perfil para la distribución de la aplicación, la expedición de facturas se realiza tanto en ASA como en Estaciones de Combustible e inclusive el los aeropuertos.

| TABLA | Oficinas Generales | Aeropuertos | Estaciones de Combustible |
|---|---|---|---|
| aa) fac_c_tarcom | P | R | R |
| bb) fac_c_tartua | P | R | R |

Ahora bien, si realizamos las actividades anteriores pero con el resto de los procesos podremos determinar la siguiente tabla:

| TABLA | Oficinas Generales | Aeropuertos | Estaciones de Combustible |
|---|---|---|---|
| e) asa_d_abordador | | P | |
| h) asa_d_excxreg | R | P | |
| i) asa_m_abordador | $R_1/P_2$ | $P_1/R_2$ | |
| k) asa_m_manifa | | P | |
| l) asa_m_manifs | $R_1/P_2$ | $P_1/R_2$ | |
| m) asa_m_registro | $R_1/P_2$ | $P_1/R_2$ | |
| n) cco_d_itinerario | | P | |
| o) cco_d_planeacion | | P | |
| p) cco_m_itinerario | | P | |
| q) cmb_m_remision | $R_1/P_2$ | | $P_1/R_2$ |
| r) cmb_m_succion | $R_1/P_2$ | | $P_1/R_2$ |
| s) fac_d_facconc | $R_{12}$ | $P_1$ | $P_2$ |
| t) fac_d_factura | $R_{12}$ | $P_1$ | $P_2$ |
| u) fac_h_cierre | $R_{12}$ | $P_1$ | $P_2$ |
| v) fac_m_factura | $R_{12}$ | $P_1$ | $P_2$ |

Como se puede observar las tablas cco_d_itinerario, cco_m_itinerario y cco_d_planeacion; no son requeridas en otros sitios por lo cual no se requiere replicación de las mismas, es decir, son utilizadas exclusivamente por los aeropuertos.

Todas aquellas tablas que se marcaron como **R**eplicadas en Oficinas Generales, es debido a que la información que es capturada en los aeropuertos y/o estaciones de combustibles para que sea facturada en Oficinas Generales, por eso el aeropuerto es sitio primario y Oficinas Generales sitio replicado. Cuando la información es facturada en las Oficinas Generales, los registros de estas tablas se marcan para identificar que ya fueron facturados y de esta forma ese cambio necesita regresar al sitio original, es por eso que estas tablas actuan ahora como primarias OG, y replicadas en aeropuertos y estaciones de combustible.

Conjuntando la información de las dos tablas anteriores obtenemos una matriz a la cual le llamaremos Matriz de Uso Global[15], y lo cual se presenta a continuación:

**Tabla III.5. Matriz de Uso Global de Datos**

| TABLA | Oficinas Generales | Aeropuertos | Estaciones de Combustible |
|---|---|---|---|
| a) asa_c_avion | P | R | R |
| b) asa_c_cliente | P | R | R |
| c) asa_c_matricula | P | R | R |
| d) asa_c_pais | P | R | R |
| e) asa_d_abordador | | P | |
| f) asa_d_clidom | P | R | R |
| g) asa_d_conxmat | P | R | R |
| h) asa_d_excxreg | R | P | |
| i) asa_m_abordador | $R_1/P_2$ | $P_1/R_2$ | |
| j) asa_m_contrato | P | R | R |
| k) asa_m_manifa | | P | |
| l) asa_m_manifs | $R_1/P_2$ | $P_1/R_2$ | |
| m) asa_m_registro | $R_1/P_2$ | $P_1/R_2$ | |
| n) cco_d_itinerario | | P | |
| o) cco_d_planeacion | | P | |
| p) cco_m_itinerario | | P | |
| q) cmb_m_remision | $R_1/P_2$ | | $P_1/R_2$ |
| r) cmb_m_succion | $R_1/P_2$ | | $P_1/R_2$ |
| s) fac_d_facconc | $R_{12}$ | $P_1$ | $P_2$ |
| t) fac_d_factura | $R_{12}$ | $P_1$ | $P_2$ |
| u) fac_h_cierre | $R_{12}$ | $P_1$ | $P_2$ |
| v) fac_m_factura | $R_{12}$ | $P_1$ | $P_2$ |
| w) scf_d_autorizacion | P | R | R |
| x) scf_d_lugxaut | P | R | R |
| y) scf_m_autorizacion | P | R | R |
| z) fac_c_tarifa | P | R | R |
| aa) fac_c_tarcom | P | R | R |
| bb) fac_c_tartua | P | R | R |

[15] Esta Matriz de Uso Global no es parte de la metodología de Marie Buretta, sin embargo, derivado de la experiencia personal, es de gran utilidad al momento de la implementación. .

En esta matriz es posible identificar no solo los sitios primarios y replicados de cada tabla, si no que también permite identificar grupos de tablas que tengan un comportamiento similar y a los cuales se les dará el nombre de casos durante la implementación.

### 6. Esquema de distribución de los datos

El esquema de distribución orienta no solo los flujos de datos intraaplicaciones (aquellos que son internos a la aplicación) sino también los flujos de interaplicaciones (aquellos asociados con fuentes externas primarias y destinos replicados). Este primer paso en la distribución de datos llegará a ser muy significativo y este se propone sea validado en pasos siguientes con respecto a las restricciones, capacidades, tecnologías y requerimientos en niveles de servicio.

Esta actividad se realiza utilizando los productos resultados de los pasos anteriores como son:

- Perfil de distribución de aplicaciones[16].
- Perfil de infraestructura física[17].
- Listado de grupos de recuperación[18].
- Perfil de uso de procesos - datos[19].
- Acuerdos sobre los niveles de servicio.
- Conocimiento de la directriz estratégica de la compañía con respecto a las tecnologías preferidas.
- La matriz del uso global de los datos.

Todo esto ayudará a formular el esquema de distribución de datos, si existen algunos puntos de los pasos anteriores que no se han completado o no están muy claros, este es el momento adecuado para corregir estos puntos.

A pesar de que la matriz del uso global de los datos nos permite tener una visión global sobre de cómo se utilizan los datos con relación a los sitios existente, el esquema de distribución debe orientarse en primera instancia a los datos transaccionales creados y modificados por las aplicaciones, siguiendo los pasos que se listan a continuación:

1. Identificar lugares viables para la distribución. Utilizar el perfil de distribución de aplicaciones y el perfil de la infraestructura física para auxiliar en las identificación de procesos.

[16] Tabla III.1. Perfil para la distribución de la aplicación presentada en la página 46 del presente.
[17] Tabla III.2. Perfil de la infraestructura física presentada en la página 49.
[18] Tabla III.3. Agrupación Lógica de Entidades presentada en la página 46.
[19] Tabla III.4. Matriz CRUD presentada en la página 59.

2. Consolidar la selección de un modelo de distribución. Si no se ha seleccionado el modelo de distribución deberá seleccionar uno de ellos sin olvidar que estos modelos son solo plantillas que pueden ser adaptadas o adoptados.

3. Distribuir los grupos de datos identificados, considerando las siguientes premisas:

   - Colocar los datos cerca de los puntos de uso máximo.
   - Si la misma instancia de la entidad de datos se actualizará de diversos lugares, será necesario considerar la centralización de datos.
   - Si el mismo tipo de entidad (no instancia) será actualizada de diversos lugares, debe considerarse la partición vertical.
   - Si la entidad de datos existe en el lugar, considerar compartir la copia existente.

Será necesario tener presente que entre más oportunos sean requeridos los datos, en menor número de lugares deben distribuirse estos datos. Por el contrario, si los datos no se necesitan con mucha oportunidad, entonces mayor puede ser el número de lugares en los que se pueden distribuir estos datos.

Una vez que se ha determinado el lugar para los datos transaccionales, el siguiente paso consiste en decidir la ubicación de los datos de solo lectura (o de referencia) utilizados por la aplicación. Si los requerimientos de latencia en este tipo de datos es cero, esto es, tiempo real, entonces las aplicaciones deberían accesar a la información directamente a la copia primaria. Si la latencia requerida es cercana al tiempo real, esto es, de unos cuantos segundos a unas cuantas horas, entonces la replicación de estos datos es posible, ubicándolos en los lugares en dónde serán utilizados por medio de una replicación asíncrona *base de datos–a–base de datos* o *proceso–a–proceso*. Si los requerimientos de latencia son mayores a ocho horas, entonces la replica de estos datos puede realizarse por medio de una actualización completa o incremental vía replicación o mediante una propagación de eventos delta.

Después de haber establecido los datos transaccionales o de referencia, el siguiente paso es identificar los sitios destino que recibirán los datos. Si es posible, se recomienda identificar las decisiones de colocación gráficamente. Muestre los lugares, los datos que serán almacenados en cada lugar, y los anchos de banda que existen entre los nodos de la red.

Una vez que la representación gráfica está completa, realice una estimación del volumen en los flujos de datos que existirán a través de la red. Esto solo será un análisis que nos dará un diseño preliminar y en el cual se deben incluir:

- Flujos de datos transaccionales. Para los procesos "pesados" marcados en el perfil de uso de procesos - datos, estimar los picos que pueden presentarse en el tráfico de datos entre nodos. Esto implica estimar la cantidad de datos enviados o recibidos de o hacia un sitio durante la ejecución normal de las aplicaciones.

- Flujos de datos replicados. Para la replicación asíncrona que provee datos de o hacia algún sitio, se debe de igual forma, estimar los picos que pudieran presentarse entre nodos. Esto conlleva a determinar la razón de cambió para los datos primarios y la tasa de envío para los procesos "pesados". Deberá sumar este valor al de flujos de transacciones calculados anteriormente.

- Factor de error. Para ajustar, las cantidades obtenidas a factores de error, multiplicar los valores anteriores por un factor de 2. O si no se está satisfecho puede utilizarse un factor de 2.5 o 3 con los valores estimados. Este último valor que se obtenga será el tráfico que debe soportar como mínimo nuestra infraestructura física para que no se presenten problemas cuando existan picos en la actividad diaria.

En la Figura III.1. Esquema gráfico de la distribución de datos. se propone una representación gráfica para nuestro esquema de replicación. Solo será necesario complementar este esquema con una descripción de los volúmenes de información que fluirán a través de la red digital.

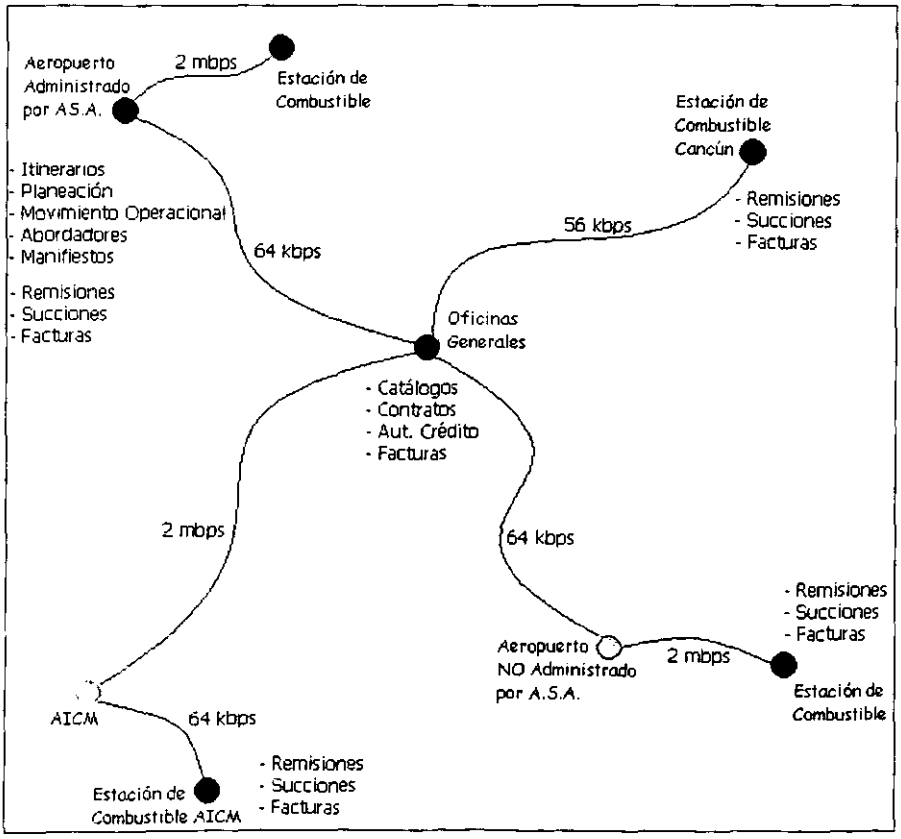



**Figura III.1. Esquema gráfico de la distribución de datos.**

El punto foráneo que genera más información dentro del esquema de replicación es la estación de combustibles del AICM, de tal forma que con realizar una estimación para esta, podemos validar si el ancho de banda será suficiente para el resto de los aeropuertos. Por otra parte, el punto que generará más información aún se encuentra en Oficinas Generales. Así, con realizar una estimación de estos dos puntos podremos medir nuestros requerimientos contra los recursos disponibles.

7. Validar el esquema de distribución contra las restricciones existentes así como con las capacidades y tecnologías.

En este punto se debe asegurar que la infraestructura física pueda soportar el esquema propuesto de distribución de datos. De ser así, el esquema podrá operar adecuadamente y entonces podemos continuar con el siguiente punto que se refiere a una validación del rendimiento. En caso contrario, debemos corregir o eliminar alguna restricción, mejorar la capacidad de la infraestructura física existente y/o mejorar la funcionalidad de las tecnologías existentes.

Para realizar esta actividad se debe verificar cada nodo dentro del esquema con respecto a todas y cada una de las restricciones, limitaciones de capacidad y a cualquier requerimiento tecnologico específico.

Cuando se evalúe un nodo del esquema con respecto a alguna restricción podemos utilizar el perfil de infraestructura física y el esquema de distribución de datos para garantizar los siguientes puntos:

- Soporte a las plataformas existentes.
- Soporte al nivel de seguridad.
- Nivel de soporte adecuado y disponible.
- Cualquier restricción listada en el perfil de la infraestructura física pueda ser resuelto.

Al evaluar un nodo con relación a alguna limitación en cuanto a capacidad se debe garantizar:

- Que no existe ninguna limitación por parte de la plataforma. Se debe validar que la cantidad de datos y el número de usuarios concurrentes son soportados por las plataformas disponibles.
- Que no existe ninguna limitación a nivel base de datos. Validar que la cantidad de datos así como los tipos de datos requeridos y el número de usuarios concurrentes son soportados por el DBMS.
- Que el ancho de banda existente entre los nodos será suficiente para soportar el tráfico de datos.

El resultado de este punto será una lista que contenga todas las restricciones y limitaciones tanto de capacidad como de tecnología existentes con relación al esquema de replicación para así poder

tomar la decisión si se regresa a punto anterior y se plantea un esquema de distribución alternativo o si se solventa la lista resultado de esta actividad.

Para el caso de ASA solo se detecta que la conexión hacia la estación de combustibles de Cancún es vía módem pero esto de resuelve con herramientas de monitoreo propias del Replication Server que se encarga de verificar que los nodos de replicación permanezcan activos.

8. Validar el esquema de distribución contra los requerimientos de nivel de servicio.

El objetivo de este punto es garantizar que el esquema de replicación de datos que se ha propuesto así como la infraestructura física existente podrán soportar los requerimientos en cuanto al rendimiento acordado para lograr el nivel de servicio requerido. De ser así, entonces el esquema de replicación es funcional y se puede comenzar con su implementación. En caso contrario, se procede a modificar el esquema de distribución, mejorar la infraestructura existente o negociar los niveles de servicio requeridos.

9. Implementar el esquema de distribución.

Una vez que el esquema propuesto ha sido validado, corresponde continuar con la implementación de dicho esquema. Esta implementación será desarrollada en el capítulo IV de este trabajo.

## Consideraciones sobre el Replication Server

La metodología anterior nos da como resultado un esquema de replicación a nivel conceptual, mismo que se plantea en forma general, es decir, no se orienta a alguna herramienta específica, así que tendremos que precisar algunas limitaciones que Replication Server como cualquier otra herramienta de software posee para que se tenga presente durante el desarrollo del mismo.

1. Replication Server solo puede filtrar los registros, basándose en la información que estos mismos poseen, es decir, no puede realizar filtros en base a columnas de otras tablas o en base a datos de otras filas.

2. Considerar una replicación con un máximo de dos niveles. Aunque Replication Server puede soportar hasta tres niveles de replicación, es muy recomendable solo considerar dos. Ya que un tercer nivel implica un análisis y diseño más complejo

## Recomendaciones para crear un esquema de replicación

El diseño de bases de datos en un ambiente distribuido resulta ser significativamente más complejo que un ambiente centralizado ya que deben tomarse en cuenta parámetros tales como: las características de la red, la distribución de la información, los datos redundantes, así como las limitaciones del Replication Server.

En la medida de lo posible es muy recomendable diseñar el esquema de la base de datos orientado al ambiente de replicación, ya que de lo contrario será posible realizar esta replicación

correctamente pero podría llegar a implicar mayor tiempo de diseño, implementación y administración del mismo.

Estas recomendaciones son las siguientes:

- En las bases de datos replicadas activar la integridad referencial por medio de triggers
- Se recomienda trabajar con tipos de datos del sistema y no tipos de datos definidos por el usuario.
- Si se requiere contar con llaves únicas que se generen automáticamente a lo largo de todas las bases de datos que formen parte del esquema de replicación, se recomienda:

    1. Agregar un campo que identifique el origen.
    2. Asignar un rango de números a cada sitio.
    3. Utilizar un mecanismo externo para generar la llave única en todo el esquema.

- Es de gran ayuda incluir campos identificadores de sitio primario que indiquen el lugar en dónde el dato fue capturado inicialmente, esto principalmente en los modelos consolidados corporativos.

Al concluir este capítulo se ha mostrado la metodología que nos ayudará para identificar los origenes y destinos de cada una de las tablas participantes dentro de la replicación de datos, sin embargo, el proceso mismo de implementación puede ser confuso y complicado, sobre todo en relación a la creación de los scripts que serán utilizados en el Replication Server, por lo cual, el siguiente capítulo nos ayudará a concluir con la implemtación de lo ya diseñado en la "Figura III.1. Esquema gráfico de la distribución de datos.".

# CAPITULO IV. PROGRAMA GENERADOR DE SCRIPTS DE REPLICATION SERVER (GSRS)

*OBJETIVO: Presentar el programa generador de scripts de replicación mismo que facilitará la generación de estos scripts para agilizar el proceso de implementación de un esquema de replicación.*

Dentro de la implementación de un esquema de replicación, Sybase ofrece soluciones gráficas por medio de las cuales se pueden crear replication definitions o subscripciones a las mismas, sin embargo, en la práctica esto puede resultar repetitivo y hasta cierto punto tedioso, de allí que lo que se acostumbra usualmente, es generar scripts[20] que permitan configurar un ambiente de replicación mediante la ejecución de procesos en lote y que además puedan ser reutilizados por si se requiere de múltiples sitios o por sí se llega a dañar el ambiente de replicación.

Además, como parte de este capítulo, se describirán las instrucciones que estos scripts comúnmente incluyen para posterior a ello especificar de que manera puede automatizarse el proceso para la creación de los mismos. Sin embargo, es necesario establecer las actividades requeridas para realizar la implementación de la replicación de un esquema cualquiera.

## Implementación del Esquema

La implementación de un esquema de replicación con Replication Server, puede ser considerada por medio de las siguiente actividades:

1. Instalación y configuración de los Replication Server a utilizar.
2. Creación de rutas desde los Replication Server primarios a los replicados.
3. Diseño y creación de scripts a utilizar en Replication Server.
4. Ejecución de los scripts.
5. Validación de la implementación.

### Instalación y configuración de los Replication Server

Dado que Replication Server es un software, será necesario en primera instancia instalarlo en cada sitio que vaya a participar dentro del esquema de replicación. Este proceso de instalación y configuración no será descrito ya que el mismo puede ser encontrado en los manuales de

[20] Se utilizará el término ***script*** para identificar a un archivo de texto cuyo contenido son comandos de algún producto de software especifico, y que este caso serán comandos de Replication Server o de un DBMS, lo anterior en virtud de que el término ***script*** se traduce al español como manuscrito, escritura, guión, libreto, escribir el guión para una película.

instalación del producto, sin embargo, si es posible realizar recomendaciones para su instalación y configuración. Algunas de ellas son:

- Es altamente recomendable instalar un Replication Server por cada punto dentro de la WAN, es decir, si se tiene DBMS dentro de una WAN, será suficiente un solo Replication Server para recibir y distribuir las transacciones entre esos DBMS. Lo anterior a menos que las bases a datos a incluir dentro del esquema de replicación sean altamente transaccionales, entonces considerar la posibilidad de instalar más de un Replication Server para cada sitio.

- En el servidor dónde se instale el Replication Server es conveniente contar con un Adaptive Server Enterprise el cual contenga solo la base de datos propia del Replication Server, es decir, el diccionario datos que este utiliza.

- Si se tiene un solo nodo central y una cantidad considerable de nodos replicados, y además estos son altamente transaccionales, entonces considerar incluir Replication Server intermedios que disminuyan la carga del nodo central.

### Creación de rutas desde los Replication Server primarios a los replicados

Cuando ya se han instalado y configurado los Replication Server y las bases de datos que forman parte del esquema de replicación, es necesario especificar las rutas por medio de las cuales se transmitirá la información, es decir, tenemos que indicarle al Replication Server que existe una ruta del $RS_A$ hacia el $RS_B$, es decir, que se puede replicar información desde $RS_A$ hacia $RS_B$. Lo anterior no implica la replicación de $RS_B$ a $RS_A$, ya que para lograr esto se requiere definir la ruta en este sentido.

El comando de Replication Server para definir la ruta es:

```
CREATE ROUTE TO RSB
```

Obviamente se ejecuta desde el $RS_A$.

### Diseño y creación de scripts

El diseño y creación de los scripts suele ser una actividad sumamente laboriosa ya que para cada tabla primaria se requiere crear un replication definition, misma que como se verá más adelante implica practicamente reescribir la estructura de dicha tabla. Así, que si tenemos una base de datos con decenas de tablas, entonces esta actividad representará un gran trabajo tan solo para crear las replication definitions, que ciertamente es el noventa porciento de la creación de los scripts y que además se presta a errores de captura. Es por eso que más adelante se explica el programa GSRS, mismo que tiene como objetivo crear estos scripts.

### Ejecución de los Scripts

Los scripts de Replication Server serán ejecutados de la siguiente forma:

```
isql -Uusuario -Ppassword -SREPSERVER -iscript.sql
```

En dónde *usuario* y *password* corresponden a la identificación del usuario y contraseña respectivamente del *REPlicationSERVER* sobre el cual se van a ejecutar los scripts.

### Validación del esquema

Para realizar la validación del esquema es necesario verificar que los scripts no hayan generado error alguno y posterior a ello es altamente recomendable realizar pruebas de replicación, es decir, insertar, actualizar y eliminar registros a diferentes tablas que esten replicando para verificar que las transacciones son replicadas a los servidores que esten subscritos.

Antes de que comencemos a ver como se construyen los scripts, se presentan los comandos del Replication Server más comúnmente utilizados dentro de los mismos.

## Comandos del Replication Server

Las instrucciones que comúnmente se utilizan para configurar un ambiente de replicación son las siguientes:

### create replication definition

Este comando como su nombre lo indica, crea una replication definition[21] dentro del Replication Server.

Sintaxis:

```
 create replication definition replication_definition
 with primary at data_server.database
[with all tables named [table_owner.] 'table_name' |
[with  primary table named [table_owner.]'table_name']
 with  replicate table named [table_owner.]'table_name']]
(
 column_name [as replicate_column_name] datatype [null | not null]
             [map to [published_datatype]]
[,column_name [as replicate_column_name] datatype [null | not null]]
             [map to [published_datatype]]  )
 primary key (column_name [, column_name]...)
[searchable_columns (column_name [, column_name]...)]
[send  standby [{all | replication definition} columns]]
[replicate {minimal | all} columns]
[replicate_if_changed (column_name [, column_name]...)]
[always_replicate  (column_name [, column_name]...)]
```

Parámetros:

[21] Vease Replication Definition en la página 29.

replication_definition.- Este parámetro indica el nombre con el cual se identificará a la replication definition dentro del ambiente del Replication Server. Este nombre debe se único.

**with primary at** data_server.database.- Especifica la ubicación de los datos primarios[22], es decir, se indica en que servidor y base de datos se encuentra la tabla que será replicada.

**with all tables named**.- Indica el nombre de la tabla primaria y replicada.

**with primary table** named.- Indica el nombre de la tabla en la base de datos primaria.

**with replicate table** named.- Indica el nombre de la tabla en el base de datos replicada.

column_name.- Este parámetro representa el nombre de una columna dentro de la tabla primaria, solo aquellas columnas que se incluyan dentro de la replication definition serán replicadas.

**as** replicate_column_name.- Especifica el nombre de una columna en la tabla replicada en la cual el dato de la columna primaria será copiada. Esta cláusula se utiliza cuando el nombre de las columnas en la tabla primaria y replicada son diferentes.

datatype.- El tipo de dato de la columna en la tabla primaria.

**null / not null**.- Esta cláusula aplica solo para columnas del tipo imagen o texto, y representa si la columna permiten o no un valor nulo. Es muy importante que esta propiedad coincida con la tabla primaria.

**map to** published_column_name.-

**primary key** column_name.- En esta sentencia se indica las columnas que conforman la llave primaria dentro de la tabla primaria y replicada. Todas y cada una de las columnas que aquí se especifiquen deben haber sido declaradas en la lista de columnas.

**searchable columns** column_name.- Especifica las columnas que pueden utilizarse al momento de realizar subscripciones en las cláusulas where de los comandos **create subscription** o **create article**.

**send stand by**.- Especifica como se utilizará la replication definition cuando se replica a una base de datos stand by dentro de un modelo warm stand by[23]

**replicate minimal columns / replicate all columns**.- Envía todas las columnas de la replication definition en cada transacción o solo aquellas que son necesarias para realizar las operaciones de actualización y borrado[24] (update o delete) en la base de datos replicada.

[22] Véase Bases de datos primarias en la página 29.
[23] Véase Modelo WARM-Standby en la página 41.
[24] Véase UPDATE y DELETE en la página 15.

**replicate_if_changed**.- Replica columnas de texto o imagen solo cuando los datos de la columna cambian.

**always_replicate**.- Replica siempre las columnas de imagen o texto.

Este comando resulta ser el más complejo durante la implementación de un esquema de replicación y va directamente relacionado con el Esquema de Distribución de Datos[25] planteado en el capítulo anterior.

DROP REPLICATION DEFINITION

Este comando elimina una replication definition del ambiente de replicación.

Sintaxis:

```
drop replication definition replication_definition
```

Parámetros:

replication_definition.- Es el nombre de la replication definition a eliminar.

A pesar de que en la práctica el eliminar un replication definition suele ser poco común, es muy conveniente contar con los scripts para el borrado de las replication definitions por si se necesita desconectar a algún sitio del esquema. Es muy importante considerar que para eliminar una replication definition debe no existir subscripción alguna a ella.

create subscription

Crea e inicializa una subscripción así como la materialización de los datos. Esta subscripción será a una replication definition, function replication definition[26] o publicación.

Sintaxis:

```
 create subscription sub_name
 for {table_rep_def |
      function_rep_def |
      publication pub_name with primary at data_server.database}
 with  replicate at data_server.database
[where {column_name | @param_name} {< | > | >= | <= | = | &} value
  [and {column_name | @param_name} {< | > | >= | <= | = | &} value]...]
[without  holdlock | incrementally |without  materialization]
[subscribe to truncate table]
[for new articles]
```

[25] Véase Replication Definition en la página 29 ya que cada tabla primaria requerirá de una replication definition.
[26] Function replication definition.- Es una variación de replication definition aplicada a procedimientos almacenados (store procedures), sin embargo, esta variación no se menciona en el presenta trabajo por ir más allá de los objetivos del mismo.

Parámetros:

sub_name.- Nombre de la subscripción, mismo que debe ser único por cada replication definition.

**for** table_rep_def.- Especifica la replication definition para la cual se crea la subscripción.

**for** function_rep_def.- Especifica la function replication definition para la cual se crea la subscripción.

**for publication** pub_name.- Especifica la publicación para la cual se crea la subscripción.

**with primary at** data_server.database.- Indica la ubicación de los datos primarios.

**with replicate** at data_server.database.- Indica la ubicación de los datos replicados.

**where**.- Establece el los criterios para una columna que deben ser cumplidos para que los datos se repliquen del lugar primario al lugar replicado. Una cláusula where puede ser compuesta de una o más comparaciones sencillas en las cuales una searchable column o serchable parameter[27] especificados en una replication definition son comparados con valores literales por medio de operadores relacionales: <, >, <=, >=, = o & (El operador & solo funciona con el tipo de datos rs_addres mismo que no se maneja en este documento por no ser un trabajo para conocer a fondo el Replication Server). Además es posible unir las comparaciones por medio de operadores lógicos **and**.

column_name.- Una columna de la tabla primaria.

@param_name.- El nombre de un parámetro de un procedimiento almacenado replicado.

value.- Un valor para una columna o parámetro especifico

**whitout holdlock**.- Selecciona los datos de la base de datos primaria sin realizar un bloqueo sobre los datos. Los renglones son aplicados en la base de datos replicada en transacciones de 10 registros (materialización no atómica).

**incrementally**.- Inicializa la subscripción y aplica los datos en transacciones de 10 registros cada una, pero mantiene un bloqueo de los datos primarios para realizar una materialización atómica.

Este comando es el complemento a create replication definition, ya que con el comando create replication definition, se especifican las características de la tabla que será replicada, y por medio de create subscription se define que sitios están conectados a la replication definition.

[27] Serchable parameter se utiliza para function replication definition mismo que como ya fue comentado en las páginas anteriores no se incluye en este trabajo. Si se desea mayor referencia al respecto se recomienda revisar el Manual de Referencia de Replication Server.

drop subscription

Este comando tiene como finalidad el eliminar una subscripción desde un sitio replicado a un sitio primario, esto implica que la replicación del primario al replicado se suspenda definitivamente para la(s) tabla(s) involucradas.

```
drop subscription sub_name
 for {table_rep_def | function_rep_def |
       {article article_name in pub_name |
        publication pub_name}
       with primary at data_server.database }
with  replicate at data_server.database
[without  purge
 [with suspension [at active  replicate only]] |
 [incrementally] with purge]
```

create publication

Crea una publicación pub_name cuyo sitio primario será *data_server* y la base de datos primaria será *database*.

```
create publication pub_name with  primary at data_server.database
```

create article

Por medio de este comando le indicamos a Replication Server que debe incluir dentro de una publicacion un nuevo articulo relacionado con una replication definition y en el cual pueden incluirse condiciones para filtrar la información.

```
create article article_name
for pub_namewith
primary at data_server.database
with  replication definition {table_rep_def |
                              function_rep_def}
[where {column_name | @param_name}
       {< | > | >= | <= | = | &} value  [and {column_name | @param_name}
       {< | > | >= | <= | = | &} value]...
  [or  where {column_name | @param_name}
       {< | > | >= | <= | = | &} value  [and {column_name | @param_name}
       {< | > | >= | <= | = | &} value]...]...]
```

drop article

Elimina un articulo que se ha definido dentro de una publicación.

```
drop article article_name
for pub_namewith
primary at data_server.database
[drop_repdef]
```

validate publication

Este comando tiene como finalidad el validar que el estado de una publicación es correcto.

```
validate publication pub_namewith
primary at data_server.database
```

drop publication

Drop publication da la posibilidad de eliminar una publicación que ya no esté en uso.

```
drop publication pub_namewith
primary at data_server.database
[drop_repdef]
```

## Generación de Scripts

Si observamos detenidamente las sintaxis de los comandos que se han mencionado anteriormente, podemos percatarnos que la labor para crear estos scripts como ya se ha mencionado, es altamente repetitiva y esta puede ser automatizada inclusive con el uso de procesadores de texto, sin embargo, el caso del primer comando es muy especial ya que dentro de él se debe incluir la lista de las columnas que forman parte de la replication definition (usualmente son todas las de la tabla), de tal forma que si tenemos una tabla de productos que fue creada mediante la siguiente instrucción:

```
CREATE TABLE productos
(
  prodid           numeric(10)   not null,
  categid          integer       not null,
  descripcion      varchar(255)  not null,
  unidad           varchar(15)   null,
  preciounitario   numeric(12,2) not null,
  existencia       numeric(8)    not null,
  tipocosteo       char(4)       not null,
  fechaespera      datetime      null,
  existenciaespera numeric(8)    null,
  PRIMARY KEY (prodid)
)
```

Una replication definition válida pudiera ser:

```
CREATE REPLICATION DEFINITION ORIGENproductos
WITH PRIMARY AT DBMS.BASE_DATOS
WITH ALL TABLES NAMED 'productos'
(
  prodid           numeric,
  categid          integer,
  descripcion      varchar(255),
  unidad           varchar(15),
```

```
  preciounitario   numeric,
  existencia       numeric,
  tipocosteo       char(4),
  fechaespera      datetime,
  existenciaespera numeric
)
WITH PRIMARY KEY (prodid)
SEARCHABLE COLUMS (prodid,categid)
```

Como se puede observar no existe mucha diferencia entre el comando SQL CREATE TABLE y el comando de Replication Server CREATE REPLICATION DEFINITION. Sin embargo, ambos comandos operan sobre ámbitos y tiempos diferentes, es decir, uno se utiliza para crear la tabla dentro de la base de datos y el segundo tiene como función indicarle a Replication Server que replicará una estructura de datos con los campos y longitudes que se especifican.

Lo anterior pudiera parecer de poca relevancia, sin embargo, si fuera necesario crear replication definitions de 300 tablas y cada tabla en promedio posee 15 campos, entonces sería necesario que se escribiera de forma totalmente manual un script con por lo menos 4500 campos, mismos que debieran ser verificados y validados para no causar problemas posteriores.

Ahora bien, si fuera posible tener acceso a la estructura de la tabla productos mediante comandos SQL, la propia base de datos no daría la oportunidad de poder automatizar la creación de dichos scripts, y si esto se lleva a un ambiente de un programa cliente servidor, entonces podemos automatizar la creación de los scripts. Lo anterior fue la base del programa que permite agilizar el proceso de implementación de un esquema de replicación basado en Replication Server.

Y si este programa programa como ya se mencionó, es capaz de generar los scripts para las replication definitions que son el comando más laborioso, entonces es posible considerar que este mismo programa genere los scripts para el resto de las instrucciones con lo cual seremos capaces de generar dichos scripts y solo bastará ejecutarlos para crear un esquema funcional de replicación basado en Replication Server.

## Bases del Generador de Scripts de Replication Server (GSRS).

En el tema anterior se menciona la idea del programa GSRS, cuyo diseño se realizará para trabajar en el ambiente Microsoft Windows, mismo que mediante una interfaz gráfica permitirá dar mantenimiento a las plantillas que sirvan como base para crear los scripts de replicación. Es decir, si fuera necesario crear una replication definition, podríamos contar con una plantilla como la siguiente:

```
CREATE REPLICATION DEFINITION %ORIGEN%%table_name%
WITH PRIMARY AT %PDS%.%PDB%
WITH ALL TABLES NAMED %table_name%
(
  %column_list%
)
WITH PRIMARY KEY (%pk_column_list%)
SEARCHABLE COLUMNS (%pk_column_list%)
```

```
go
```

En esta plantilla podemos decir que todo identificador delimitado en ambos lados por el signo de %, se considerará como una variable que será reemplazada por un valor *x* al momento de generar los scripts. Por ejemplo, si tenemos las siguientes tablas en la base de datos:

| asa_c_aeropuerto | | | |
|---|---|---|---|
| pk | asa_c_aeropuerto | char(3) | not null |
| | ciudad_ae_str | varchar(60) | null |
| | asa_c_pais | char(3) | not null |

| asa_c_pais | | | |
|---|---|---|---|
| pk | asa_c_pais | char(3) | not null |
| | descripcion_pa_str | varchar(30) | null |

| fac_m_factura | | | |
|---|---|---|---|
| pk | serie_fac_str | char(1) | not null |
| pk | asa_c_aeropuerto | char(3) | not null |
| pk | fac_m_factura | numeric(10) | not null |
| | fecha_fac_dt | datetime | null |
| | asa_c_cliente | integer | null |
| | total_fac_n | numeric(12,2) | null |
| | totaliva_fac_n | numeric(12,2) | null |

Y necesitamos crear las replication definitions correspondientes, al realizar la sustitución respectiva obtendríamos lo siguiente:

```
CREATE REPLICATION DEFINITION %ORIGEN%asa_c_aeropuerto
WITH PRIMARY AT %PDS%.%PDB%
WITH ALL TABLES NAMED asa_c_aeropuerto
(
  asa_c_aeropuerto char(3),
  ciudad_ae_str    varchar(60),
```

```
  asa_c_pais        char(3)
)
WITH PRIMARY KEY (asa_c_aeropuerto)
SEARCHABLE COLUMNS (asa_c_aeropuerto)
go

CREATE REPLICATION DEFINITION %ORIGEN%asa_c_pais
WITH PRIMARY AT %PDS%.%PDB%
WITH ALL TABLES NAMED asa_c_pais
(
  asa_c_pais          char(3),
  descripcion_pa_str varchar(60),
)
WITH PRIMARY KEY (asa_c_pais)
SEARCHABLE COLUMNS (asa_c_pais)
go

CREATE REPLICATION DEFINITION %ORIGEN%fac_m_factura
WITH PRIMARY AT %PDS%.%PDB%
WITH ALL TABLES NAMED fac_m_factura
(
  serie_fac_str     char(1),
  asa_c_aeropuerto char(3),
  fac_m_factura     numeric,
  fecha_fac_dt      datetime,
  asa_c_cliente     integer,
  total_fac_n       numeric,
  totaliva_fac_n    numeric
)
WITH PRIMARY KEY (serie_fac_str, asa_c_aeropuerto, fac_m_factura)
SEARCHABLE COLUMNS (serie_fac_str, asa_c_aeropuerto, fac_m_factura)
go
```

Se puede observar que el identificador **%column_list%** se ha reemplazado por la lista de las columnas de cada una de las tablas. Esto se debe a que en la propia base de datos se almacena esta estructura y esto se aprovecha para no tener que capturar nuevamente todas las columnas. De igual forma el identificador **%pk_column_list%** se ha reemplazado por aquellas columnas que forman la llave primaria de la tabla.

Así podemos observar que solo falta reemplazar las variables %ORIGEN%, %PDS% y %PDB% mismas que respectivamente pueden ser sustituidas de la siguiente forma:

| | |
|---|---|
| %ORIGEN% | Identificador del sitio a dónde pertenece la repication definition, por ejemplo: MEX, TIJ, MTY, GDL, etc. |
| %PDS% | Identificador del Servidor de Datos Primario (Primary Data Server). |
| %PDB% | Identificador de la Base de Datos Primaria (Primary Data Base). |

Al reemplazar las variables anteriores, se completa el script para la creación de las replication definition, y esto se logra a partir de una plantilla, misma que se utiliza de forma general para crear la replication definition de cualquier tabla.

En caso de ser necesario, este código puede ser modificado para ajustarlo a las necesidades reales, sin embargo, la parte más laboriosa a sido automatizada.

Ahora bien, si fuera necesario contar con los scripts para crear las subscripciones, podríamos utilizar un plantilla como la siguiente:

```
CREATE SUBSCRIPTION %ORIGEN%%table_name%%DESTINO%
FOR %ORIGEN%%table_name%
WITH REPLICATE AT %RDS%.%RDB%
WITHOUT MATERIALIZATION
go
```

La cual al realizar los reemplazos correspondientes, quedaría de la siguiente forma:

```
CREATE SUBSCRIPTION %ORIGEN%asa_c_aeropuerto%DESTINO%
FOR %ORIGEN%asa_c_aeropuerto
WITH REPLICATE AT %RDS%.%RDB%
WITHOUT MATERIALIZATION
go


CREATE SUBSCRIPTION %ORIGEN%asa_c_pais%DESTINO%
FOR %ORIGEN%asa_c_pais
WITH REPLICATE AT %RDS%.%RDB%
WITHOUT MATERIALIZATION
go


CREATE SUBSCRIPTION %ORIGEN%fac_m_factura%DESTINO%
FOR %ORIGEN%fac_m_factura
WITH REPLICATE AT %RDS%.%RDB%
WITHOUT MATERIALIZATION
go
```

Obviamente, a los scripts anteriores es necesario reemplazar las variables %ORIGEN%, %DESTINO%, %RDS% y %RDB% por valores que sean significativos al esquema que se vaya a implementar.

El programa GSRS es capaz de administrar variables definidas por el usuario, con la finalidad de que se ajuste a las necesidades especificas de los scripts a crear. Además de incluir variables predefinidas tales como %column_list% y %pk_column_list% cuyo valor se establece automáticamente dependiendo de la tabla que se este procesando.

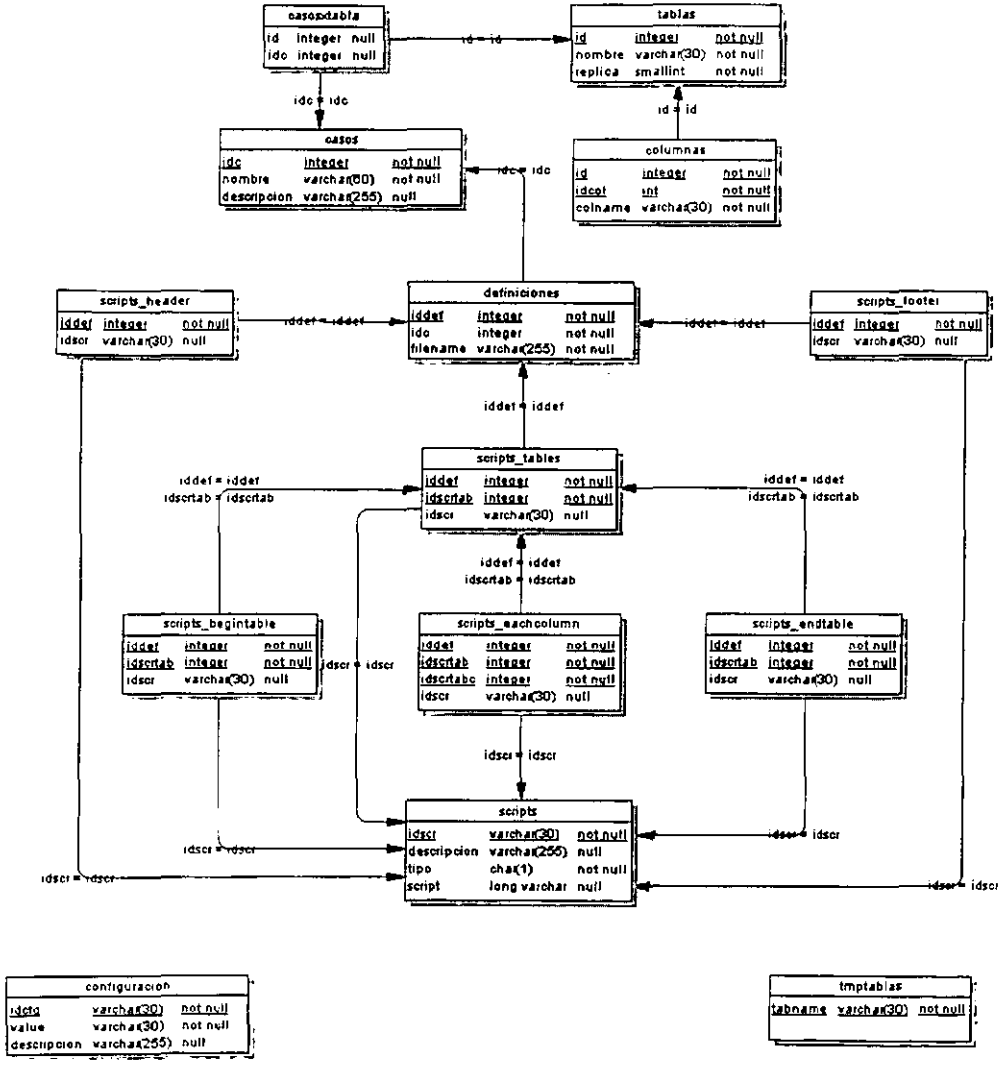
## Desarrollo del GSRS.

Debido a que el tema de esta tesis consiste en la implementación de esquemas de replicación y no el desarrollo mismo del programa GSRS, solo se mencionarán los aspectos generales de este para que pueda ser utilizado en la generación del los scripts cuando se necesite implantar un esquema de replicación basado en Replication Server.

Este programa fue diseñado por un servidor, como resultado de la necesidad de crear y dar mantenimiento a los scripts de replicación durante una fase de desarrollo del esquema de replicación de Aeropuertos y Servicios Auxiliares. Esta creación y mantenimiento de scripts como se ha intentado demostrar a lo largo de la presente, puede resultar laboriosa y compleja sobre todo en bases de datos de gran tamaño, ya que el encargado de realizar esta tarea practicamente tiene que reescribir la estructura de la base de datos, misma que ya existe y que alguna herramienta debió crear. Por lo que se pretende aprovechar la existencia misma de las tablas para crear los scripts de las replication definitions, dándole un valor adicional al permitir generar los scripts complemento del esquema global. Es decir, cuando se crean replication definitions, será necesario crear publicaciones, articulos y subscripciones mismas que también requieren un script.

Para el crear esta primera versión del GSRS, se utilizó como herramienta de desarrollo a Power Builder 7.0, aunque es necesario contar con una herramienta que permita administrar y almacenar las estructuras y datos con los que GSRS opera. A estos últimos los identificaremos como **repositorio**, utilizando a Sybase Adaptive Server Anywhere como la base de datos en dónde la estructuras y datos del repositorio serían almacenados, esto para evitar crear estructuras de datos propias del GSRS con lo cual el desarrollo del mismo implicaría un mayor esfuerzo.

La estructura de este repositorio es la siguiente:




El funcionamiento del GSRS se basa en seis opciones principales:

- Base de Datos.
- Etiquetas.
- Scripts.

- Casos.
- Scripts por Caso.
- Tablas por Caso.

En dónde cada opción se describe a continuación.

Base de Datos

Esta opción tiene como finalidad extraer de la base de datos todas aquellas tablas así como las columnas y llaves primarias respectivas, con el propósito de almacenar esta estructura dentro del repositorio del GSRS, y poder crear los scripts cuantas veces sea necesario aunque no se tenga acceso directo a la base de datos.

Para que el GSRS, pueda extraer la estructura de la base de datos necesaria para crear los scripts de replicación, se diseñaron tres procedimientos almacenados cuya función es listar las tablas (rsg_sp_tables), columnas y tipos de datos (rsg_sp_columns) o los índices únicos (rsp_sp_indexes).

El código de estos procedimientos se presenta a continuación:

Para lograr esto y pensando en poder accesar a diferentes DBMS's y no solo Sybase, entonces se han creado dos procedimientos almacenados llamados rsg_sp_columns y rsg_sp_indexes mismos que tienen el objetivo.

El código de estos procedimientos almacenados se presenta a continuación:

```
CREATE PROCEDURE rsg_sp_tables
BEGIN
  SELECT name FROM sysobjects WHERE type="U" ORDER BY name
END
```

```
CREATE PROCEDURE rsg_sp_columns @table_name varchar(30) AS
BEGIN
  DECLARE @table_id integer

  SELECT @table_id = object_id(@table_name)

  SELECT column_name = c.name,
         data_type = d.data_type+convert(smallint,  isnull(d.aux,
ascii(substring("666AAA@@@CB??GG", 2*(d.ss_dtype%35+1)+2-
8/c.length,1))-60)),
```

```
        type_name =
rtrim(substring(d.type_name,1+isnull(d.aux,ascii(substring("III<<<MM
MI<<A<A",2*(d.ss_dtype%35+1)+2-8/c.length,1))-60), 13)),
        prec = isnull(convert(int, c.prec),isnull(convert(int,
d.data_precision),convert(int,c.length)))+isnull(d.aux,
convert(int,ascii(substring("???AAAFFFCKFOLS",2*(d.ss_dtype%35+1)+2-
8/c.length,1))-60)),
        length = isnull(convert(int, c.length),convert(int,
d.length)) + convert(int,
isnull(d.aux,ascii(substring("AAA<BB<DDDHJSPP",2*(d.ss_dtype%35+1)+2
-8/c.length,1))-64)),
        scale = isnull(convert(smallint, c.scale),convert(smallint,
d.numeric_scale))+convert(smallint,isnull(d.aux,ascii(substring("<<<
<<<<<<<<<<<?",2*(d.ss_dtype%35+1)+2-8/c.length,1))-60)),
        radix = d.numeric_radix,
        nullable =  convert(smallint, convert(bit, c.status&8)),
        ss_data_type = c.type,
        colid = c.colid
    FROM syscolumns c,
        sysobjects o,
        sybsystemprocs.dbo.spt_datatype_info d,
        systypes t
   WHERE o.id = @table_id
     AND c.id = o.id
     AND c.usertype = t.usertype
     AND t.type = d.ss_dtype
     AND d.ss_dtype IN (111, 109, 38, 110)
     AND c.usertype < 100

   UNION

  SELECT column_name = c.name,
        data_type =
d.data_type+convert(smallint,isnull(d.aux,ascii(substring("666AAA@@@
CB??GG",2*(d.ss_dtype%35+1)+2-8/c.length,1))-60)),
        type_name =
rtrim(substring(d.type_name,1+isnull(d.aux,ascii(substring("III<<<MM
MI<<A<A",2*(d.ss_dtype%35+1)+2-8/c.length,1))-60), 13)),
        prec = isnull(convert(int, c.prec),isnull(convert(int,
d.data_precision),convert(int,c.length)))+isnull(d.aux,
convert(int,ascii(substring("???AAAFFFCKFOLS",2*(d.ss_dtype%35+1)+2-
8/c.length,1))-60)),
        length = isnull(convert(int, c.length),convert(int,
d.length)) + convert(int,
isnull(d.aux,ascii(substring("AAA<BB<DDDHJSPP",2*(d.ss_dtype%35+1)+2
-8/c.length,1))-64)),
        scale = isnull(convert(smallint, c.scale),
convert(smallint, d.numeric_scale)) +convert(smallint,
isnull(d.aux,ascii(substring("<<<<<<<<<<<<<<<?",2*(d.ss_dtype%35+1)+2
-8/c.length,1))-60)),
        radix = d.numeric_radix,
        nullable =convert(smallint, convert(bit, c.status&8)),
```

```
        ss_data_type = c.type,
        colid = c.colid
    FROM syscolumns c,
        sysobjects o,
        sybsystemprocs.dbo.spt_datatype_info d,
        systypes t
   WHERE o.id = @table_id
     AND c.id = o.id
     AND c.usertype = t.usertype
     AND t.type = d.ss_dtype
     AND (d.ss_dtype NOT IN (111, 109, 38, 110)
                    OR c.usertype >= 100)
ORDER BY colid

END



CREATE PROCEDURE rsg_sp_indexes @objname varchar(30) AS
BEGIN

  DECLARE @indid integer

  select indid,
        name,
        ( select count(*)
            from master.dbo.spt_values v
           where i.status & v.number = v.number
             and v.type = "I"
             and v.number = 2 ) uniqueindex,
        ( select count(*)
            from master.dbo.spt_values v
           where i.status & v.number = v.number
             and v.type = "I"
             and v.number = 1 ) ignoredupkey,
        ( select count(*)
            from master.dbo.spt_values v
           where i.status & v.number = v.number
             and v.type = "I"
             and v.number = 4 ) ignoreduprow,
        ( select count(*)
            from master.dbo.spt_values v
           where i.status & v.number = v.number
             and v.type = "I"
             and v.number = 64 ) allowduporow
    into #indexes
    from sysindexes i
   where id = object_id(@objname)

 select @indid = min(indid)
```

```
    from #indexes
   where uniqueindex >= 1

  CREATE TABLE #keys
  (
    keyname varchar(30)
  )

  if @indid is not null
  begin

    declare @i int
    declare @thiskey varchar(30)
    select @i = 1

    set nocount on

    while @i <= 16
    begin
      select @thiskey = index_col(@objname, @indid, @i)

      if @thiskey is NULL
      begin
        goto keysdone
      end
      insert into #keys values ( @thiskey )
      select @i = @i+1
    end
  end

  keysdone:

  select * from #keys

END
```

*Aunque los procedimientos almacenados anteriores funcionan para el DBMS de Sybase, estos nos dan la oportunidad de que el programa GSRS, pueda ser utilizado para otros manejadores como pudieran ser Oracle, Informix, DB2 o Microsoft SQL Server. Para lograr lo anterior, es necesario que estos procedimientos sean trasladados hacia los lenguajes de estos manejadores, por supuesto, respetando la salida de los mismos para que pueda ser recibida sin problema alguno por el GSRS.*

## Etiquetas

En la opción etiquetas, el usuario podrá definir identificadores a los cuales pueda asignarles valor alguno dependiendo de los scripts que requiera, es decir, el usuario podrá definir un identificador llamado %ORIGEN% que sirva para especificar el lugar a dónde los scripts que se generen deban relacionarse, así con solo cambiar este valor podrán generarse scripts prar diferentes sitios.

## Scripts

Por medio de esta opción, se definen las plantillas que van a ser utilizadas, tal y como se muestra en la plantilla al inicio de la sección Bases del Generador de Scripts de Replication Server (GSRS). dentro de este mismo capítulo. A cada plantilla se le asigna un nombre para que pueda ser utilizado en Scripts por Caso. Un ejemplo de plantillas pueden ser las siguientes:

```
ID        SCRIPT
REPDEF1   CREATE REPLICATION DEFINITION %ORIGEN%%table_name%
          WITH PRIMARY AT %PDS%.%PDB%
          WITH ALL TABLES NAMED %table_name%
          (
            %column_list%
          )
          WITH PRIMARY KEY (%pk_column_list%)
          SEARCHABLE COLUMNS (%pk_column_list%)

REPDEF1   CREATE REPLICATION DEFINITION %DESTINO%%table_name%
          WITH PRIMARY AT %RDS%.%RDB%
          WITH ALL TABLES NAMED %table_name%
          (
            %column_list%
          )
          WITH PRIMARY KEY (%pk_column_list%)
          SEARCHABLE COLUMNS (%pk_column_list%)

GO        go
```

## Casos

Sencillamente dentro de esta opción se enlistan los diferentes casos en los que se pueden clasificar las tablas de la base de datos.

Para identificar los casos, podemos basarnos en la Matriz de Uso Global de Datos que se presentó en la página 64, en ella se pueden identificar tablas que tienen un comportamiento similar, tal es el caso de las tablas asa_c_avion, asa_c_cliente, asa_c_matricula y asa_c_pais; mismas que son primarias en las oficinas generales y replicadas tanto en aeropuertos como en las estaciones de combustibles.

## Scripts por Caso

Por medio de esta opción es posible relacionar los scripts que se relacionan con un caso específico en archivos de salida individuales, con el propósito de crear los comandos listos en estos archivos para que puedan ser ejecutados directamente sobre la base de datos o sobre Replication Server. Por ejemplo, para el caso que agrupe las tablas asa_c_avion, asa_c_cliente, asa_c_matricula y asa_c_pais, serán requeridos scripts para generar:

- Replication Definitions,
- Publicaciones,
- Articulos y
- Subscripciones.

De tal forma que podemos decir que para este caso específico o conjunto de tablas se requerirán los siguientes scripts:

- crerep.sql
- crepub.sql
- creart.sql
- cresub.sql

Siendo cada uno de estos scripts un archivo que contedrá los comandos un conjunto de Scripts (definidos estos en la opción Scripts descrita con anterioridad). Y para cada uno de estos archivos se definen los scripts que conformarán al archivo, por ejemplo, podemos decir que el script crerep.sql contendrá los siguientes scripts:

```
REDEF1
GO
```

Ambos definidos durante la descripción de la opción scripts, y con lo cual al generar los scripts se utilizarán las plantillas.

### Tablas por Caso

Como última opción, será necesario especificar cuales son las tablas que se relacionan con un caso específico, siendo este el objetivo de esta opción.

### Crear Scripts

Esta opción como su nombre lo indica ejecuta la creación de los scripts, utilizando todos los elementos que se han descrito del GSRS.

# CAPITULO V. CONCLUSIONES

Al finalizar este trabajo podemos darnos cuenta de la importancia que la replicación de datos asume dentro de aquellas organizaciones con un tamaño tal, que estas requieren mantener un control centralizado pero con las bondades que la descentralización pueda brindar. De esta forma, es posible que estas organizaciones puedan establecer políticas a sus unidades de negocio remotas, pero sin sacrificar la independencia de estas. Más aún, si consideramos que en la actualidad muchas compañías operan con más de una sola oficina, entonces todas y cada una de ellas de alguna forma tienen la necesidad de recibir, procesar y analizar información proveniente de cada uno de sus centros de operación; así como, el envío de políticas y formas de trabajo, con la finalidad de que estas últimas sean implementadas en cada uno de esos centros. Así, cualquier organización que posea características como las que se han mencionado, y la cual utilice sistemas y bases de datos para su operación diaria, podrá hacer uso de un esquema de replicación que le permita de manera automática, contar con una transmisión bidireccional de información con sus centros de trabajo. Lo anterior, sin olvidar claro, la independencia de cada uno de esos centros para realizar sus actividades diarias.

Ahora bien, la replicación de datos nos dará la oportunidad no solo de automatizar el intercambio de información entre diversos puntos, sino también implícitamente ayudará a contar con sitios alternativos de trabajo en caso de contingencia alguna, con lo cual, la operación diaria de algún centro de trabajo no se verá afectada, de esta forma, podemos decir que la replicación de datos también nos permite crear esquemas redundantes de datos, mismos que dan la posibilidad de continuar con la operación diaria de las organizaciones, disminuyendo el riesgo de perder información ante cualquier eventualidad.

Es muy importante resaltar que a lo largo del presente trabajo, en ningún momento fue mencionada la necesidad de modificar o crear aplicación alguna como parte misma de un esquema de replicación, es decir, el diseñar o implementar un esquema de replicación no implica necesariamente el modificar las aplicaciones existentes o el desarrollo de nuevas aplicaciones dentro de la organización. Por el contrario, siempre se hizo referencia a las aplicaciones ya existentes, sin embargo, el comprender el funcionamiento de dichas aplicaciones es necesario e indispensable, para poder determinar el comportamiento de la replicación de datos. Con lo anterior, podemos afirmar que el diseño e implantación de un esquema de replicación, no requiere de la modificación de las aplicaciones existentes o la creación de nuevas aplicaciones, pero sí requiere del conocimiento de estas para poder realizar un buen diseño y por lo tanto lograr una implantación exitosa.

Tomando en cuenta lo anterior, podemos afirmar que un esquema de replicación debe crearse de acuerdo a las necesidades de la empresa y/o de las aplicaciones que esta utiliza. Lo anterior, en virtud de que un usuario final no trabaja directamente con la base de datos (o con el DBMS para ser más preciso), sino más bien este usuario final utiliza aplicaciones diseñadas para cumplir con sus actividades diarias dentro de la empresa (generalmente), y las cuales son las que hacen uso directo del DBMS.

Conforme al párrafo anterior, y considerando que un esquema de replicación debe ajustarse a las **necesidades** de la empresa, es posible asegurar que un esquema de replicación no solo estará compuesto por los modelos ya definidos en el propio Replication Server, sino que también podrán incluirse modelos alternativos que se ajusten a los requerimientos específicos de la empresa. Por lo tanto, a pesar de que los modelos de replicación definidos en la documentación propia del Replication Server son finitos, es posible crear los modelos adecuados que se ajusten a la funcionalidad que se persigue dentro del esquema de replicación.

Sin embargo, el diseño de un esquema de replicación, requiere de una metodología misma que permita seguir una línea de acción bien definida, con el propósito de no perder el contorno general del esquema de distribución, ya que de no ser así, el costo que pudiera resultar de una mala planeación (como en cualquier otra actividad) puede ser muy alto, por lo que, es mucho más conveniente invertir tiempo para planear lo que se va a hacer y no invertir tiempo en como se va a corregir lo que ya se ha hecho. Por lo tanto, podemos determinar que el uso de una metodología para diseñar un esquema de replicación permitirá que el diseño de este esquema se ajuste a las necesidades y condiciones reales existentes.

La metodología que se ha presentado en este trabajo establece esa línea a seguir, la cual nos permitirá identificar no solo a los elementos participantes dentro del esquema de replicación, es decir: sitios, bases de datos, tablas y procesos entre otros; sino también los recursos existentes a utilizar, mismos que pudieran llegar a ser una limitación y los cuales deben ser considerados durante el buen diseño de un esquema de replicación.

Como parte de la metodología ya mencionada, se presenta la Matriz de uso global de datos, misma que es el resultado de la experiencia de un servidor en el diseño de esquemas de replicación, y la cual permitirá identificar y establecer flujos de información así como puntos primarios y replicados, mismos que serán de gran ayuda en la definición gráfica del esquema de replicación.

Por todo lo anterior, podemos concluir que el buen **DISEÑO** de un esquema de replicación el cual se base en una metodología, nos permitirá definir un esquema de replicación conforme los requerimientos y recursos existentes, adecuándose a las necesidades y condiciones reales de la empresa.

Una vez que se tiene el diseño de un esquema de replicación, es necesario llevar a cabo la **IMPLEMENTACION** del mismo, por lo que es necesario desarrollar los scripts que permitan configurar el esquema. Para ello y como se mostró en el cuarto capítulo, se propone la utilización del GSRS (Generador de Scripts de Replication Server), cuyo propósito es extraer de la base de datos la información necesaria para la creación de estos scripts. De tal forma y concluyendo, se puede asegurar que el GSRS dará la facilidad para generar los scripts necesarios para la creación y mantenimiento de un esquema de replicación.

El futuro de la replicación de datos se está redefiniendo de tal forma que en los últimos años se ha planteado la necesidad de replicar información entre DBMS heterogéneos, como pueden ser Sybase, Oracle, Informix, DB2 e inclusive MS SQL Server; cada uno de ellos siendo un sitio primario o replicado dentro de un esquema de replicación. Lo anterior se presenta también como resultado de la globalización, ya que la fusión de compañías implica que cada una de las partes que

entran en la fusión cuenta con plataformas y DBMS diferentes, lo cual implica contar esquemas de movimiento de datos heterogéneos. Sin embargo, sin importar plataforma o DBMS, la metodología que se ha planteado puede ser utilizada sin contratiempo alguno, y sin importar el DBMS que sea utilizado como puntos primarios o replicados, lo anterior en virtud de que en un nivel conceptual siempre será considerado como un DBMS.

Sin embargo, como se mencionó durante el desarrollo del GSRS, será necesario crear los procedimientos necesarios que permitan a este último operar con DBMS diferentes a Sybase, siendo este último la plataforma sobre la cual se desarrollo este programa.

Así, cuando alguien tome la decisión de automatizar la transmisión de información o mejor dicho crear un esquema de replicación, el presente trabajo podrá ser de gran utilidad para considerar todos los elementos que intervienen dentro de un esquema de replicación, y si fuera el caso de implementarlo con Replication Server, entonces podrán hacer uso no solo de la metodología que se incluye, sino de las herramientas que se han creado como resultado de la experiencia personal durante el **DISEÑO E IMPLEMENTACION DE ESQUEMAS DE REPLICACION CON REPLICATION SERVER**, durante poco más de cuatro años.

# APENDICE A. CODIGO FUENTE GSRS.

## Objetos del GSRS

### Listado de Objetos

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01 Time: 11:55:39

Application entries.

| Name | Date | Time | Size |
|---|---|---|---|
| app_script | 24/08/00 | 17:05:00 | (4392) |

DataWindow entries.

| Name | Date | Time | Size |
|---|---|---|---|
| ddw_config | 30/06/00 | 11:01:15 | (3356) |
| ddw_scripts | 30/06/00 | 11:01:17 | (3001) |
| ddw_tablas | 30/06/00 | 11:01:16 | (3377) |
| dw_casos | 30/06/00 | 11:01:15 | (4210) |
| dw_casosxtabla | 30/06/00 | 11:01:17 | (4895) |
| dw_config | 30/06/00 | 11:01:16 | (4163) |
| dw_definiciones | 30/06/00 | 11:01:15 | (3938) |
| dw_defscripts | 30/06/00 | 11:01:17 | (4528) |
| dw_factura | 30/06/00 | 11:01:16 | (18747) |
| dw_process | 30/06/00 | 11:01:15 | (8878) |
| dw_process_secc | 30/06/00 | 11:01:17 | (9191) |
| dw_scriptsl | 30/06/00 | 11:01:16 | (4377) |
| dw_sp_columns | 30/06/00 | 11:57:11 | (8501) |
| dw_sp_columnsant | 30/06/00 | 11:56:58 | (14518) |
| dw_sp_pkeys | 30/06/00 | 11:56:22 | (3014) |
| dw_sp_pkeysant | 30/06/00 | 11:56:07 | (5492) |
| dw_statusbar | 30/06/00 | 11:01:17 | (3570) |
| dw_sysobjects | 30/06/00 | 11:01:16 | (2832) |
| dw_tablas | 30/06/00 | 11:01:15 | (3679) |

Function entries.

| Name | Date | Time | Size |
|---|---|---|---|
| gf_generate | 30/06/00 | 11:40:36 | (6111) |
| gf_padl | 30/06/00 | 11:01:16 | (1441) |
| gf_padr | 30/06/00 | 11:01:15 | (1441) |
| gf_replaceall | 30/06/00 | 11:01:14 | (1617) |
| gf_strcount | 30/06/00 | 11:01:16 | (1525) |
| gf_strextract | 30/06/00 | 11:01:15 | (1867) |
| gf_strtran | 30/06/00 | 11:01:14 | (1499) |

Menu entries.

| Name | Date | Time | Size |
|---|---|---|---|
| m_edit | 30/06/00 | 11:01:16 | (13206) |
| m_item | 30/06/00 | 11:01:15 | (2370) |
| m_repgen | 25/02/00 | 23:37:25 | (14586) |
| m_tags | 30/06/00 | 11:01:16 | (3101) |

UserObject entries.

| Name | Date | Time | Size |
|---|---|---|---|
| u_cb | 30/06/00 | 11:01:14 | (1652) |
| uo_statusbar | 30/06/00 | 11:53:10 | (10933) |
| uo_vars | 30/06/00 | 11:01:15 | (17968) |

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01 Time: 11:55:39

Window entries.

| Name | Date | Time | Size |
|---|---|---|---|
| w_casos | 30/06/00 | 11:33:21 | (35468) |
| w_config | 30/06/00 | 11:01:16 | (10627) |
| w_main | 30/06/00 | 11:01:15 | (26169) |
| w_scripts | 30/06/00 | 11:01:14 | (18144) |
| w_setcasos | 30/06/00 | 11:01:17 | (24779) |
| w_statusbar | 30/06/00 | 11:01:16 | (5935) |
| w_tablas | 30/06/00 | 11:01:14 | (23683) |

Project entries.

| Name | Date | Time | Size |
|---|---|---|---|
| repgen | 3/08/99 | 14:19:11 | (596) |

app_script

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:31:26


Application: app_script


Global Variables
transaction     SQLServer
transaction     SQLAny50



End of Global Variables



Script for: nt open  event
SQLServer=CREATE transaction
SQLAny50 =CREATE transaction

SQLServer.DBMS="SYC"
SQLServer.Database="compras"
SQLServer.ServerName="SYB_ASE12"
SQLServer.LogId="rep_user"
SQLServer.LogPass="rep_user_ps"
SQLServer.Autocommit=True


CONNECT USING SQLServer;
If SQLServer.SQLCode < 0 Then
   MessageBox( "Conección","Falló la conexión a "+ SQLServer.ServerName,
StopSign!)
   Return
End if


SQLAny50.DBMS="ODBC"
SQLAny50.Database="Replicacion"
SQLAny50.ServerName="Replicacion"
SQLAny50.DBParm=  "ConnectOption ='SQL_DRIVER_CONNECT, SQL_DRIVER_NOPROMPT;'" +
&
                  "ConnectString='DSN=replicacion;Uid=dbo;Pwd=dbo';"
SQLAny50.Autocommit = True


//SQLAny50.DBParm="ConnectOption ='SQL_DRIVER_CONNECT,SQL_DRIVER_NOPROMPT;'" + &
//                "ConnectString='DSN=replicacion;Uid=Dba;Pwd=sql';"
//
////"ConnectOption ='SQL_DRIVER_CONNECT,SQL_DRIVER_NOPROMPT;'" + &
//            "ConnectString='DSN=" + SQLCA.Database + &
//              ";UID= dba" &
//              ";PWD= sql';"
//
```

```
CONNECT USING SQLAny50;
If SQLAny50.SQLCode < 0 Then
   MessageBox( "Conección","Falló la conexión a SQLAny50",StopSign!)
   Return
End if

Open( w_main )
end event

End of Script


Script for: nt close event
DISCONNECT USING SQLServer;
-

DESTROY SQLServer
DESTROY SQLAny50


end event

End of Script



Application References

Window                w_main
                      D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL

End of Application References
```

## gf_generate

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:35:09

Function    :  gf_generate
Return Type :  integer
Arguments   :
                    string              as_sql
End of Arguments

Long         ll_Caso, ll_RowsHeader, ll_Header
String       ls_File = "", ls_FileName
String       ls_TableName = ""
Long         ll_Row, ll_iddef
Boolean      lb_FileOpened = False
Integer      li_Handle
String       ls_Path = "D:\Mis documentos\Personal\PowerBuilder\RSSG\", ls_Caso
String       ls_Script = ""
datastore    lds_proceso, lds_seccion
uo_Vars      luo_Vars
pointer      oldpointer

oldpointer = SetPointer(HourGlass!)

lds_proceso = CREATE datastore
lds_seccion = CREATE datastore

lds_proceso.DataObject = "dw_config"
lds_proceso.Settransobject( SQLAny50 )
lds_proceso.Retrieve()

lds_seccion.DataObject = "dw_process_secc"
lds_seccion.Settransobject( SQLAny50 )

luo_Vars.Resetvars()
For ll_Row = 1 To lds_Proceso.RowCount()
  luo_Vars.Setvar( lds_proceso.getitemstring( ll_Row, "idcfg" ), &
                   lds_proceso.getitemstring( ll_Row, "value" ) )
Next

lds_proceso.DataObject = "dw_process"
lds_proceso.Settransobject( SQLAny50 )
If as_SQL <> "" Then
  li_Handle = lds_proceso.SetSQLSelect( as_sql )
End if
lds_proceso.Retrieve()


w_main.uo_1.of_Message( "Generando Scripts..." )
w_main.uo_1.of_SetAvance( 0 )

For ll_Row = 1 To lds_Proceso.RowCount()

  // Determina si ha cambiado el archivo que se está procesando (Solo si ll_Row
es may
or a 1 )
```

```
  If lds_proceso.getitemNumber( ll_Row, "iddef" ) <> ll_iddef And ll_Row > 1
Then
      // Si ya cambió se debe procesar el Pie del Archivo
    w_main.uo_1.of_Message( ls_Filename+' (pie)' )
    ll_RowsHeader = lds_seccion.Retrieve( ll_iddef,3)
    For ll_Header = 1 to ll_RowsHeader
      ls_Script = lds_seccion.GetItemString( ll_Header, "script" )

      luo_Vars.ReplaceVars( ls_Script )

      FileWrite( li_Handle, ls_Script )
    Next

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:35:09

  End if


  if ll_Caso <> lds_Proceso.GetItemNumber( ll_Row, "caso" ) Then
    ll_Caso = lds_Proceso.GetItemNumber( ll_Row, "caso" )
    ls_Caso = lower(lds_Proceso.GetItemString( ll_Row, "nombre" ))
    luo_Vars.SetVar( "cname", ls_Caso )
    If lb_FileOpened Then
      FileClose( li_Handle )
      lb_FileOpened = False
    End if
  end if

  if ls_File <> lds_Proceso.GetItemString( ll_Row, "archivo" ) Then
    ls_File = lds_Proceso.GetItemString( ll_Row, "archivo" )
    If lb_FileOpened Then
      FileClose( li_Handle )
      lb_FileOpened = False
    End if
  end if


  If Not lb_FileOpened Then
    ls_FileName = ls_Path+ls_File+"."+ls_Caso
    li_Handle = FileOpen( ls_Filename, StreamMode!, Write!, LockWrite!, Replace!
)
    lb_FileOpened = True

    // Determina si ha cambiado el archivo que se está procesando
    If lds_proceso.getitemNumber( ll_Row, "iddef" ) <> ll_iddef Then
      // Si ya cambió se debe procesar el Encabezado del Archivo
      ll_iddef = lds_proceso.getitemNumber( ll_Row, "iddef" )
      w_main.uo_1.of_Message( ls_Filename+' (encabezado)' )
      ll_RowsHeader = lds_seccion.Retrieve( ll_iddef,1 )
      For ll_Header = 1 to ll_RowsHeader
        ls_Script = lds_seccion.GetItemString( ll_Header, "script" )

        luo_Vars.ReplaceVars( ls_Script )

        FileWrite( li_Handle, ls_Script )
      Next
    End if
```

```
   End if

   if ls_TableName <> lds_Proceso.GetItemString( ll_Row, "tabla" ) Then
     ls_Tablename = lds_Proceso.GetItemString( ll_Row, "tabla" )
     luo_Vars.SetVar( "table_name", ls_TableName )
   end if

   ls_Script = lds_Proceso.GetItemString( ll_Row, "script" )

   w_main.uo_1.of_Message( ls_Filename+' ('+ls_tablename+')' )
   luo_Vars.ReplaceVars( ls_Script )

   FileWrite( li_Handle, ls_Script )

   w_main.uo_1.of_SetAvance( Integer( ( ll_Row / lds_Proceso.RowCount() ) * 100 )
)
   yield()

Next

If ll_Row > 1 Then
  w_main.uo_1.of_Message( ls_Filename+' (pie)' )
  ll_RowsHeader = lds_seccion.Retrieve( ll_iddef, 3 )
  For ll_Header = 1 to ll_RowsHeader

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:35:09

    ls_Script = lds_seccion.GetItemString( ll_Header, "script" )

    luo_Vars.ReplaceVars( ls_Script )

    FileWrite( li_Handle, ls_Script )
  Next
End if


FileClose( li_Handle )
lb_FileOpened = False

DESTROY lds_proceso
DESTROY lds_seccion

w_main.uo_1.of_Message( "Listo." )

SetPointer(oldpointer)

Return 1
end function
```

## gf_padl

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:37:10

Function    :  gf_padl
Return Type :  string
Arguments   :
                    string              c
                    integer             n
End of Arguments

/*****************************************************************************
******
*****
* Funcion:              PadL
* Objetivo:Que una cadena sea de una longitud especifica, agregando esp
acios a
*                       la Izquierda o tomando la parte izquierda de la cadena.
* Parámetros:
*    c                  String que se va a modificar.
*    n                  Número de posiciones que se desean para la cadena.
*****************************************************************************
******
****/
If IsNull( c ) Then
  c=""
End if
Choose Case len( c )
   Case Is < n
     c = Space( n - len( c ) ) + c
   Case Is > n
     c = Right( c,n )
End Choose

Return c
end function
```

## gf_padr

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:38:22

Function    :  gf_padr
Return Type :  string
Arguments   :
                    string              c
                    integer             n
End of Arguments

/*****************************************************************************
******
*****
* Funcion:              PadL
* Objetivo:Que una cadena sea de una longitud especifica, agregando esp
acios a
*                       la derecha o tomando la parte derecha de la cadena.
* Parámetros:
*    c                  String que se va a modificar.
*    n                  Número de posiciones que se desean para la cadena.
*****************************************************************************
******
****/
If IsNull( c ) Then
  c=""
End if

Choose Case len( c )
   Case Is < n
      c = c + Space( n - len( c ) )
   Case Is > n
     c = Left( c,n )
End Choose

Return c
end function
```

## gf_replaceall

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:39:21

Function    :  gf_replaceall
Return Type :  integer
Arguments   :
     ref            string              cadena
                    string              cadenaant
                    string              cadenaact
End of Arguments

string ls_stringtmp
string ls_string
Long   ll_i

ls_string = cadena

Do

  ll_i = Pos( ls_string, cadenaant )
  If ll_i > 0 Then
    ls_stringtmp = Replace( ls_string, ll_i, Len(cadenaant), cadenaact )
    If ls_stringtmp <> "" Then
      ls_string = ls_stringtmp
    End if
  End if
Loop Until ll_i = 0

cadena = ls_string

Return 1
end function
```

## gf_strcount

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:40:43

Function    :  gf_strcount
Return Type :  integer
Arguments   :
                    string              sstr
                    string              ssubstr
End of Arguments

/*****************************************************************************
****
*
*     Función: gf_StrCount()
*
*       Autor: Antonio Morales Téllez
*
*       Tarea: Contar el numero de veces que se encuentra <sSubStr> en <sStr>
*
*  Argumentos: <sStr>      String Origen del que se desea saber el numero de
veses
*                           que se encuentra el SubString <sSubStr>
*              <sSubStr>   String Origen
*
*     Retorna: <Integer>   El numero de veces que se encontro <sSubStr> dentro
de
*                           <sStr>
*
*    Ejemplos: gf_StrExtract( "Uno;Dos;Tres;Cuatro", ";" )
*                    Retorna 3
*
*              gf_StrExtract( "Escuela|Camion|Tren|Hospital", "TREN" )
*                    Retorna 0
*
*****************************************************************************
***/


Integer iRet
Integer iPos
Integer iSubStrLen

iSubStrLen = Len( sSubStr )

Do While Len( sStr ) > 0

   iPos = Pos( sStr, sSubStr )

   If iPos > 0 Then
      iRet ++
      sStr = Mid( sStr, iPos + iSubStrLen )
   Else
      Exit
   End If
```

```
Loop

Return iRet

end function
```

## gf_strextract

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:41:42

Function    :  gf_strextract
Return Type :  string
Arguments   :
                    string              sstr
                    string              ssep
                    integer             ipos
End of Arguments

/*****************************************************************************
****
*
*     Función: gf_StrExtract()
*
*       Autor: Antonio Morales Téllez
*
*       Tarea: Extrae de un String separado por un string definido
*
*  Argumentos: <sStr>      String Origen del que se desea substrer
*              <sSep>      String separador
*              <iPos>      Posición que se desea extraer
*     Retorna: <String>    El String Segun los parametros
*
*    Ejemplos: gf_StrExtract( "Uno;Dos;Tres;Cuatro", ";", 2 )
*                    Retorna "Dos"
*
*              gf_StrExtract( "Escuela|Camion|Tren|Hospital", "|", 4 )
*                    Retorna "Hospital"
*
*****************************************************************************
***/

Integer iPosSep
String sSubStr
Integer iP
Integer iSepLen

iSepLen = Len( sSep )

If Pos( sStr, sSep ) = 0 Then
   If iPos = 1 Then
      Return sStr
   Else
      Return ""
   End if
End If

sStr += sSep

For iP = 1 to iPos

   iPosSep = Pos( sStr, sSep )
```

```
    If iPosSep > 0 Then
       sSubStr = Left( sStr, iPosSep - 1 )
       sStr = Mid(  sStr, iPosSep + iSepLen )
    Else
       If iPos = iP Then
          sSubStr = sStr
       Else
          sSubStr = ""
       End if
       Exit
    End If

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01     Time: 12:41:42

Next

Return sSubStr


end function
```

## gf_strtran

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:42:33

Function    :  gf_strtran
Return Type :  string
Arguments   :
                    string              asString
                    string              old_str
                    string              new_str
End of Arguments

/*****************************************************************************
****
*
*     Función: gf_StrTran()
*
*       Autor: Antonio Morales Téllez
*
*       Tarea: En <asString> remplaza todos los caracteres que concuerden con
*              <old_str> y se reemplazaran con <new_str>
*
*  Argumentos: <sString>   String Origen del que se desea reemplazar
*              <old_str>   String que se desea reemplazar
*              <new_str>   String que se desea poner en ves de <old_str>
*     Retorna: <String>    El String reemplazado
*
*    Ejemplos: gf_StrTran( "ABCDEFGHIJKL", "CD", "cd" )
*                    Retorna "ABcdEFGHIJKL"
*
*              gf_StrTran( "prea_{mes} - preb_{mes}", "{mes}", "enero" )
*                    Retorna "prea_enero - preb_enero"
*
*****************************************************************************
***/

long start_pos=1

start_pos = Pos(asString, old_str, start_pos)

DO WHILE start_pos > 0

   asString = Replace(asString, start_pos, Len(old_str), new_str)

   start_pos = Pos(asString, old_str, start_pos+Len(new_str))
LOOP

Return asString
end function
```

m_edit

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:43:25

Instance Variables
  String is_Type
  String is_Value

End of Instance Variables


public subroutine of_settype (string as_type, string as_value)
is_type  = lower(as_type)
is_value = as_value
end subroutine



Menu = m_edit
MenuItem = m_crearscript    "Crear Scripts"
Visible = true      Enabled = true

Script for: nt clicked  event
MessageBox( Upper(is_type), 'Generar Scripts '+is_value  )

String   ls_SQL

Choose Case is_type
  Case 'script'
    ls_SQL = 'SELECT "casos"."idc", ~r~n' + &
             '       "casos"."nombre", ~r~n' + &
             '       "definiciones"."filename", ~r~n' + &
             '       "defscripts"."idscr", ~r~n' + &
             '       "scripts"."script", ~r~n' + &
             '       "definiciones"."iddef", ~r~n' + &
             '       "defscripts"."idds", ~r~n' + &
             '       "defscripts"."ord", ~r~n' + &
             '       "tablas"."nombre" ~r~n' + &
             '  FROM "casos", ~r~n' + &
             '       "casosxtabla", ~r~n' + &
             '       "definiciones", ~r~n' + &
             '       "defscripts", ~r~n' + &
             '       "scripts", ~r~n' + &
             '       "tablas" ~r~n' + &
             ' WHERE "casos"."idc"          = "casosxtabla"."idc"  and  ~r~n' +
&
             '       "casosxtabla"."idc"    = "definiciones"."idc" and  ~r~n' +
&
             '       "definiciones"."iddef" = "defscripts"."iddef" and  ~r~n' +
&
             '       "defscripts"."idscr"   = "scripts"."idscr"    and  ~r~n' +
&
             '       "casosxtabla"."id"     = "tablas"."id"        and~r~n' + &
             '       "defscripts"."seccion" = 2 and~r~n' + &
             '       "definiciones"."iddef" in ( SELECT "definiciones"."iddef"
~r~n' +
```

```
 &
                '                                                FROM "definiciones",  ~r~n' +
&
                '                                                     "defscripts",  ~r~n' + &
                '                                                     "scripts" ~r~n' + &
                '                                               WHERE "definiciones"."iddef" =
"defs
cripts"."iddef" and  ~r~n'    + &
                '                                                     "defscripts"."idscr" =
"script
s"."idscr"       and  ~r~n'    + &
                '                                                     "scripts"."idscr" =
'+Char(39)
+is_value+Char(39)+')  ~r~n' + &
                '   UNION ~r~n' + &
                '  SELECT "casos"."idc", ~r~n' + &
                '         "casos"."nombre", ~r~n' + &
                '         "definiciones"."filename", ~r~n' + &
                '         '+Char(39)+Char(39)+', ~r~n' + &

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:43:25

                '         '+Char(39)+Char(39)+', ~r~n' + &
                '         "definiciones"."iddef", ~r~n' + &
                '         -1, ~r~n' + &
                '         -1, ~r~n' + &
                '         '+Char(39)+Char(39)+'~r~n' + &
                '    FROM "casos", ~r~n' + &
                '         "definiciones" ~r~n' + &
                ' WHERE ( "casos"."idc" = "definiciones"."idc" ) and ~r~n' + &
                '         "definiciones"."iddef" in ( SELECT "definiciones"."iddef"
~r~n'
 + &
                '                                                FROM "definiciones",  ~r~n' +
&
                '                                                     "defscripts",  ~r~n' + &
                '                                                     "scripts" ~r~n' + &
                '                                               WHERE "definiciones"."iddef" =
"defs
cripts"."iddef" and  ~r~n'    + &
                '                                                     "defscripts"."idscr" =
"script
s"."idscr"       and  ~r~n'    + &
                '                                                     "scripts"."idscr" =
'+Char(39)
+is_value+Char(39)+')  ~r~n' + &
               'ORDER BY 1 ASC, ~r~n' + &
                '         3 ASC, ~r~n' + &
                '         9 ASC, ~r~n' + &
                '         8 ASC, ~r~n' + &
                '         6 ASC, ~r~n' + &
                '         7 ASC ~r~n'

  Case 'caso'
    ls_SQL = 'SELECT "casos"."idc", ~r~n' + &
                '       "casos"."nombre", ~r~n' + &
                '       "definiciones"."filename", ~r~n' + &
```

```
                '       "defscripts"."idscr", ~r~n' + &
                '       "scripts"."script", ~r~n' + &
                '       "definiciones"."iddef", ~r~n' + &
                '       "defscripts"."idds", ~r~n' + &
                '       "defscripts"."ord", ~r~n' + &
                '       "tablas"."nombre" ~r~n' + &
                '  FROM "casos", ~r~n' + &
                '       "casosxtabla", ~r~n' + &
                '       "definiciones", ~r~n' + &
                '       "defscripts", ~r~n' + &
                '       "scripts", ~r~n' + &
                '       "tablas" ~r~n' + &
                ' WHERE "casos"."idc"          = "casosxtabla"."idc"  and  ~r~n' +
&
                '       "casosxtabla"."idc"    = "definiciones"."idc" and  ~r~n' +
&
                '       "definiciones"."iddef" = "defscripts"."iddef" and  ~r~n' +
&
                '       "defscripts"."idscr"   = "scripts"."idscr"    and  ~r~n' +
&
                '       "casosxtabla"."id"     = "tablas"."id"        and~r~n' + &
                '       "defscripts"."seccion" = 2 and~r~n' + &
                '       "casos"."nombre"       =
'+Char(39)+is_value+Char(39)+'~r~n'+ &
                '   UNION ~r~n' + &
                '  SELECT "casos"."idc", ~r~n' + &
                '         "casos"."nombre", ~r~n' + &
                '         "definiciones"."filename", ~r~n' + &
                '         '+Char(39)+Char(39)+', ~r~n' + &
                '         '+Char(39)+Char(39)+', ~r~n' + &
                '         "definiciones"."iddef", ~r~n' + &
                '         -1, ~r~n' + &
                '         -1, ~r~n' + &
                '         '+Char(39)+Char(39)+'~r~n' + &
                '    FROM "casos", ~r~n' + &
                '         "definiciones" ~r~n' + &
                '   WHERE ( "casos"."idc" = "definiciones"."idc" ) and ~r~n' + &
                '       "casos"."nombre"       =
'+Char(39)+is_value+Char(39)+'~r~n'+ &
               'ORDER BY 1 ASC, ~r~n' + &

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:43:25

                '         3 ASC, ~r~n' + &
                '         9 ASC, ~r~n' + &
                '         8 ASC, ~r~n' + &
                '         6 ASC, ~r~n' + &
                '         7 ASC ~r~n'
  Case 'tabla'
    ls_SQL = 'SELECT "casos"."idc", ~r~n' + &
                '       "casos"."nombre", ~r~n' + &
                '       "definiciones"."filename", ~r~n' + &
                '       "defscripts"."idscr", ~r~n' + &
                '       "scripts"."script", ~r~n' + &
                '       "definiciones"."iddef", ~r~n' + &
                '       "defscripts"."idds", ~r~n' + &
                '       "defscripts"."ord", ~r~n' + &
```

```
                '       "tablas"."nombre" ~r~n' + &
                '  FROM "casos", ~r~n' + &
                '       "casosxtabla", ~r~n' + &
                '       "definiciones", ~r~n' + &
                '       "defscripts", ~r~n' + &
                '       "scripts", ~r~n' + &
                '       "tablas" ~r~n' + &
                ' WHERE "casos"."idc"          = "casosxtabla"."idc"  and  ~r~n' +
&
                '       "casosxtabla"."idc"    = "definiciones"."idc" and  ~r~n' +
&
                '       "definiciones"."iddef" = "defscripts"."iddef" and  ~r~n' +
&
                '       "defscripts"."idscr"   = "scripts"."idscr"    and  ~r~n' +
&
                '       "casosxtabla"."id"     = "tablas"."id"        and~r~n' + &
                '       "defscripts"."seccion" = 2 and~r~n' + &
                '       "casos"."idc" in ( SELECT "casos"."idc" ~r~n'+ &
                '                            FROM "casos", ~r~n'+ &
                '                                 "casosxtabla", ~r~n'+ &
                '                                 "tablas" ~r~n'+ &
                '                           WHERE "casos"."idc"      =
"casosxtabla"."idc
" and ~r~n'+ &
                '                                 "casosxtabla"."id" =
"tablas"."id"
  and ~r~n'+ &
                '                                 "tablas"."nombre"  =
'+Char(39)+is_valu
e+Char(39)+' ) ~r~n'+ &
                '   UNION ~r~n' + &
                '  SELECT "casos"."idc", ~r~n' + &
                '         "casos"."nombre", ~r~n' + &
                '         "definiciones"."filename", ~r~n' + &
                '         '+Char(39)+Char(39)+', ~r~n' + &
                '         '+Char(39)+Char(39)+', ~r~n' + &
                '         "definiciones"."iddef", ~r~n' + &
                '         -1, ~r~n' + &
                '         -1, ~r~n' + &
                '         '+Char(39)+Char(39)+'~r~n' + &
                '    FROM "casos", ~r~n' + &
                '         "definiciones" ~r~n' + &
                '   WHERE ( "casos"."idc" = "definiciones"."idc" ) and ~r~n' + &
                '       "casos"."idc" in ( SELECT "casos"."idc" ~r~n'+ &
                '                            FROM "casos", ~r~n'+ &
                '                                 "casosxtabla", ~r~n'+ &
                '                                 "tablas" ~r~n'+ &
                '                           WHERE "casos"."idc"      =
"casosxtabla"."idc
" and ~r~n'+ &
                '                                 "casosxtabla"."id" =
"tablas"."id"
  and ~r~n'+ &
                '                                 "tablas"."nombre"  =
'+Char(39)+is_valu
e+Char(39)+' ) ~r~n'+ &
               'ORDER BY 1 ASC, ~r~n' + &
               '         3 ASC, ~r~n' + &
```

```
                '          9 ASC, ~r~n' + &
                '          8 ASC, ~r~n' + &

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:43:25

                '          6 ASC, ~r~n' + &
                '          7 ASC ~r~n'
End Choose

If ls_SQL <> "" Then
  gf_Generate( ls_SQL )
End if



end event

End of Script



MenuItem = m_-1     "-"
Visible = true      Enabled = true

MenuItem = m_editar     "Editar"
Visible = true      Enabled = true

Script for: nt clicked  event
MessageBox( Upper(is_type), 'Editar '+is_value  )
/*Choose Case is_type
  Case 'script'
    MessageBox( 'Script', 'Eliminar '+is_value  )
  Case 'caso'
  Case 'tabla'
End Choose*/
end event

End of Script



MenuItem = m_-      "-"
Visible = true      Enabled = true

MenuItem = m_eliminar     "Eliminar"
Visible = true      Enabled = true

Script for: nt clicked  event
MessageBox( 'Script', 'Eliminar '+is_value  )

/*Choose Case is_type
  Case 'script'
    MessageBox( 'Script', 'Eliminar '+is_value  )
  Case 'caso'
  Case 'tabla'
End Choose*/
```

```
end event

End of Script
```

m_item

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01     Time: 12:46:04

Instance Variables
powerobject ipo

End of Instance Variables


public function integer of_setpo(powerobject apo)

  ipo=apo
Return 1
end function


Menu = m_item
```

## m_repgen

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:47:02

Menu = m_repgen
MenuItem = m_archivo    "&Archivo"
Visible = true     Enabled = true

MenuItem = m_herramientas    "&Herramientas"
Visible = true     Enabled = true

MenuItems for m_archivo
MenuItem = m_guardarcomo    "&Guardar como ..."
Visible = true     Enabled = true

MenuItem = m_6    "-"
Visible = true     Enabled = true

MenuItem = m_salir    "&Salir"
Visible = true     Enabled = true

MenuItems for m_herramientas
MenuItem = m_etiquetas    "&Etiquetas"
Visible = true     Enabled = true

Script for: nt clicked  event
open( w_config, w_main )
end event

End of Script


MenuItem = m_-3    "-"
Visible = true     Enabled = true

MenuItem = m_scripts    "&Scripts"
Visible = true     Enabled = true

Script for: nt clicked  event
open( w_scripts, w_main )
end event

End of Script


MenuItem = m_-    "-"
Visible = true     Enabled = true

MenuItem = m_casos    "&Casos"
Visible = true     Enabled = true

Script for: nt clicked  event
open( w_casos, w_main )
end event
```

```
End of Script


MenuItem = m_-1     "-"
Visible = true      Enabled = true

MenuItem = m_tablas     "&Tablas"
Visible = true      Enabled = true

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01     Time: 12:47:02


Script for: nt clicked  event
open( w_tablas )
end event

End of Script


MenuItem = m_-2     "-"
Visible = true      Enabled = true

MenuItem = m_casosvstablas     "Casos &vs. Tablas"
Visible = true      Enabled = true

Script for: nt clicked  event
open( w_setcasos, w_main )
end event

End of Script
```

m_tags

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01     Time: 12:48:48

public function integer of_popmenu(integer xpos, integer ypos, powerobject apo )
integer i
For i = 1 to UpperBound( This.Item )
  This.Item[i].FUNCTION DYNAMIC of_SetPO( apo )
Next
This.Popmenu( xpos, ypos )
Return 1
end function



public function integer of_additem (string as_text, string as_tag)
m_item lm_menu

lm_menu = CREATE m_item

lm_menu.Text = as_text
lm_menu.tag  = as_tag

This.Item[ UpperBound(This.Item)+1 ] = lm_menu

Return 1
end function


Menu = m_tags
```

## u_cb

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:50:20

User Object: u_cb
X = 0                Y = 0               Width = 325         Height = 109
TabOrder = 1         Visible = true      Enabled = true      Text = "none"
```

## uo_statusbar

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:51:29

User Object: uo_statusbar
X = 0              Y = 0              Width = 2034       Height = 240
TabOrder = 0       Visible = true     Enabled = true     BackColor = 78748035
ObjectType = customvisual!

Instance Variables
Private:
  uostr_columns   istr_Columns[]
  Window    iw_Frame
  integer      ii_DWHeight
  integer      ii_DWWidth
  integer      ii_OleWidth

End of Instance Variables


Script for: nt resize  event
/**** Procesa el ancho del DW ****/
dw_1.X = 1
dw_1.Width  = newwidth

/**** Procesa el alto del DW ****/
// El alto de DW debe ser igual al alto del detalle del DW
dw_1.Y = 1
dw_1.Height     = ii_DWHeight

This.X      = 1
This.Width  = newwidth - 1

This.Y      = newheight - ii_DWHeight + 1
This.Height = ii_DWHeight + 1

of_ResizeDWColumns()

Return 1
end event

End of Script



public function boolean of_setparentwindow (window aw_frame)
If IsValid( aw_frame ) Then
  iw_frame = aw_frame
Else
  SetNull(iw_frame)
End if
return IsValid( aw_frame )
end function
```

```
public subroutine of_message (string as_message)
dw_1.SetItem( 1, "mensaje", as_message )
yield()
end subroutine

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:51:29


private function integer of_resizedwcolumns ()
integer  i
Long     ll_Delta, ll_tmp
String   ls_ColName

ll_Delta = dw_1.Width - ii_DWWidth

For i=1 to UpperBound(istr_columns)
  ls_ColName = istr_Columns[i].colname
  If ls_ColName = 'mensaje' Then
    ll_tmp = Integer( dw_1.Describe( ls_ColName+".Width" ) )
    dw_1.Modify(ls_ColName+".Width='"+String( ll_tmp+ll_Delta )+"'")
  Else
    ll_tmp = Integer( dw_1.Describe( ls_ColName+".X" ) )
    dw_1.Modify(ls_ColName+".X='"+String( ll_tmp+ll_Delta )+"'")
  End if
  If ls_ColName = 'avance' then
     hpb_porcien.X = ll_tmp+ll_Delta
  End if
Next

ii_DWWidth = dw_1.Width

Return 1
end function



private function integer of_getdwcolumns ()
Integer  i, pos, index
String   ls_Objects, ls_Name, ls_Band, ls_X, ls_Width
Boolean  lb_Mensaje = False

ls_Objects = dw_1.Describe("DataWindow.Objects")

Do While ls_Objects <> ""
  pos = Pos( ls_Objects, "~t" )
  If Pos = 0 Then
    ls_Name = ls_Objects
    ls_Objects = ""
  Else
    ls_Name = Mid( ls_Objects, 1, pos - 1 )
    ls_Objects = Mid( ls_Objects, pos + 1 )
  End if
  If ls_Name <> "" Then
    ls_Band  = dw_1.Describe( ls_Name+".Band" )
    If ls_Band = 'detail' Then
```

```
      ls_X      = dw_1.Describe( ls_Name+".X" )
      ls_Width = dw_1.Describe( ls_Name+".Width" )
      index = UpperBound(istr_columns) + 1
      istr_columns[ index ].colname = ls_name
      istr_columns[ index ].x       = integer(ls_X)
      istr_columns[ index ].width   = integer(ls_Width)
      If lower(ls_name) = 'mensaje' Then
        lb_Mensaje = True
      End if
      If lower(ls_name) = 'avance' Then
        ii_OleWidth = istr_columns[ index ].width
      End if
    End if
  End if

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:51:29

Loop

ii_DWWidth = 0

For i=1 to UpperBound(istr_columns)
  If istr_columns[ index ].x + istr_columns[ index ].width > ii_DWWidth Then
    ii_DWWidth = istr_columns[ index ].x + istr_columns[ index ].width
    index = i
  End if
Next

ii_DWWidth += 37

If Not lb_Mensaje Then
  MessageBox( 'Status Bar', "El DW que se definió para los mensajes no tiene la
column
a 'mensaje'  en el detalle", Exclamation! )
End if

hpb_porcien.width = ii_OleWidth

Return 1
end function



public subroutine of_setavance (integer ai_avance)
pointer     oldpointer
oldpointer = SetPointer(HourGlass!)

hpb_porcien.position = ai_avance

SetPointer( oldpointer )
end subroutine



Structure  uostr_columns
   powerobject  classdefinition
   string  colname
```

```
   decimal  x
   decimal  width
End of Structure


DataWindow: dw_1
X = 0                 Y = 4                 Width = 1943          Height = 168
TabOrder = 1          Visible = true        DataObject = "dw_statusbar"
LiveScroll = true     BorderStyle = stylebox!

Script for: nt constructor  event
ii_DWHeight = Integer( dw_1.Describe("DataWindow.Detail.Height") ) + 18
of_GetDWColumns()


end event

End of Script
```

## uo_vars

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:53:31

User Object: uo_vars


Instance Variables
uostr_vars  istr_vars[]
integer     ii_size = 0
uostr_cols  istr_Columns[], istr_pkcolumns[]

CONSTANT STRING is_ColumnList= "column_list"
CONSTANT STRING is_ParmsList= "parms_list"
CONSTANT STRING is_VarsList= "vars_list"
CONSTANT STRING is_ColInsertList= "colinsert_list"
CONSTANT STRING is_TablePK= "table_pk"
CONSTANT STRING is_ColSearch= "column_search"


End of Instance Variables


Script for: nt constructor  event
SetVar( "table_name", "" )
SetVar( is_columnlist, "" )
SetVar( is_tablepk, "" )
SetVar( is_colsearch, "" )
SetVar( is_parmslist, "" )
SetVar( is_varslist, "" )
SetVar( is_colinsertlist, "" )
end event

End of Script



public subroutine resetvars ()
uostr_vars lstr_vars[]

istr_vars = lstr_vars
ii_size   = 0

SetVar( "table_name", "" )
SetVar( is_columnlist, "" )
SetVar( is_tablepk, "" )
SetVar( is_colsearch, "" )
SetVar( is_parmslist, "" )
SetVar( is_varslist, "" )
SetVar( is_colinsertlist, "" )

Return
end subroutine
```

```
private function integer getindex (string id)
integer i

For i = 1 To ii_size
  If istr_vars[i].id = id Then
    Return i
  End if
Next

Return 0

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01     Time: 12:53:31

end function



public function string getvar (string id)
Return istr_vars[ getindex(id)].value
end function



public function integer setvar (string id, string value)
integer index, ret

index = getindex(id)

If  index = 0 Then
  ii_Size ++
  index = ii_Size
  ret = 1
Else
  ret = 0
End if

istr_vars[ index ].id = id
istr_vars[ index ].value = value

If id = 'table_name' Then
  GetTableInfo( value )
End if

return ret


end function



public subroutine replacevars (ref string script)
String   id, value
Integer  i, index, n, ascii, index1, index2

String   ls_Funcs[] = {".FORCOLUMN", ".FORPKCOLUMN"}
String   ls_Funcion, ls_Sentencia, ls_Ciclo, ls_Termino, ls_tmp, ls_FORColumn
```

```
For i=1 To ii_Size
  id    = '%'+istr_Vars[i].id+'%'
  value = istr_Vars[i].value

  Choose Case istr_Vars[i].id
    Case is_ColumnList, is_ParmsList, is_VarsList, is_ColInsertList
      index = Pos( script, id )
      If index > 0 Then
        n = 0
        DO
          n++
          ascii = Asc( Mid( script, index - n, 1 ) )
        LOOP UNTIL Pos( "~r~n", Mid( script, index - n, 1 ) ) > 0   Or ( index -
n ) <
 0
        If index - n < 0 Then
          n = index - 1
        Else
          n = n - 1
        End if
        gf_ReplaceAll( value, "~r~n", "&r'&n'"+Space(n) )

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:53:31

        value = trim( value )
        gf_ReplaceAll( value, "&r'&n'", "~r~n" )
      End if
  End Choose

  gf_ReplaceAll( script, id, value )

Next


For i = 1 To UpperBound( ls_Funcs )
  ls_Funcion = ls_Funcs[i]
  Choose Case ls_Funcion
    Case ".FORCOLUMN"

      index = Pos( script, ls_Funcion )

      Do While index > 0

        getFunction( script, ".FORCOLUMN", ls_Sentencia, ls_Ciclo, ls_Termino,
Index1,
 Index2 )

        ls_ForColumn = ""

        For n = 1 to UpperBound( istr_Columns )
          ls_Tmp = ls_Sentencia
    gf_ReplaceAll( ls_Tmp, "%column_name%", istr_Columns[n].column_name )

    gf_ReplaceAll( ls_Tmp, "%column_rstype%", istr_Columns[n].column_rstype )

          gf_ReplaceAll( ls_Tmp, "%coumns_sqltype%",
istr_Columns[n].column_sqltype )
```

```
          If n = UpperBound( istr_Columns ) Then
            ls_tmp += ls_termino
          Else
            ls_tmp += ls_ciclo + "~r~n"
          End if
          ls_ForColumn += ls_Tmp
        Next

        If index > 0 Then
          n = 0
          DO
            n++
            ascii = Asc( Mid( script, index - n, 1 ) )
          LOOP UNTIL Pos( "~r~n", Mid( script, index - n, 1 ) ) > 0   Or ( index
- n )
 < 0

          If index - n < 0 Then
            n = index - 1
          Else
            n = n - 1
          End if
          gf_ReplaceAll( ls_ForColumn, "~r~n", "&r"&n"+Space(n) )
          value = trim( ls_ForColumn )
          gf_ReplaceAll( ls_ForColumn, "&r"&n"", "~r~n" )
        End if

        script = Replace( script, index1, index2 - index1 + 1, ls_ForColumn )

        index = Pos( script, ls_Funcion )

      Loop

    Case ".FORPKCOLUMN"

_
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:53:31


      index = Pos( script, ls_Funcion )

      Do While index > 0

        getFunction( script, ".FORPKCOLUMN", ls_Sentencia, ls_Ciclo, ls_Termino,
Index
1, Index2 )

        ls_ForColumn = ""

        For n = 1 to UpperBound( istr_PKColumns )
          ls_Tmp = ls_Sentencia
    gf_ReplaceAll( ls_Tmp, "$column_name$", istr_PKColumns[n].column_name )

          If n = UpperBound( istr_PKColumns ) Then
            ls_tmp += ls_termino
          Else
            ls_tmp += ls_ciclo
          End if
```

```
            ls_ForColumn += ls_Tmp
          Next

          If index > 0 Then
            n = 0
            DO
              n++
              ascii = Asc( Mid( script, index - n, 1 ) )
            LOOP UNTIL Pos( "~r~n", Mid( script, index - n, 1 ) ) > 0   Or ( index
- n )
 < 0
            If index - n < 0 Then
              n = index - 1
            Else
              n = n - 1
            End if
            gf_ReplaceAll( ls_ForColumn, "~r~n", "&r"&n*"+Space(n) )
            value = trim( ls_ForColumn )
            gf_ReplaceAll( ls_ForColumn, "&r"&n*", "~r~n" )
          End if

          script = Replace( script, index1, index2 - index1 + 1, ls_ForColumn )

          index = Pos( script, ls_Funcion )

        Loop

    End Choose

Next
end subroutine



protected function integer gettableinfo (string table)
integer  ll_Row, i
String   ls_ColumnList
String   ls_PKList
String   ls_SearchList

uostr_Cols  lstr_Cols[]

String   ls_ParmsList
String   ls_VarsList
String   ls_ColInsertList

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:53:31

String   ls_Col, ls_Type, ls_Len, ls_Prec


istr_columns   = lstr_Cols
istr_pkcolumns = lstr_Cols


If table = ""  Then Return 0
```

```
datastore sp_pkeys
datastore sp_columns

sp_columns = CREATE datastore
sp_pkeys   = CREATE datastore

sp_columns.DataObject = 'dw_sp_columns'
sp_pkeys.DataObject   = 'dw_sp_pkeys'

sp_columns.SetTransObject( SQLServer )
sp_pkeys.SetTransObject( SQLServer )

ll_Row = sp_columns.Retrieve( table )

For ll_Row = 1 to sp_columns.RowCount()

  ls_Col  = sp_columns.GetItemString( ll_Row, "column_name" )
  ls_Type = sp_columns.GetItemString( ll_Row,"type_name" )
  ls_Len  = String( sp_columns.GetItemNumber( ll_Row, "length" ) )
  ls_Prec = String( sp_columns.GetItemNumber( ll_Row, "prec" ) )

  istr_Columns[ ll_Row ].column_name    = ls_Col
  istr_Columns[ ll_Row ].column_sqltype = ls_Type
  istr_Columns[ ll_Row ].column_rstype  = ls_Type
  istr_Columns[ ll_Row ].column_len     = ls_Len
  istr_Columns[ ll_Row ].column_prec    = ls_Prec

  ls_ColumnList += ls_Col + " " + ls_Type
  ls_ParmsList    += "@"+ls_Col + " " + ls_Type
  ls_VarsList     += "@"+ls_Col
  ls_ColInsertList+= ls_Col


  Choose Case ls_Type
    Case "char", "varchar"
      ls_ColumnList += "("+ls_Len+")"
      ls_ParmsList  += "("+ls_Len+")"

      istr_Columns[ ll_Row ].column_sqltype += "("+ls_Len+")"
      istr_Columns[ ll_Row ].column_rstype  += "("+ls_Len+")"


    Case "numeric"
      If Integer(ls_Prec) > 0 Then
        ls_ParmsList  += "("+ls_Len+")"
        istr_Columns[ ll_Row ].column_sqltype += "("+ls_Len+")"
      Else
        ls_ParmsList  += "("+ls_Len+","+ls_Prec+")"
        istr_Columns[ ll_Row ].column_sqltype += "("+ls_Len+","+ls_Prec+")"
      End if
    Case Else
      ls_ColumnList += ""
  End Choose

  If ll_Row < sp_columns.RowCount() Then

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:53:31
```

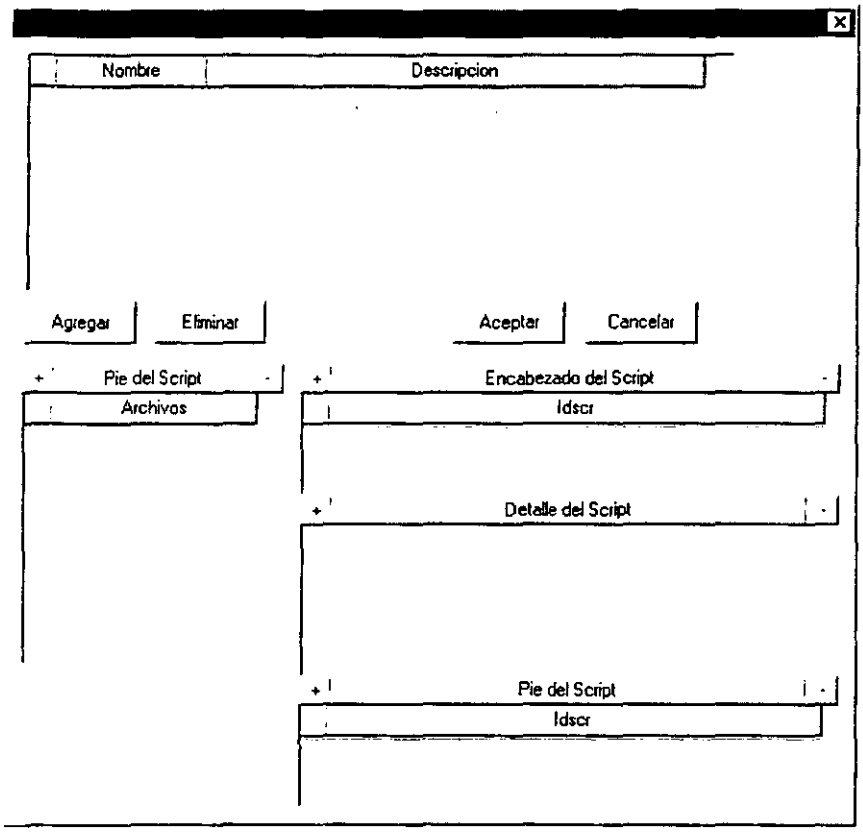
```
    ls_ColumnList     += ",~r~n"
    ls_ParmsList      += ",~r~n"
    ls_VarsList       += ",~r~n"
    ls_ColInsertList += ",~r~n"
  Else
    ls_ColumnList += ""
    ls_ParmsList      += ""
    ls_VarsList       += ""
    ls_ColInsertList += ""
  End if

Next

sp_pkeys.Retrieve( table )
For ll_Row=1 to sp_pkeys.RowCount()
  ls_Col = sp_pkeys.GetItemString( ll_Row, "keyname" )

  istr_PKColumns[ ll_Row ].column_name    = ls_Col

  ls_PKList += ls_Col
  If ll_Row < sp_pkeys.RowCount() Then
    ls_PKList += ","
  End if
Next

ls_SearchList = ls_PKList

SetVar( is_columnlist, ls_ColumnList )
SetVar( is_tablepk, ls_PKList )
SetVar( is_colsearch, ls_SearchList )
SetVar( is_parmslist, ls_ParmsList )
SetVar( is_varslist, ls_VarsList )
SetVar( is_colinsertlist, ls_ColInsertList )

DESTROY sp_columns
DESTROY sp_pkeys


Return 1
end function



private function integer getfunction (ref string as_script, string as_funcion,
ref str
ing as_sentencia, ref string as_parm1, ref string as_parm2, ref integer ai_ini,
ref in
String   ch, chlimit, ls_Ciclo, ls_Termino
integer  index, n, index1, index2, ascii

index = Pos( as_script, as_Funcion )

If Index <= 0 Then Return 0

      n  = index + len( as_Funcion )
      ch = Mid( as_script, n, 1 )
      // Busca (
```

```
        Do While ch <> "("
          n++
          ch = Mid( as_script, n, 1 )
        Loop
        n++
        index1 = n

        // Busca ' ó "
        Do While Pos( "'"+'"', ch ) = 0

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:53:31

          n++
          ch = Mid( as_script, n, 1 )
        Loop
        chlimit = ch
        n++
        ch = Mid( as_script, n, 1 )
        index1 = n

        // Busca ' ó " según haya iniciado la cadena
        Do While ch <> chlimit
          n++
          ch = Mid( as_script, n, 1 )
        Loop
        n++
        ch = Mid( as_script, n, 1 )
        index2 = n

as_Sentencia = Mid( as_script, index1, index2 - index1 - 1 )  // Al obtener "" ó
 '' se delimita la sentencia a procesar

        // Busca inicio de cadena ' ó "
        Do While Pos( "'"+'"', ch ) = 0
          n++
          ch = Mid( as_script, n, 1 )
        Loop
        chlimit = ch
        n++
        ch = Mid( as_script, n, 1 )
        index1 = n

        // Busca ' ó " según haya iniciado la cadena
        Do While ch <> chlimit
          n++
          ch = Mid( as_script, n, 1 )
        Loop
        n++
        ch = Mid( as_script, n, 1 )
        index2 = n


        ls_Ciclo = Mid( as_script, index1, index2 - index1 - 1 )  // Al obtener ""
ó ''
la cadena de ciclo
```

```
        // Busca inicio de cadena ' ó "
        Do While Pos( "'"+'"', ch ) = 0
          n++
          ch = Mid( as_script, n, 1 )
        Loop
        chlimit = ch
        n++
        ch = Mid( as_script, n, 1 )
        index1 = n

        // Busca ' ó " según haya iniciado la cadena
        Do While ch <> chlimit
          n++
          ch = Mid( as_script, n, 1 )
        Loop
        n++
        ch = Mid( as_script, n, 1 )
        index2 = n

        ls_Termino = Mid( as_script, index1, index2 - index1 - 1 )  // Al obtener
"" ó '
' la cadena de término

_
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:53:31


        // Busca )
        Do While ch <> ")"
          n++
          ch = Mid( as_script, n, 1 )
        Loop

        index1 = index
        index2 = n


        as_Parm1 = ls_Ciclo
        as_Parm2 = ls_Termino

        ai_ini   = index1
        ai_fin   = index2

Return 1
end function



Structure  uostr_vars
   powerobject  classdefinition
   string  id
   string  value
End of Structure


Structure  uostr_cols
   powerobject  classdefinition
   string  column_name
```

```
   string  column_sqltype
   string  column_len
   string  column_prec
   string  column_rstype
End of Structure
```

w_casos




```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:54:54

Window: w_casos
X = 297              Y = 496              Width = 2491         Height = 2112
Visible = true       Enabled = true       TitleBar = true      ControlMenu = true
Border = true        WindowType = response!                    WindowState =
normal!
BackColor = 78748035

Instance Variables
long  il_idc, il_iddef

End of Instance Variables


Script for: nt open  event
dw_casos.retrieve()
```

```
end event

End of Script

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:54:54


MultilineEdit: mle_script
X = 846              Y = 1244             Width = 1582         Height = 400
TabOrder = 11        Limit = 0            Visible = true       Enabled = true
Border = true        TabStop = true       TextColor = 33554432
BackColor = 1073741824                    Alignment = left!    TextCase = anycase!
BorderStyle = stylelowered!

Script for: nt constructor  event
Visible = False
end event

End of Script


CommandButton: cb_add_defscr_h
X = 846              Y = 836              Width = 96           Height = 76
TabOrder = 160       Visible = true       Enabled = true       Text = "+"

Script for: nt clicked  event
long ll_Row

ll_Row = dw_defscripts_h.InsertRow(0)

dw_defscripts_h.SetItem( ll_Row, "iddef", il_iddef )
dw_defscripts_h.SetItem( ll_Row, "seccion", 1 )
end event

End of Script


StaticText: st_3
X = 937              Y = 836              Width = 1394         Height = 76
TabOrder = 0         Visible = true       Text = "Encabezado del Script"
TextColor = 33554432                      BackColor = 67108864
Border = true        BorderColor = 0      BorderStyle = styleraised!
Alignment = center! FillPattern = solid!


StaticText: st_2
X = 937              Y = 1176             Width = 1394         Height = 76
TabOrder = 0         Visible = true       Text = "Detalle del Script"
TextColor = 33554432                      BackColor = 67108864
Border = true        BorderColor = 0      BorderStyle = styleraised!
Alignment = center! FillPattern = solid!


CommandButton: cb_add_defscr_d
```

```
X = 846              Y = 1176             Width = 96           Height = 76
TabOrder = 110       Visible = true       Enabled = true       Text = "+"

Script for: nt clicked  event
long ll_Row

ll_Row = dw_defscripts_d.InsertRow(0)

dw_defscripts_d.SetItem( ll_Row, "iddef", il_iddef )
dw_defscripts_d.SetItem( ll_Row, "seccion", 2 )
end event

End of Script

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:54:54


DataWindow: dw_defscripts_h
X = 846              Y = 904              Width = 1577         Height = 272
TabOrder = 150       Visible = true       Enabled = true
DataObject = "dw_defscripts"              VScrollBar = true    Border = true
LiveScroll = true    BorderStyle = stylelowered!

Script for: nt constructor  event
This.SetRowFocusIndicator( p_1 )
This.Settransobject( SQLAny50 )
end event

End of Script


Script for: nt losefocus  event
This.Update()
end event

End of Script


Picture: p_1
X = 841              Y = 932              Width = 73           Height = 72
TabOrder = 0         Visible = true       Enabled = true
PictureName = "D:\Archivos de programa\Sybase\PowerBuilder 7.0\Code
Examples\Example A
BorderStyle = stylebox!

Script for: nt constructor  event
Visible = False
end event

End of Script


DataWindow: dw_defscripts_d
```

```
X = 846                 Y = 1244                Width = 1577          Height = 400
TabOrder = 130          Visible = true          Enabled = true
DataObject = "dw_defscripts"                    VScrollBar = true     Border = true
LiveScroll = true    BorderStyle = stylelowered!

Script for: nt constructor  event
//This.SetRowFocusIndicator( p_1, 0, 0 )
This.Settransobject( SQLAny50 )
end event

End of Script


Script for: nt losefocus  event
This.Update()
end event

End of Script

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:54:54


CommandButton: cb_5
X = 32                  Y = 836                 Width = 96            Height = 76
TabOrder = 60           Visible = true          Enabled = true        Text = "+"

Script for: nt clicked  event
long ll_Row

ll_Row = dw_definiciones.InsertRow(0)
dw_definiciones.SetItem( ll_Row, "idc", il_idc )
end event

End of Script


User Object: cb_4
X = 1682                Y = 676                 Width = 325           Height = 108
TabOrder = 20           Visible = true          Enabled = true        Text = "Cancelar"

Script for: nt clicked  event
call super::clicked;Close( parent )
end event

End of Script


User Object: cb_3
X = 1298                Y = 676                 Width = 325           Height = 108
TabOrder = 30           Visible = true          Enabled = true        Text = "Aceptar"

Script for: nt clicked  event
call super::clicked;Integer li_ok
```

```
SQLAny50.Autocommit = False

li_ok = dw_casos.Update( true )
If li_ok = -1 Then goto ERROR

li_ok = dw_definiciones.Update( true )
If li_ok = -1 Then goto ERROR

li_ok = dw_defscripts_h.Update( true )
If li_ok = -1 Then goto ERROR

li_ok = dw_defscripts_d.Update( true )
If li_ok = -1 Then goto ERROR

li_ok = dw_defscripts_f.Update( true )
If li_ok = -1 Then goto ERROR

COMMIT USING SQLAny50;

SQLAny50.Autocommit = True

Close( Parent )

Return

ERROR:

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:54:54

ROLLBACK USING SQLAny50;

SQLAny50.Autocommit = True

Return
end event

End of Script


User Object: cb_2
X = 421              Y = 676              Width = 325          Height = 108
TabOrder = 40        Visible = true       Enabled = true       Text = "Eliminar"

Script for: nt clicked  event
call super::clicked;dw_casos.DeleteRow(0)
end event

End of Script


User Object: cb_1
X = 41               Y = 676              Width = 325          Height = 108
TabOrder = 50        Visible = true       Enabled = true       Text = "Agregar"
```

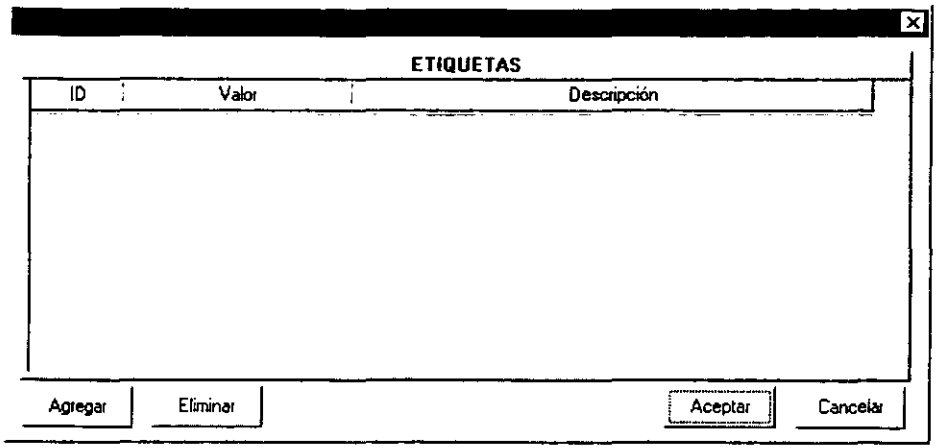
```
Script for: nt clicked  event
call super::clicked;Long ll_Row
ll_Row = dw_casos.InsertRow(0)

dw_casos.SetRow( ll_Row )
dw_casos.ScrollToRow( ll_Row )
dw_casos.SetColumn("nombre")
end event

End of Script


DataWindow: dw_definiciones
X = 32              Y = 904             Width = 763         Height = 708
TabOrder = 170      Visible = true      Enabled = true
DataObject = "dw_definiciones"          VScrollBar = true   Border = true
LiveScroll = true   BorderStyle = stylelowered!

Script for: nt constructor  event
This.SetRowFocusIndicator( p_1 )
This.Settransobject( SQLAny50 )
end event

End of Script


Script for: nt losefocus  event

This.Update()
end event

End of Script


Script for: nt rowfocuschanged  event
If currentrow >= 1 And currentrow <= This.RowCount() Then
_
  dw_defscripts_h.retrieve( il_iddef, 1 )
  dw_defscripts_d.retrieve( il_iddef, 2 )
  dw_defscripts_f.retrieve( il_iddef, 3 )
Else
  dw_defscripts_h.Reset()
  dw_defscripts_d.Reset()
  dw_defscripts_f.Reset()
  il_iddef = -1
End if
end event

End of Script


DataWindow: dw_casos
X = 41              Y = 36              Width = 2071        Height = 616
```

```
TabOrder = 10        Visible = true       Enabled = true       DataObject =
"dw_casos"
VScrollBar = true    Border = true        LiveScroll = true
BorderStyle = stylelowered!

Script for: nt constructor  event
This.SetRowFocusIndicator( p_1 )
This.Settransobject( SQLAny50 )
end event

End of Script


Script for: nt rowfocuschanged  event
If currentrow >= 1 And currentrow <= This.RowCount() Then
  il_idc = This.GetItemNumber(currentrow,"idc")
  dw_definiciones.retrieve( il_idc )
Else
  dw_definiciones.Reset()
  il_idc = -1
End if
end event

End of Script


CommandButton: cb_del_defscr_h
X = 2327             Y = 836              Width = 96           Height = 76
TabOrder = 100       Visible = true       Enabled = true       Text = "-"

Script for: nt clicked  event
dw_defscripts_h.Deleterow(0)
end event

End of Script


DataWindow: dw_defscripts_f
X = 841              Y = 1708             Width = 1577         Height = 272
TabOrder = 140       Visible = true       Enabled = true
DataObject = "dw_defscripts"              VScrollBar = true    Border = true
LiveScroll = true    BorderStyle = stylelowered!

Script for: nt constructor  event
This.SetRowFocusIndicator( p_1 )
This.Settransobject( SQLAny50 )
end event

End of Script
-
Script for: nt losefocus  event
This.Update()
end event
```

```
End of Script


Script for: nt sqlpreview  event
//
If False Then
End if
end event

End of Script


StaticText: st_1
X = 123              Y = 836              Width = 581          Height = 76
TabOrder = 0         Visible = true       Text = "Pie del Script"
TextColor = 33554432                      BackColor = 67108864
Border = true        BorderColor = 0      BorderStyle = styleraised!
Alignment = center! FillPattern = solid!


CommandButton: cb_6
X = 699              Y = 836              Width = 96           Height = 76
TabOrder = 70        Visible = true       Enabled = true       Text = "-"

Script for: nt clicked  event
dw_definiciones.Deleterow(0)
end event

End of Script


CommandButton: cb_add_defscr_f
X = 846              Y = 1640             Width = 96           Height = 76
TabOrder = 80        Visible = true       Enabled = true       Text = "+"

Script for: nt clicked  event
long ll_Row

ll_Row = dw_defscripts_f.InsertRow(0)

dw_defscripts_f.SetItem( ll_Row, "iddef", il_iddef )
dw_defscripts_f.SetItem( ll_Row, "seccion", 3 )
end event

End of Script


StaticText: st_4
X = 937              Y = 1640             Width = 1394         Height = 76
TabOrder = 0         Visible = true       Text = "Pie del Script"
TextColor = 33554432                      BackColor = 67108864
Border = true        BorderColor = 0      BorderStyle = styleraised!
Alignment = center! FillPattern = solid!
```

```
CommandButton: cb_del_defscr_f
X = 2327            Y = 1640            Width = 96          Height = 76
TabOrder = 120      Visible = true      Enabled = true      Text = "-"

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:54:54


Script for: nt clicked  event
dw_defscripts_f.Deleterow(0)
end event

End of Script


CommandButton: cb_del_defscr_d
X = 2327            Y = 1176            Width = 96          Height = 76
TabOrder = 90       Visible = true      Enabled = true      Text = "-"

Script for: nt clicked  event
dw_defscripts_d.Deleterow(0)
end event

End of Script
```

w_config




```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:57:26

Window: w_config
X = 1075              Y = 485              Width = 2707         Height = 1129
Visible = true        Enabled = true       TitleBar = true      ControlMenu = true
Border = true         WindowType = response!                    WindowState =
normal!
BackColor = 78748035


User Object: cb_1
X = 42                Y = 893              Width = 325          Height = 109
TabOrder = 21         Visible = true       Enabled = true       Text = "Agregar"

Script for: nt clicked  event
call super::clicked;Long ll_Row
ll_Row = dw_config.InsertRow(0)

dw_config.SetRow( ll_Row )
dw_config.ScrollToRow( ll_Row )
dw_config.SetColumn("idcfg")
end event

End of Script


User Object: cb_2
X = 421               Y = 893              Width = 325          Height = 109
TabOrder = 20         Visible = true       Enabled = true       Text = "Eliminar"

Script for: nt clicked  event
call super::clicked;dw_config.DeleteRow(0)
```

```
end event

End of Script

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:57:26


User Object: cb_3
X = 1911            Y = 893             Width = 325         Height = 109
TabOrder = 10       Visible = true      Enabled = true      Text = "Aceptar"

Script for: nt clicked  event
call super::clicked;Integer li_ok

li_ok = dw_config.Update( true )
If li_ok = -1 Then goto ERROR

Close( Parent )

Return

ERROR:

Return
end event

End of Script


User Object: cb_4
X = 2295            Y = 893             Width = 325         Height = 109
TabOrder = 40       Visible = true      Enabled = true      Text = "Cancelar"

Script for: nt clicked  event
call super::clicked;Close( parent )
end event

End of Script


StaticText: st_2
X = 33              Y = 33              Width = 2602        Height = 77
TabOrder = 0        Visible = true      Text = "ETIQUETAS"  TextColor = 33554432
BackColor = 67108864                    Border = true       BorderColor = 0
BorderStyle = styleraised!              Alignment = center! FillPattern = solid!


DataWindow: dw_config
X = 51              Y = 101             Width = 2565        Height = 765
TabOrder = 30       Visible = true      Enabled = true      DataObject =
"dw_config"
VScrollBar = true   Border = true       LiveScroll = true
BorderStyle = stylelowered!
```

```
Script for: nt constructor  event
Settransobject( SQLAny50 )
Retrieve()
end event

End of Script

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:57:26


StaticText: st_1
X = 33              Y = 85              Width = 2602        Height = 797
TabOrder = 0        Visible = true      TextColor = 33554432
BackColor = 67108864                    Border = true       BorderColor = 0
BorderStyle = styleraised!              Alignment = left!   FillPattern = solid!
```

## w_main

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01     Time: 12:58:39

Window: w_main
X = 535              Y = 528              Width = 2949         Height = 1836
Visible = true       Enabled = true       TitleBar = true
Title = "Replication Server"              MenuName = "m_repgen"
ControlMenu = true   MinBox = true        MaxBox = true        Resizable = true
WindowType = main!   WindowState = normal!                     BackColor = 78748035
Icon = "D:\Mis documentos\Power Builder\StoreRep\repgen.ico"
Pointer = "SizeWE!"

Instance Variables
long  il_Delta
long  il_hCasos
long  il_hScripts
long  il_hTablas
m_edit   im_Edit

End of Instance Variables


Script for: nt close  event
DESTROY im_edit
-

End of Script



Script for: nt mousemove  event
If flags=1 Then
  If This.Pointer = 'SizeWE!' Then
     /*lv_main.X     = xpos + il_Delta
     lv_main.Width = This.Width - lv_main.X + 1 - (2 * tv_main.X)
     tv_main.Width = lv_main.X - ( tv_main.X * 1.5 ) */
  End if
Else
  If xpos >= tv_main.Width  and xpos <= lv_main.X Then
    This.Pointer = 'SizeWE!'
    il_Delta = lv_main.X - xpos
    If il_Delta = 0 Then
      il_Delta = 10
    End if

  Else
    This.Pointer = 'arrow!'
  End if

End if
end event
```

```
End of Script


Script for: nt open  event
tv_main.Width = ( This.Width - ( 3 * tv_main.X ) ) / 2
lv_main.Y     = tv_main.Y
lv_main.X     = ( tv_main.X * 1.5 ) + tv_main.Width
lv_main.Width = This.Width - lv_main.X + 1 - (2 * tv_main.X)


/*w_main.dw_statusbar.SetItem( 1, "fecha", string(  today(), "dd/mm/yyyy" ) )
w_main.dw_statusbar.SetItem( 1, "hora", string(  now(), "hh:mm:ss" ) )*/


im_Edit = CREATE m_edit
end event

End of Script


Script for: nt resize  event

uo_1.Event Resize( newwidth, newheight )

Return 1
end event

End of Script


User Object: uo_1
X = 1490              Y = 1064              Width = 2034          Height = 228
TabOrder = 21         Visible = true        Enabled = true        Border = true
BackColor = 78748035                        ObjectType = customvisual!

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:58:39


CommandButton: cb_1
X = 1303              Y = 1488              Width = 247           Height = 108
TabOrder = 30         Visible = true        Enabled = true        Text = "none"

Script for: nt clicked  event
Long        ll_Caso
String      ls_File = "", ls_FileName
String      ls_TableName = ""
Long        ll_Row
Boolean     lb_FileOpened = False
Integer     li_Handle
String      ls_Path = "D:\MIS DOCUMENTOS\POWER BUILDER\STOREREP\DEMO\"
String      ls_Script = ""
datastore   lds_proceso
uo_Vars     luo_Vars
```

```
Pointer     oldpointer

oldpointer = SetPointer(HourGlass!)

lds_proceso = CREATE datastore

lds_proceso.DataObject = "dw_config"
lds_proceso.Settransobject( SQLAny50 )
lds_proceso.Retrieve()
luo_Vars.Resetvars()
For ll_Row = 1 To lds_Proceso.RowCount()
  luo_Vars.Setvar( lds_proceso.getitemstring( ll_Row, "idcfg" ), &
                   lds_proceso.getitemstring( ll_Row, "value" ) )
Next

lds_proceso.DataObject = "dw_process"
lds_proceso.Settransobject( SQLAny50 )
lds_proceso.Retrieve()


uo_1.of_Message( "Generando Scripts..." )
uo_1.of_SetAvance( 0 )

For ll_Row = 1 To lds_Proceso.RowCount()

  if ll_Caso <> lds_Proceso.GetItemNumber( ll_Row, "caso" ) Then
    ll_Caso = lds_Proceso.GetItemNumber( ll_Row, "caso" )
    If lb_FileOpened Then
      FileClose( li_Handle )
      lb_FileOpened = False
    End if
  end if

  if ls_File <> lds_Proceso.GetItemString( ll_Row, "archivo" ) Then
    ls_File = lds_Proceso.GetItemString( ll_Row, "archivo" )
    If lb_FileOpened Then
      FileClose( li_Handle )
      lb_FileOpened = False
    End if
  end if

  If Not lb_FileOpened Then
    ls_FileName = ls_Path+ls_File+"."+String( ll_Caso, "000" )
    li_Handle = FileOpen( ls_Filename, StreamMode!, Write!, LockWrite!, Replace!
)
    lb_FileOpened = True
  End if


Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:58:39

  if ls_TableName <> lds_Proceso.GetItemString( ll_Row, "tabla" ) Then
    ls_Tablename = lds_Proceso.GetItemString( ll_Row, "tabla" )
    luo_Vars.SetVar( "table_name", ls_TableName )
  end if

  ls_Script = lds_Proceso.GetItemString( ll_Row, "script" )
```

```
   luo_Vars.ReplaceVars( ls_Script )

   FileWrite( li_Handle, ls_Script )

   uo_1.of_SetAvance( Integer( ( ll_Row / lds_Proceso.RowCount() ) * 100 ) )

Next


FileClose( li_Handle )
lb_FileOpened = False

DESTROY lds_proceso

uo_1.of_Message( "Listo." )

SetPointer(oldpointer)
end event

End of Script



ListView: lv_main
X = 562              Y = 24               Width = 645          Height = 1744
TabOrder = 20        Visible = true       Enabled = true       Border = true
TextColor = 0        BackColor = 16777215
BorderStyle = stylelowered!               ButtonHeader = true ShowHeader = true
LabelWrap = true     Scrolling = true     HideSelection = true
SortType = unsorted!                      View = listviewreport!
LargePictureWidth = 32                    LargePictureHeight = 32
LargePictureMaskColor = 553648127         SmallPictureWidth = 16
SmallPictureHeight = 16                   SmallPictureMaskColor = 553648127
StatePictureWidth = 0                     StatePictureHeight = 0
StatePictureMaskColor = 553648127
LargePictureName = { "D:\Mis documentos\Power Builder\StoreRep\repgen.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\scripts.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\scriptssl.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\script.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\scriptsl.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\tables.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\tablessl.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\table.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\tablesl.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\casos.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\casossl.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\caso.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\casosl.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\scriptadd.ico" }
SmallPictureName = { "D:\Mis documentos\Power Builder\StoreRep\repgen.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\scripts.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\scriptssl.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\script.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\scriptsl.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\tables.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\tablessl.ico", &
       "D:\Mis documentos\Power Builder\StoreRep\table.ico", &
```

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:58:39

        "D:\Mis documentos\Power Builder\StoreRep\tablesl.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\casos.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\casossl.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\caso.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\casosl.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\scriptadd.ico" }

Script for: nt constructor  event
This.AddColumn("ID" , Left! , 500)
This.AddColumn("Descripción" , Left! , 1000)
This.AddColumn("Fecha/Hora" , Left! , 300)
end event

End of Script



Script for: nt rightclicked  event
ListViewItem   llvi_Item
String         ls_Type

This.GetItem( index, llvi_Item )

Choose Case llvi_Item.PictureIndex
  Case 4   // scripts
    ls_Type = 'script'
  Case 8   // Tablas
    ls_Type = 'tabla'
  Case 12  // Casos
    ls_Type = 'caso'
End Choose

Choose Case llvi_Item.PictureIndex
  Case 4,8,12
    im_Edit.of_SetType( ls_Type, String( llvi_Item.Label ) )
    im_Edit.PopMenu( 1, 1 )
End Choose
end event

End of Script



TreeView: tv_main
X = 27              Y = 20              Width = 494         Height = 1744
TabOrder = 10       Visible = true      Enabled = true      Border = true
TextColor = 0       BackColor = 16777215
BorderStyle = stylelowered!             HasButtons = true   HasLines = true
LinesAtRoot = true  DisableDragDrop = true                  HideSelection = true
Indent = 0          SortType = unsorted!                    PictureWidth = 16
PictureHeight = 16  PictureMaskColor = 553648127            StatePictureWidth =
0
StatePictureHeight = 0                  StatePictureMaskColor = -1
PictureName = { "D:\Mis documentos\Power Builder\StoreRep\repgen.ico", &
```

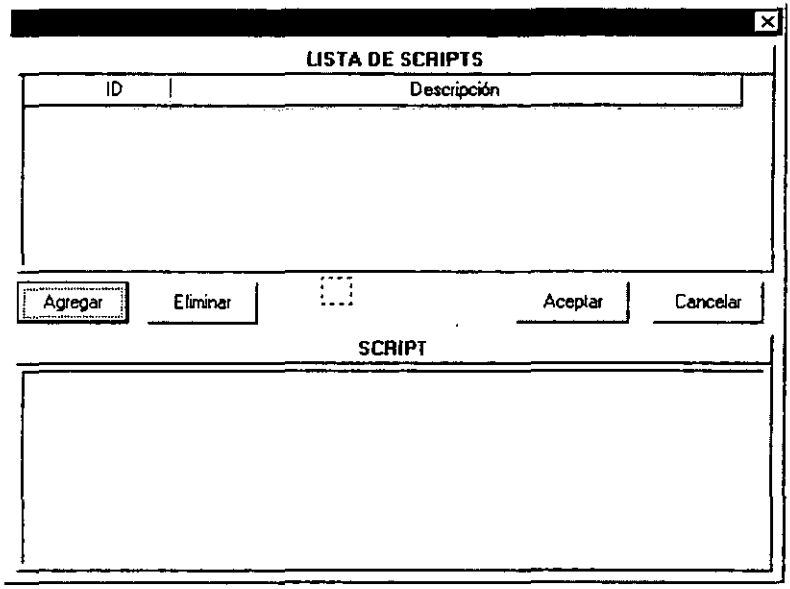
```
        "D:\Mis documentos\Power Builder\StoreRep\scripts.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\scriptssl.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\script.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\scriptsl.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\tables.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\tablessl.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\table.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\tablesl.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\casos.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\casossl.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\caso.ico", &

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:58:39

        "D:\Mis documentos\Power Builder\StoreRep\casosl.ico", &
        "D:\Mis documentos\Power Builder\StoreRep\scriptadd.ico" }

Script for: nt constructor  event
long handle1
treeviewitem tvi_item

tvi_item.label        = 'Replication Server'
tvi_item.data         = 'Replication Server~t'
tvi_item.PictureIndex = 1
tvi_item.SelectedPictureIndex = 1
handle1 = tv_main.InsertItemFirst( 0, tvi_item )

tvi_item.label        = 'Scripts'
tvi_item.data         = 'Scripts~t'
//tvi_item.Children     = True
tvi_item.PictureIndex = 2
tvi_item.SelectedPictureIndex = 3
tvi_item.StatePictureIndex = 3
il_hScripts = tv_main.InsertItemLast( handle1, tvi_item )

tvi_item.label        = 'Casos'
tvi_item.data         = 'Casos~t'
//tvi_item.Children     = True
tvi_item.PictureIndex = 10
tvi_item.SelectedPictureIndex = 11
tvi_item.StatePictureIndex = 11
il_hCasos  = tv_main.InsertItemLast( handle1, tvi_item)

tvi_item.label        = 'Tablas'
tvi_item.data         = 'Tablas~t'
//tvi_item.Children     = True
tvi_item.PictureIndex = 6
tvi_item.SelectedPictureIndex = 7
tvi_item.StatePictureIndex = 7
il_hTablas = tv_main.InsertItemLast( handle1, tvi_item )


end event

End of Script
```

```
Script for: nt itemexpanded  event
//Long ll_tvi
//
//ll_tvi = tv_main.FindItem(ChildTreeItem!,handle)
//
//tv_main.SelectItem( handle )
end event

End of Script


Script for: nt rightclicked  event
TreeViewItem   ltvi_Item
String         ls_Type

This.GetItem( handle, ltvi_Item )

Choose Case ltvi_Item.PictureIndex
  Case 4   // scripts

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:58:39

    ls_Type = 'script'
  Case 8   // Tablas
    ls_Type = 'tabla'
  Case 12  // Casos
    ls_Type = 'caso'
End Choose

Choose Case ltvi_Item.PictureIndex
  Case 4,8,12
    im_Edit.of_SetType( ls_Type, String( ltvi_Item.Label ) )
    im_Edit.PopMenu( 1, 1 )
End Choose



end event

End of Script


Script for: nt selectionchanged  event
treeviewitem   tvi_item
listviewitem   lvi_item
datastore      ds_fill
Long           ll_Row
Long           ll_tvi
Pointer        old_Pointer

old_Pointer = SetPointer( HourGlass! )

ds_fill = CREATE datastore
```

```
This.GetItem( newhandle, tvi_item )

Choose Case tvi_item.level
  Case 2

    Choose Case lower(tvi_item.Label)
      Case 'scripts'

        If Not tvi_item.Children Then
          ds_fill.DataObject = 'dw_scripts1'
          ds_fill.SetTransObject( SQLAny50 )
          ds_fill.Retrieve()
          tv_main.SetRedraw( false )
          For ll_Row = 1 to ds_fill.RowCount()
            tvi_item.Label = ds_fill.GetItemString( ll_Row, "idscr" )
            tvi_item.Data  = ds_fill.GetItemString( ll_Row, "idscr"
)+"~t"+ds_fill.Get
ItemString( ll_Row, "descripcion" )
            tvi_item.PictureIndex = 4
            tvi_item.SelectedPictureIndex = 5
            tv_main.InsertItemLast( il_hScripts, tvi_item )
          Next
          tv_main.SetRedraw( true )
        End if

        lvi_item.Label        = 'Agregar'
        lvi_item.PictureIndex = 14

      Case 'casos'

        If Not tvi_item.Children Then
          ds_fill.DataObject = 'dw_casos'

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:58:39

          ds_fill.SetTransObject( SQLAny50 )
          ds_fill.Retrieve()
          tv_main.SetRedraw( false )
          For ll_Row = 1 to ds_fill.RowCount()
            tvi_item.Label = ds_fill.GetItemString( ll_Row, "nombre" )
            tvi_item.Data  = ds_fill.GetItemString( ll_Row, "nombre"
)+"~t"+ds_fill.Ge
tItemString( ll_Row, "descripcion" )
            tvi_item.PictureIndex = 12
            tvi_item.SelectedPictureIndex = 13
            tv_main.InsertItemLast( il_hCasos, tvi_item )
          Next
          tv_main.SetRedraw( true )
        End if


        lvi_item.Label        = 'Agregar'
        lvi_item.PictureIndex = 14


      Case 'tablas'
```

```
        If Not tvi_item.Children Then
          ds_fill.DataObject = 'ddw_tablas'
          ds_fill.SetTransObject( SQLAny50 )
          ds_fill.Retrieve()
          tv_main.SetRedraw( false )
          For ll_Row = 1 to ds_fill.RowCount()
            tvi_item.Label = ds_fill.GetItemString( ll_Row, "nombre" )
            tvi_item.Data  = ds_fill.GetItemString( ll_Row, "nombre" )+"~t"
            tvi_item.PictureIndex = 8
            tvi_item.SelectedPictureIndex = 9
            tvi_item.StatePictureIndex = 9
            tv_main.InsertItemLast( il_hTablas, tvi_item )
          Next
          tv_main.SetRedraw( true )
        End if

        lvi_item.Label        = 'Agregar'
        lvi_item.PictureIndex = 14

    End Choose

    lv_main.SetRedraw( false )
    lv_main.DeleteItems()
    If lvi_item.Label = 'Agregar' Then
      lv_main.AddItem( lvi_item  )
    End if
    ll_tvi = tv_main.FindItem(ChildTreeItem!,newhandle)
    Do While ll_tvi <> -1
      This.GetItem( ll_tvi, tvi_item )
      lvi_item.Label = tvi_item.Data
      lvi_item.PictureIndex = tvi_item.PictureIndex
      lv_main.AddItem( lvi_item  )
      ll_tvi = tv_main.FindItem(NextTreeItem!,ll_tvi)
    Loop
    lv_main.SetRedraw( true )


  Case 3

End Choose

DESTROY ds_fill

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 12:58:39

SetPointer( old_Pointer )
end event

End of Script
```

W_scripts




```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:01:52

Window: w_scripts
X = 1075              Y = 485              Width = 2277         Height = 1493
Visible = true        Enabled = true       TitleBar = true      ControlMenu = true
Border = true         WindowType = response!                    WindowState =
normal!
BackColor = 78748035

Instance Variables
Protected:
m_tags im_popup

End of Instance Variables


Script for: nt close  event
DESTROY im_popup
end event

End of Script



Script for: nt open  event
datastore   lds_Config
integer     i
```

```
dw_scriptsl.Retrieve()

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:01:52


im_popup = CREATE m_tags

lds_Config = CREATE datastore
lds_Config.DataObject = 'dw_config'
lds_Config.SetTransObject( SQLAny50 )
lds_Config.Retrieve()
For i = 1 to lds_Config.Rowcount()
  im_popup.of_AddItem( lds_Config.getItemString( i, 'descripcion' ),
lds_Config.getIte
mString( i, 'idcfg') )
Next

DESTROY lds_Config
end event

End of Script



MultilineEdit: mle_script
X = 37              Y = 853              Width = 2172         Height = 505
TabOrder = 20       Limit = 0            Visible = true       Enabled = true
Border = true       AutoHScroll = true   AutoHScroll = true   TabStop = true
TextColor = 33554432                     BackColor = 1073741824
Alignment = left!   TextCase = anycase!  BorderStyle = stylelowered!

Script for: nt losefocus  event
dw_scriptsl.SetItem( dw_scriptsl.GetRow(), "script", This.text )
end event

End of Script


Script for: nt rbuttondown  event
If flags = 6 Then
  im_popup.of_Popmenu( this.x+xpos, this.y+ypos, This )
End if


end event

End of Script



Script for: nt replacetext  event
Return This.ReplaceText ( "%"+as_text+"%" )

end event
```

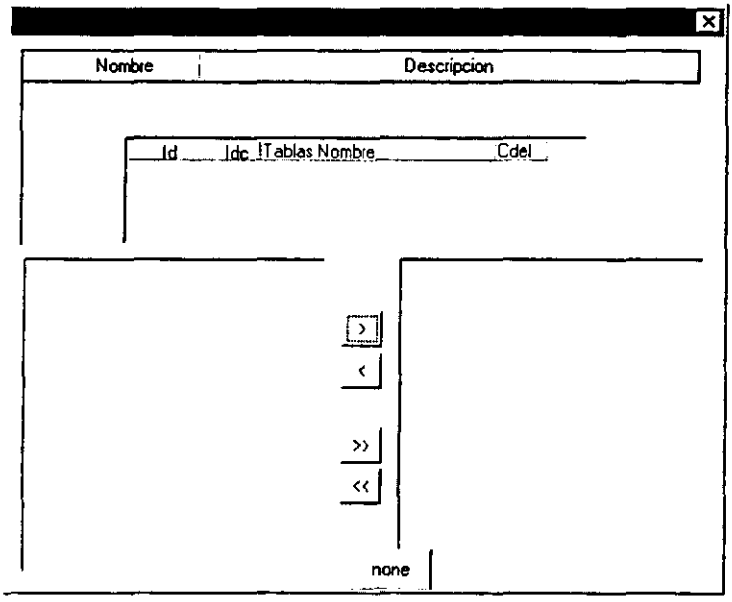
```
End of Script


Picture: p_1
X = 910                 Y = 617                 Width = 92              Height = 73
TabOrder = 0            Visible = true          Enabled = true
PictureName = "d:\mis documentos\power builder\storerep\left.bmp"
BorderStyle = stylebox!

Script for: nt constructor  event
Visible = false
end event

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01     Time: 13:01:52


End of Script


User Object: cb_4
X = 1875                Y = 629                 Width = 325             Height = 109
TabOrder = 30           Visible = true          Enabled = true          Text = "Cancelar"
Cancel = true

Script for: nt clicked  event
call super::clicked;Close( parent )
end event

End of Script


User Object: cb_3
X = 1482                Y = 629                 Width = 325             Height = 109
TabOrder = 40           Visible = true          Enabled = true          Text = "Aceptar"
Default = true

Script for: nt clicked  event
call super::clicked;If dw_scriptsl.Update( true ) = 1 Then
  Close( Parent )
End if
end event

End of Script


User Object: cb_2
X = 403                 Y = 629                 Width = 325             Height = 109
TabOrder = 60           Visible = true          Enabled = true          Text = "Eliminar"

Script for: nt clicked  event
call super::clicked;dw_scriptsl.DeleteRow(0)
end event
```

```
End of Script


User Object: cb_1
X = 23              Y = 629             Width = 325         Height = 109
TabOrder = 10       Visible = true      Enabled = true      Text = "Agregar"

Script for: nt clicked  event
call super::clicked;Long ll_Row
ll_Row = dw_scripts1.InsertRow(0)

dw_scripts1.SetRow( ll_Row )
dw_scripts1.ScrollToRow( ll_Row )
dw_scripts1.SetColumn("idscr")
end event

End of Script

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:01:52


StaticText: st_4
X = 23              Y = 765             Width = 2204        Height = 77
TabOrder = 0        Visible = true      Text = "SCRIPT"     TextColor = 0
BackColor = 78748035                    Border = true       BorderColor = 0
BorderStyle = styleraised!              Alignment = center! FillPattern = solid!


StaticText: st_3
X = 23              Y = 21              Width = 2204        Height = 77
TabOrder = 0        Visible = true      Text = "LISTA DE SCRIPTS"
TextColor = 0       BackColor = 78748035                    Border = true
BorderColor = 0     BorderStyle = styleraised!              Alignment = center!
FillPattern = solid!


StaticText: st_1
X = 23              Y = 841             Width = 2204        Height = 533
TabOrder = 0        Visible = true      TextColor = 0       BackColor = 78748035
Border = true       BorderColor = 0     BorderStyle = styleraised!
Alignment = left!   FillPattern = solid!


DataWindow: dw_scripts1
X = 33              Y = 89              Width = 2177        Height = 505
TabOrder = 50       Visible = true      Enabled = true
DataObject = "dw_scripts1"              VScrollBar = true   Border = true
LiveScroll = true   BorderStyle = stylelowered!

Script for: nt constructor  event
This.SetTransObject( SQLAny50 )
This.SetRowFocusIndicator(p_1,5,0)
end event

End of Script
```

```
Script for: nt rowfocuschanged  event
If currentrow >= 1 And currentrow <= RowCount() Then
  mle_script.Text = This.GetItemString( currentrow, "script" )
End if

end event

End of Script
```

```
StaticText: st_2
X = 23              Y = 77              Width = 2204        Height = 529
TabOrder = 0        Visible = true      TextColor = 0       BackColor = 78748035
Border = true       BorderColor = 0     BorderStyle = styleraised!
Alignment = center! FillPattern = solid!
```

w_setcasos




```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:02:36

Window: w_setcasos
X = 1075              Y = 485              Width = 2108         Height = 1521
Visible = true        Enabled = true       TitleBar = true      ControlMenu = true
Border = true         WindowType = response!                    WindowState =
normal!
BackColor = 78748035

Script for: nt open  event
datastore      ds_tablas
long           ll_Rows, ll_Row
listviewitem   lvi_item


ds_tablas = CREATE datastore

ds_tablas.DataObject = 'ddw_tablas'
ds_tablas.Settransobject( SQLAny50 )
ll_Rows = ds_tablas.Retrieve()
For ll_Row = 1 to ll_Rows
  lvi_item.Label        = ds_tablas.GetItemString( ll_Row, "nombre" )
  lvi_item.PictureIndex = 1
  lvi_item.Data         = ds_tablas.GetItemNumber( ll_Row, "id" )
  lv_tablas.AddItem( lvi_item  )
Next
```

```
DESTROY ds_tablas
end event

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:02:36


End of Script



public function integer wf_tablas2casos (integer index, ref listviewitem item)
listviewitem   lvi_itemput
Long           ll_Row

lvi_itemput.Label        = item.Label
lvi_itemput.Data         = item.Data
lvi_itemput.PictureIndex = item.PictureIndex
lv_setcasos.AddItem( lvi_itemput )

ll_Row = dw_casosxtabla.Find( "id="+string(item.Data)+" and idc="+string(
dw_casos.get
Itemnumber( dw_casos.getrow(), "idc" ) ), 0 , dw_casosxtabla.RowCount() )
If ll_Row > 0 Then
  dw_casosxtabla.SetItem( ll_Row, "cdel", 1 )
Else
  ll_Row = dw_casosxtabla.InsertRow(0)
  dw_casosxtabla.SetItem( ll_Row, "id", long(item.Data) )
  dw_casosxtabla.SetItem( ll_Row, "idc", dw_casos.getItemnumber(
dw_casos.getrow(), "i
dc" ) )
  dw_casosxtabla.SetItem( ll_Row, "tablas_nombre", item.Label )
  dw_casosxtabla.SetItem( ll_Row, "cdel", 1 )
End if

Return 0
end function



public function integer wf_casos2tablas (integer index, ref listviewitem item)
long ll_row

listviewitem   lvi_itemput

lvi_itemput.Label        = item.Label
lvi_itemput.Data         = item.Data
lvi_itemput.PictureIndex = item.PictureIndex
lv_tablas.AddItem( lvi_itemput )

ll_Row = dw_casosxtabla.Find( "id="+string(item.Data)+" and idc="+string(
dw_casos.get
Itemnumber( dw_casos.getrow(), "idc" ) ), 0 , dw_casosxtabla.RowCount() )
If ll_Row > 0 Then
  dw_casosxtabla.SetItem( ll_Row, "cdel", 0 )
End if

Return 0
```

```
end function


CommandButton: cb_5
X = 997              Y = 1317            Width = 247         Height = 109
TabOrder = 21        Visible = true      Enabled = true      Text = "none"

Script for: nt clicked  event
long ll_Row
For ll_Row = dw_casosxtabla.RowCount() to 1 Step -1
  if dw_casosxtabla.GetItemNumber( ll_Row, "cdel" ) = 0 Then
    dw_casosxtabla.DeleteRow( ll_Row )
  end if
Next

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:02:36


dw_casosxtabla.Update( True, true )
end event

End of Script



DataWindow: dw_casosxtabla
X = 330              Y = 253             Width = 1358        Height = 285
TabOrder = 40        Visible = true      Enabled = true
DataObject = "dw_casosxtabla"            VScrollBar = true   Border = true
LiveScroll = true    BorderStyle = stylelowered!

Script for: nt constructor  event
Visible = False
SetTransobject(sqlany50)
retrieve()
end event

End of Script



ListView: lv_setcasos
X = 1139             Y = 569             Width = 892         Height = 813
TabOrder = 70        Visible = true      Enabled = true      Border = true
TextColor = 33554432                     BackColor = 1073741824
BorderStyle = stylelowered!              ButtonHeader = true ShowHeader = true
LabelWrap = true     Scrolling = true    HideSelection = true
SortType = ascending!                    View = listviewsmallicon!
LargePictureWidth = 32                   LargePictureHeight = 32
LargePictureMaskColor = 553648127        SmallPictureWidth = 16
SmallPictureHeight = 16                  SmallPictureMaskColor = 553648127
StatePictureWidth = 0                    StatePictureHeight = 0
StatePictureMaskColor = 536870912
LargePictureName = { "Layer!" }
SmallPictureName = { "Layer!" }
```

```
CommandButton: cb_4
X = 974              Y = 1005            Width = 119         Height = 93
TabOrder = 60        Visible = true      Enabled = true      Text = ">>"

Script for: nt clicked  event
Integer i
listviewitem lvi_itemget, lvi_itemput

For i = 1 To lv_tablas.TotalItems()
  lv_tablas.GetItem( i, lvi_itemget )
  lvi_itemput.Label        = lvi_itemget.Label
  lvi_itemput.Data         = lvi_itemget.Data
  lvi_itemput.PictureIndex = lvi_itemget.PictureIndex
  lv_setcasos.AddItem( lvi_itemput )
Next

lv_tablas.DeleteItems()
lv_tablas.Arrange()
lv_setcasos.Arrange()
end event

End of Script

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:02:36



CommandButton: cb_3
X = 974              Y = 1109            Width = 119         Height = 93
TabOrder = 20        Visible = true      Enabled = true      Text = "<<"

Script for: nt clicked  event
Integer i
listviewitem lvi_itemget, lvi_itemput

For i = 1 To lv_setcasos.TotalItems()
  lv_setcasos.GetItem( i, lvi_itemget )
  lvi_itemput.Label        = lvi_itemget.Label
  lvi_itemput.Data         = lvi_itemget.Data
  lvi_itemput.PictureIndex = lvi_itemget.PictureIndex
  lv_tablas.AddItem( lvi_itemput )
Next

lv_setcasos.DeleteItems()
lv_tablas.Arrange()
lv_setcasos.Arrange()
end event

End of Script



CommandButton: cb_2
X = 974              Y = 813             Width = 119         Height = 93
TabOrder = 50        Visible = true      Enabled = true      Text = "<"
```

```
Script for: nt clicked  event
Integer i
listviewitem lvi_itemget, lvi_itemput

For i = 1 To lv_setcasos.TotalItems()
  lv_setcasos.GetItem( i, lvi_itemget )
  If lvi_itemget.Selected Then
    wf_Casos2Tablas( i, lvi_itemget )
  End if
Next

For i = lv_setcasos.TotalItems() To 1 Step -1
  lv_setcasos.GetItem( i, lvi_itemget )
  If lvi_itemget.Selected Then
    lv_setcasos.DeleteItem( i )
  End if
Next

lv_tablas.Arrange()
lv_setcasos.Arrange()
end event

End of Script



CommandButton: cb_1
X = 974              Y = 705              Width = 119          Height = 93
TabOrder = 10        Visible = true       Enabled = true       Text = ">"

Script for: nt clicked  event
Integer i
listviewitem lvi_itemget, lvi_itemput

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:02:36


For i = 1 To lv_tablas.TotalItems()
  lv_tablas.GetItem( i, lvi_itemget )
  If lvi_itemget.Selected Then
    wf_Tablas2Casos( i, lvi_itemget )
  End if
Next

For i = lv_tablas.TotalItems() To 1 Step -1
  lv_tablas.GetItem( i, lvi_itemget )
  If lvi_itemget.Selected Then
    lv_tablas.DeleteItem( i )
  End if
Next

lv_tablas.Arrange()
lv_setcasos.Arrange()
end event

End of Script
```

```
DataWindow: dw_casos
X = 28              Y = 29              Width = 1994        Height = 517
TabOrder = 30       Visible = true      Enabled = true      DataObject =
"dw_casos"
VScrollBar = true   Border = true       LiveScroll = true
BorderStyle = stylelowered!

Script for: nt constructor  event
This.Settransobject( SQLAny50 )
This.Retrieve()
end event

End of Script



Script for: nt rowfocuschanged  event
long           ll_Row, ll_Index
listviewitem   lvi_item
String         ls_labelsearch

If currentrow >= 1 And currentrow <= RowCount() Then
  cb_3.Event clicked()
  dw_casosxtabla.SetFilter( 'idc='+String( This.GetitemNumber( currentrow, "idc"
) ) )

  dw_casosxtabla.Filter()

  For ll_Row = 1 To dw_casosxtabla.RowCount()
    ls_labelsearch = dw_casosxtabla.GetItemString( ll_Row, "tablas_nombre" )
    If dw_casosxtabla.GetItemNumber( ll_Row, "cdel" ) = 1 Then
      ll_Index = lv_tablas.FindItem( 0, ls_labelsearch, FALSE, FALSE )
      If ll_Index = -1 Then
      Else
        lv_tablas.getItem( ll_index, lvi_item )
        lvi_item.Selected = True
        lv_tablas.SetItem( ll_index, lvi_item )
      End if
    End if
  Next
  cb_1.Event clicked()

Else
-

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:02:36

  dw_casosxtabla.SetFilter( 'idc= -999' )
  dw_casosxtabla.Filter()
  cb_3.Event clicked()
End if
end event

End of Script
```

```
ListView: lv_tablas
X = 37                Y = 569              Width = 892          Height = 813
TabOrder = 80         Visible = true       Enabled = true       Border = true
TextColor = 33554432                       BackColor = 1073741824
BorderStyle = stylelowered!                ButtonHeader = true ExtendedSelect =
true
ShowHeader = true     LabelWrap = true     Scrolling = true     HideSelection = true
SortType = ascending!                      View = listviewsmallicon!
LargePictureWidth = 32                     LargePictureHeight = 32
LargePictureMaskColor = 553648127          SmallPictureWidth = 16
SmallPictureHeight = 16                    SmallPictureMaskColor = 553648127
StatePictureWidth = 0                      StatePictureHeight = 0
StatePictureMaskColor = 536870912
LargePictureName = { "Layer!" }
SmallPictureName = { "Layer!" }
```

## w_statusbar

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:03:41

Window: w_statusbar
X = 1345          Y = 753           Width = 2341        Height = 269
Visible = true    Enabled = true    WindowType = popup! WindowState =
maximized!
BackColor = 16777215

Instance Variables
Environment ienv_object
Integer     ii_winmaxwidth
long     il_warningcolor=255          // Red
long     il_buttonface
long     ii_borderheight
long     ii_borderwidth
integer     ii_secondstorefresh = 60    // one minute
boolean     ib_win95=False
window      iw_ParentWindow

End of Instance Variables


Script for: nt open  event
This.PostEvent("postopen")
end event

End of Script



Script for: nt postopen  event
//uo_1.of_SetFrame( iw_This )
This.Y = iw_parentwindow.Height - This.Height

Return 1
end event

End of Script



public function integer of_setpos ()
//////////////////////////////////////////////////////////////////////////////
//
// Function:   of_SetPosition
//

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:03:41

// Access:     public
```

```
//
// Arguments:  None
//
// Returns:    Integer
// 1 if it succeeds and -1 if an error occurs.
//
// Description:
// Moves this StatusBar window so it is positioned over the lower right
// hand portion of the Microhelp bar.
//
////////////////////////////////////////////////////////////////////////////////
//
// Revision History
//
// Version
// 5.0   Initial version
//
////////////////////////////////////////////////////////////////////////////////
//
// Copyright © 1996 Powersoft Corporation.  All Rights Reserved.
// Any distribution of the PowerBuilder Foundation Classes (PFC)
// source code by other than Powersoft is prohibited.
//
////////////////////////////////////////////////////////////////////////////////

long     ll_microhelp_ypos, ll_frame_xpos, ll_desiredstatubar_xpos
integer  li_xpos_extra=0, li_ypos_extra=0

// Check window reference.
If Not IsValid(iw_parentwindow) Then
   Return -1
End If

if iw_parentwindow.windowstate = minimized! then
   return 1
end if

// Windows 95 has a bigger gap on the hand right side.
If ib_win95 Then
   li_xpos_extra = 50
   li_ypos_extra = 1
End If

// The Y Position of the Status Bar is the Bottom of the Frame Window
// minus the MicroHelpHeight minus the MicroHelpBorderHeight.
ll_microhelp_ypos = (iw_parentwindow.y + iw_parentwindow.height + li_ypos_extra)
- (iw
_parentwindow.mdi_1.microhelpheight + ii_borderheight)

// The desired X Position of the Status Bar is the Frame Right End minus
// the StatusBar window.  Also subtract the extra spacing on win95.
ll_desiredstatubar_xpos = iw_parentwindow.x + iw_parentwindow.workspacewidth() +
ii_bo
rderwidth - (ii_winmaxwidth + 12 + li_xpos_extra)

// The Frame X Position.
ll_frame_xpos = iw_parentwindow.x  + (2*ii_borderwidth) + 16
```

```
if ll_desiredstatubar_xpos < ll_frame_xpos then
   // Status Bar would extend to the left of the frame.
   //this.move(ll_frame_xpos , ll_microhelp_ypos)
   this.move(1 , ll_microhelp_ypos)
   // Make the StatuBar the width of the frame.
   //this.width = workspacewidth(iw_parentwindow) - (20 + li_xpos_extra)
   this.width = iw_parentwindow.workspacewidth() - (20 + li_xpos_extra)

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:03:41

else
   // Normal as large as defined Status Bar window.
   //this.move(ll_desiredstatubar_xpos , ll_microhelp_ypos)
   this.move(1 , ll_microhelp_ypos)
   this.width  = ii_winmaxwidth
this.height = (iw_parentwindow.mdi_1.microhelpheight + ii_borderheight)

end if

Return 1*/



//of_GetEnvironment()
//of_SetPos()

/*
iw_This = This
uo_1.of_SetFrame( iw_This )
iw_This = w_statusbar

This.Y = iw_parentwindow.Height - This.Height
*/
Return 1
end function



private subroutine of_getenvironment ()
////////////////////////////////////////////////////////////////////////////////
//
// Function:  of_GetSystemSettings
//

// Access:  protected
//
// Arguments : None
//
// Returns:  Integer
// 1 if it succeeds and -1 if an error occurs.
//
// Description:
// Get the needed settings from the registry or win.ini
//
////////////////////////////////////////////////////////////////////////////////
//
// Revision History
```

```
//
// Version
// 5.0   Initial version
//
//////////////////////////////////////////////////////////////////////////////
//
// Copyright © 1996 Powersoft Corporation.  All Rights Reserved.
// Any distribution of the PowerBuilder Foundation Classes (PFC)
// source code by other than Powersoft is prohibited.
//
//////////////////////////////////////////////////////////////////////////////

/*
iw_parentwindow = w_tmp
ii_winmaxwidth  = iw_parentwindow.Width

string ls_temp
integer li_border

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:03:41


If ienv_object.Win16 Then
   // win.ini.
   ls_temp = ProfileString("win.ini", "colors", "buttonface", "192 192 192")
   //il_buttonface = of_Color(ls_temp)

   li_border =    integer(ProfileString("win.ini", "windows", "borderwidth",
"2"))
   li_border *= 15

   ii_borderheight = unitstopixels(li_border,yunitstopixels!) - 16
Else
   If ienv_object.OSType = WindowsNT! Then
      // NT registry.
      registryget( 'hkey_current_user\control panel\colors', 'buttonface'
,ls_temp)
      //il_buttonface = of_Color(ls_temp)

      registryget('hkey_current_user\control
panel\desktop','borderwidth',ls_temp)
      li_border = abs(integer(ls_temp)) * 15

      ii_borderheight = unitstopixels(li_border, yunitstopixels!) - 16
   Else
      // win95 registry.
      registryget( 'hkey_current_user\control panel\colors', 'buttonface'
,ls_temp)
      //il_buttonface = of_Color(ls_temp)

registryget('hkey_current_user\control panel\desktop\windowmetrics','borderwidth
',ls_temp)
      li_border = abs(integer(ls_temp))

      ii_borderheight = unitstopixels(li_border,yunitstopixels!)
   End If
End If
```

```
ii_borderwidth = unitstopixels(li_border,xunitstopixels!)*/
end subroutine


public function integer of_setparent (ref window aw_parent)
iw_parentwindow = aw_parent
of_SetPos()
Return 1
end function


User Object: uo_1
X = 1                Y = 1                Width = 517          Height = 97
TabOrder = 1         Visible = true       Enabled = true       Border = true
BackColor = 78748035                      ObjectType = customvisual!

Script for: nt constructor  event
call super::constructor;//Parent.Height = This.Height


end event

End of Script
```

## w_tablas

```
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:04:30

Window: w_tablas
X = 833              Y = 361             Width = 1409        Height = 1437
Visible = true       Enabled = true      TitleBar = true
Title = "Definición de Store Procedure" ControlMenu = true  Border = true
WindowType = main!  WindowState = normal!                   BackColor = 79741120

Instance Variables
DataStore   sp_pkeys

End of Instance Variables


Script for: nt open  event
string ls_name
integer i,j
DataStore n_ds_tablasSQLServer


n_ds_tablasSQLServer=CREATE DataStore

n_ds_tablasSQLServer.DataObject = 'dw_sysobjects'
n_ds_tablasSQLServer.SetTransObject( SQLServer )
n_ds_tablasSQLServer.Retrieve()

DELETE from tmptablas USING SQLAny50;
COMMIT;

j=n_ds_tablasSQLServer.RowCount()
For i=1 to n_ds_tablasSQLServer.RowCount()
  ls_name= n_ds_tablasSQLServer.Object.name[i]

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:04:30

  INSERT INTO tmptablas (tmptablas.nombre) values (:ls_name) USING SQLAny50;
Next
COMMIT;

DESTROY n_ds_tablasSQLServer

//DELETE FROM tmptablas WHERE EXISTS (SELECT nombre FROM tablas WHERE
tablas.nombre=tm
ptablas.nombre) USING SQLAny50;
//COMMIT;

// INSERT INTO tablas (nombre) SELECT nombre FROM tmptablas USING SQLAny50;
INSERT INTO tablas (nombre) SELECT nombre FROM tmptablas WHERE NOT EXISTS
(SELECT nomb
re FROM tablas WHERE tablas.nombre=tmptablas.nombre) USING SQLAny50;
COMMIT;
```

```
DELETE FROM tablas WHERE NOT EXISTS (SELECT nombre FROM tmptablas WHERE
tablas.nombre=
tmptablas.nombre) USING SQLAny50;
COMMIT;

cdw_tablas.SetTransObject( SQLAny50 )

cdw_tablas.Retrieve()


end event

End of Script



public function boolean fn_spkeys (string asp_keys)
long ll_found
string ls_string

ls_string = "column_name = '" + asp_keys + "'"

ll_found = sp_pkeys.Find(ls_string,1, sp_pkeys.RowCount( ))

if ll_found >0 then
   return true
else
   return false
end if
end function



CommandButton: cb_1
X = 636             Y = 1193            Width = 119         Height = 109
TabOrder = 20       Visible = true      Enabled = true      Text = "No"

Script for: nt clicked  event
integer i

For i=1 to cdw_tablas.RowCount()
  If cdw_tablas.Object.replica[i] Then
    cdw_tablas.Object.replica[i]=0
  Else
    cdw_tablas.Object.replica[i]=-1
  End if
Next

cdw_tablas.Update()
end event

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:04:30


End of Script
```

```
CommandButton: cb_crear
X = 988              Y = 1193            Width = 325          Height = 109
TabOrder = 30        Visible = true      Enabled = true       Text = "&Crear
Script"

Script for: nt clicked  event
integer li_FileSub,i,j,li_FileDef, li_valor
string ls_string,  ls_sql_where, ls_null
String   ls_sql_col[], ls_sql_val[], ls_sql_upd[], ls_Empty[]
String ls_TableName, ls_strings, ls_stringr

setpointer(HourGlass!)

ole_1.visible = true
ole_1.object.value = 0

DataStore sp_columns
DataStore syscolumns

sp_columns = CREATE DataStore
sp_columns.DataObject = 'dw_sp_columns'
syscolumns = CREATE DataStore
syscolumns.DataObject = 'dw_syscolumns'
sp_pkeys   = CREATE DataStore
sp_pkeys.DataObject = 'dw_sp_keys'

SQLServer.Autocommit = True
SQLAny50.Autocommit  = True

sp_columns.SetTransObject( SQLServer )
syscolumns.SetTransObject( SQLServer )
sp_pkeys.SetTransObject( SQLServer )

li_FileSub=FileOpen(
"c:\temp\procsub.sql",StreamMode!,Write!,LockWrite!,Replace!)
li_FileDef=FileOpen(
"c:\temp\procdef.sql",StreamMode!,Write!,LockWrite!,Replace!)


ls_StringS =
"/*****************************************************************************
**/~r~n" + &
            "/* Crea el script de los (store procedure) para las tablas
seleccionadas
 */~r~n" + &
            "/* Estas instrucciones deberán ejecutarse en el Servidor Replicado
 */~r~n" + &

"/*****************************************************************************
**/~r~n~r~n"


ls_StringR =
"/*****************************************************************************
**/~r~n" - &
            "/* Crea el script de los (store procedure) para las tablas
seleccionadas
```

```
 */~r~n" + &
            "/* Estas instrucciones deberán ejecutarse en el Servidor Primario
 */~r~n" + &

"/*************************************************************************
**/~r~n~r~n"


FileWrite( li_FileSub, ls_StringS )
FileWrite( li_FileDef, ls_StringR )

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:04:30

For i=1 to cdw_tablas.RowCount()

  ls_sql_col = ls_Empty
  ls_sql_val = ls_Empty
  ls_sql_upd = ls_Empty

  If cdw_tablas.Object.replica[i] Then
    ls_TableName = cdw_tablas.Object.nombre[i]
    ls_StringS   = "CREATE PROCEDURE SUBsp_"+ls_TableName+" @tipo
char(1),"+"~r~n"
 ls_StringS  += Space( 22+Len( ls_Tablename ) )+"  @asa_c_aerodest
char(3),"+"~r~n"

    ls_stringR   = "CREATE PROCEDURE DEFsp_"+ls_TableName+" @tipo
char(1),"+"~r~n"
 ls_stringR  += Space( 22+Len( ls_Tablename ) )+"  @asa_c_aerodest
char(3),"+"~r~n"


    ls_sql_where = ""

    // primarias
    sp_pkeys.Retrieve( cdw_tablas.Object.nombre[i],"dbo","ASA" )
    for j = 1 to sp_pkeys.RowCount()
      If j < sp_pkeys.RowCount() Then
         ls_sql_where = ls_sql_where + sp_pkeys.Object.column_name[j] + " = @" &
                    + sp_pkeys.Object.column_name[j] + " and "
      else
         ls_sql_where = ls_sql_where + sp_pkeys.Object.column_name[j] + " = @" &
                  + sp_pkeys.Object.column_name[j]
      end if
    next

    // columnas
    syscolumns.Retrieve( cdw_tablas.Object.nombre[i] )

    For j=1 to syscolumns.RowCount()
sp_columns.Retrieve( cdw_tablas.Object.nombre[i],"dbo","ASA", syscolumns.Object.
syscolumns_name[j] )

      if fn_spkeys(syscolumns.Object.syscolumns_name[j]) then
         ls_null = ""
      else
         ls_null = " = null"
```

```
      end if

      syscolumns.Object.syscolumns_name[j]
      If j < syscolumns.RowCount() Then

ls_StringS += Space( 22+Len( ls_Tablename ) )+"  @"+syscolumns.Object.syscolu
mns_name[j]+"  "+sp_columns.Object.type_name[1]
ls_StringR += Space( 22+Len( ls_Tablename ) )+"  @"+syscolumns.Object.syscolu
mns_name[j]+"  "+sp_columns.Object.type_name[1]
         ls_sql_col[j] = syscolumns.Object.syscolumns_name[j] + ","
         ls_sql_val[j] = "@"+syscolumns.Object.syscolumns_name[j] + ","

         if not fn_spkeys( syscolumns.Object.syscolumns_name[j] ) then
            ls_sql_upd[j] = syscolumns.Object.syscolumns_name[j] + " = @" + &
                            syscolumns.Object.syscolumns_name[j] + ","
         end if

If sp_columns.Object.type_name[1]="char" or sp_columns.Object.type_name[1]="v
archar" Then
  ls_StringS += "("+string( sp_columns.Object.length[1] )+")" + ls_null + ",~
r~n"
  ls_StringR += "("+string( sp_columns.Object.length[1] )+")" + ls_null + ",~
r~n"
         ElseIf sp_columns.Object.type_name[1]="numeric" Then
_
Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:04:30

  ls_StringS += "("+string( sp_columns.Object.precision[1] )+")" + ls_null +
",~r~n"
   ls_StringR += "("+string( sp_columns.Object.precision[1] )+")" + ls_null +
 ",~r~n"
         Else
           ls_StringS += ls_null + ",~r~n"
           ls_StringR += ls_null + ",~r~n"
         End if
      Else
ls_StringS += Space( 22+Len( ls_Tablename ) )+"  @"+syscolumns.Object.syscolu
mns_name[j]+"  "+sp_columns.Object.type_name[1]
ls_StringR += Space( 22+Len( ls_Tablename ) )+"  @"+syscolumns.Object.syscolu
mns_name[j]+"  "+sp_columns.Object.type_name[1]
         ls_sql_col[j] = syscolumns.Object.syscolumns_name[j]
         ls_sql_val[j] = "@"+syscolumns.Object.syscolumns_name[j]

        If not fn_spkeys(syscolumns.Object.syscolumns_name[j]) then
          ls_sql_upd[j] = syscolumns.Object.syscolumns_name[j] + " = @" + &
                          syscolumns.Object.syscolumns_name[j]
        End if

  If sp_columns.Object.type_name[1]="char" or sp_columns.Object.type_name[1]="va
rchar" Then
 ls_StringS += "("+string( sp_columns.Object.length[1] )+") "+ls_null+" AS~r~
n"
 ls_StringR += "("+string( sp_columns.Object.length[1] )+") "+ls_null+" AS~r~
n"
       ElseIf sp_columns.Object.type_name[1]="numeric" Then
 ls_StringS += "("+string( sp_columns.Object.precision[1] )+") "+ls_null+" AS
~r~n"
```

```
 ls_StringR += "("+string( sp_columns.Object.precision[1] )+") "+ls_null+" AS
~r~n"
        Else
          ls_StringS += ls_null + " AS ~r~n"
          ls_StringR += ls_null + " AS ~r~n"
        End if
      End if
   next

   ls_StringS += "BEGIN~r~n"
   ls_StringR += "BEGIN~r~n"
   // insert
   ls_StringS += "  if @tipo = 'I'~r~n"
   ls_StringS += "  begin~r~n"
   ls_StringS += "    INSERT INTO " + ls_Tablename+" ~r~n"
   ls_StringS += "               ( " + ls_sql_col[1]+"~r~n"
   for j = 2 to upperbound(ls_sql_col) - 1
     ls_StringS += "                 " + ls_sql_col[j]+"~r~n"
   next
   ls_StringS += "                 " + ls_sql_col[upperbound(ls_sql_col)]+"
)~r~n"
   ls_StringS += "       VALUES ( " + ls_sql_val[1]+"~r~n"
   for j = 2 to upperbound(ls_sql_val) - 1
     ls_StringS += "                 " + ls_sql_val[j]+"~r~n"
   next
   ls_StringS += "                 " + ls_sql_val[upperbound(ls_sql_val)]+"
)~r~n"
   ls_StringS += "  end~r~n~r~n"

    // delete
   ls_StringS += "  if @tipo = 'D'~r~n"
   ls_StringS += "  begin~r~n"
   ls_StringS += "    DELETE FROM " + ls_TableName +"~r~n"
   ls_StringS += "          WHERE " + ls_sql_where +"~r~n"
   ls_StringS += "  end~r~n~r~n"

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:04:30

   // update
   ls_StringS += "  if @tipo = 'U'~r~n"
   ls_StringS += "  begin~r~n"
   ls_StringS += "    UPDATE " + ls_TableName+"~r~n"
   ls_StringS += "       SET "
   For j = 1 to upperbound(ls_sql_upd)
     If not isnull(ls_sql_upd[j]) and ls_sql_upd[j] <> "" then
       ls_StringS += ls_sql_upd[j]+"~r~n             "
     End if
   Next
   ls_StringS  = RightTrim( ls_StringS )
   ls_StringS += "     WHERE " + ls_sql_where +"~r~n"
   ls_StringS += "  end~r~n~r~n"
   ls_StringS += "  RETURN ~r~n"
   ls_StringR += "  RETURN~r~n"
   ls_StringS += "END~r~ngo~r~n~r~n"
   ls_StringR += "END~r~ngo~r~n~r~n"
   FileWrite( li_FileDef, ls_StringR )
   FileWrite( li_FileSub, ls_StringS )
```

```
   End if

   li_valor = Int( ( i / cdw_tablas.RowCount()    ) * 100 )
   ole_1.object.value = li_valor

Next

FileClose( li_FileSub )
FileClose( li_FileDef )

DESTROY sp_columns
DESTROY syscolumns
DESTROY sp_pkeys

SQLServer.Autocommit=False
SQLAny50.Autocommit=False

ole_1.visible = false
setpointer(Arrow!)

MessageBox("","Proceso Terminado")

end event

End of Script



CommandButton: cb_guardar
X = 78              Y = 1193            Width = 325         Height = 109
TabOrder = 40       Visible = true      Enabled = true      Text = "&Guardar"

Script for: nt clicked  event
cdw_tablas.Update()

end event

End of Script

Library: D:\Mis documentos\Personal\PowerBuilder\RSSG\SCRIPT.PBL
Date: 2/04/01    Time: 13:04:30



DataWindow: cdw_tablas
X = 37              Y = 37              Width = 1331        Height = 1085
TabOrder = 10       Visible = true      Enabled = true      DataObject =
"dw_tablas"
VScrollBar = true   Border = true       LiveScroll = true
BorderStyle = stylelowered!


OleCustomControl: ole_1
X = 37              Y = 1193            Width = 1331        Height = 109
TabOrder = 21       Enabled = true      BackColor = 79741120
BorderStyle = stylebox!                 FocusRectangle = true
TextColor = 0       Alignment = left!
```

# APENDICE B. REPOSITORIO.

## Script para creación del repositorio

```
--
-- This command file reloads a database that was unloaded using "dbunload".
--
-- (version:  6.0.3.2747)
--

SET OPTION Statistics          = 3
go
SET OPTION Date_order          = 'YMD'
go
SET OPTION describe_java_format= 'binary'
go


------------------------------------------------
--   Create userids and grant user permissions
------------------------------------------------

GRANT CONNECT TO "DBA" IDENTIFIED BY "SQL"
go
GRANT RESOURCE, DBA, SCHEDULE TO "DBA"
go
GRANT CONNECT TO "dbo" IDENTIFIED BY ENCRYPTED
'\x24\x11\x0B\x91\x1F\x1B\xDA\x01\x70\xAE\x08\x63\x84\x8C\x28\x9E\xF6\x71\x95\x6
9\x4C\x0A\x33\xDA\x58\x40\x83\x01\x37\x62\x27\x02\x27\x53\x0F\x13'
go
GRANT GROUP TO "dbo"
go
GRANT RESOURCE, DBA TO "dbo"
go
GRANT CONNECT TO "moises" IDENTIFIED BY ENCRYPTED
'\xA5\x31\x4C\x49\x8C\x5C\x2C\x27\x45\x18\x52\x62\x6A\x9B\xA3\x19\xA2\xB9\xD5\x7
C\x72\xF9\x6D\x27\xA8\x5B\xFE\x10\xCD\xB8\xBC\x5C\xC9\x08\x09\x66'
go
commit work
go


------------------------------------------------
--   Create user-defined types
------------------------------------------------

setuser "DBA"
go
CREATE DOMAIN T_money numeric(20,4)
go
CREATE DOMAIN T_smallmoney numeric(10,4)
go
CREATE DOMAIN T_sysname varchar(30) NOT NULL
go
CREATE DOMAIN T_text long varchar
go
```

```
CREATE DOMAIN T_image long binary
go
CREATE DOMAIN T_datetime timestamp
go
CREATE DOMAIN T_smalldatetime timestamp
go
CREATE DOMAIN T_bit tinyint
go
setuser "DBA"
go

commit work
go

------------------------------------------------
--   Install user-defined classes
------------------------------------------------

commit work
go

------------------------------------------------
--   Grant user memberships
------------------------------------------------

commit work
go


------------------------------------------------
--   Create external servers
------------------------------------------------

commit work
go


------------------------------------------------
--   Create tables
------------------------------------------------

CREATE TABLE "DBA"."tmptablas"
(
      "nombre"                varchar(30) NOT NULL
)
go
CREATE TABLE "DBA"."tablas"
(
      "id"                    integer NOT NULL DEFAULT autoincrement,
      "nombre"                varchar(30) NOT NULL,
      "replica"               smallint NOT NULL DEFAULT 0,
      "status"                char(1) NULL
)
go
CREATE TABLE "DBA"."casos"
(
```

```
      "idc"                    integer NOT NULL DEFAULT autoincrement,
      "nombre"                 varchar(60) NOT NULL,
      "descripcion"            varchar(255) NULL
)
go
CREATE TABLE "DBA"."casosxtabla"
(
      "id"                     integer NULL,
      "idc"                    integer NULL
)
go
CREATE TABLE "DBA"."pbcattbl"
(
      "pbt_tnam"               char(129) NOT NULL,
      "pbt_tid"                integer NULL,
      "pbt_ownr"               char(129) NOT NULL,
      "pbd_fhgt"               smallint NULL,
      "pbd_fwgt"               smallint NULL,
      "pbd_fitl"               char(1) NULL,
      "pbd_funl"               char(1) NULL,
      "pbd_fchr"               smallint NULL,
      "pbd_fptc"               smallint NULL,
      "pbd_ffce"               char(18) NULL,
      "pbh_fhgt"               smallint NULL,
      "pbh_fwgt"               smallint NULL,
      "pbh_fitl"               char(1) NULL,
      "pbh_funl"               char(1) NULL,
      "pbh_fchr"               smallint NULL,
      "pbh_fptc"               smallint NULL,
      "pbh_ffce"               char(18) NULL,
      "pbl_fhgt"               smallint NULL,
      "pbl_fwgt"               smallint NULL,
      "pbl_fitl"               char(1) NULL,
      "pbl_funl"               char(1) NULL,
      "pbl_fchr"               smallint NULL,
      "pbl_fptc"               smallint NULL,
      "pbl_ffce"               char(18) NULL,
      "pbt_cmnt"               char(254) NULL
)
go
setuser "DBA"
go
GRANT SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES ON "DBA"."pbcattbl" TO
"PUBLIC"
go
setuser "DBA"
go
CREATE TABLE "DBA"."pbcatcol"
(
      "pbc_tnam"               char(129) NOT NULL,
      "pbc_tid"                integer NULL,
      "pbc_ownr"               char(129) NOT NULL,
      "pbc_cnam"               char(129) NOT NULL,
      "pbc_cid"                smallint NULL,
      "pbc_labl"               char(254) NULL,
      "pbc_lpos"               smallint NULL,
      "pbc_hdr"                char(254) NULL,
      "pbc_hpos"               smallint NULL,
```

```
	"pbc_jtfy"		smallint NULL,
	"pbc_mask"		char(31) NULL,
	"pbc_case"		smallint NULL,
	"pbc_hght"		smallint NULL,
	"pbc_wdth"		smallint NULL,
	"pbc_ptrn"		char(31) NULL,
	"pbc_bmap"		char(1) NULL,
	"pbc_init"		char(254) NULL,
	"pbc_cmnt"		char(254) NULL,
	"pbc_edit"		char(31) NULL,
	"pbc_tag"		char(254) NULL
)
go
setuser "DBA"
go
GRANT SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES ON "DBA"."pbcatcol" TO
"PUBLIC"
go
setuser "DBA"
go
CREATE TABLE "DBA"."pbcatfmt"
(
	"pbf_name"		char(30) NOT NULL,
	"pbf_frmt"		char(254) NULL,
	"pbf_type"		smallint NULL,
	"pbf_cntr"		integer NULL
)
go
setuser "DBA"
go
GRANT SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES ON "DBA"."pbcatfmt" TO
"PUBLIC"
go
setuser "DBA"
go
CREATE TABLE "DBA"."pbcatvld"
(
	"pbv_name"		char(30) NOT NULL,
	"pbv_vald"		char(254) NULL,
	"pbv_type"		smallint NULL,
	"pbv_cntr"		integer NULL,
	"pbv_msg"		char(254) NULL
)
go
setuser "DBA"
go
GRANT SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES ON "DBA"."pbcatvld" TO
"PUBLIC"
go
setuser "DBA"
go
CREATE TABLE "DBA"."pbcatedt"
(
	"pbe_name"		char(30) NOT NULL,
	"pbe_edit"		char(254) NULL,
	"pbe_type"		smallint NULL,
	"pbe_cntr"		integer NULL,
	"pbe_seqn"		smallint NOT NULL,
```

```
      "pbe_flag"              integer NULL,
      "pbe_work"              char(32) NULL
)
go
setuser "DBA"
go
GRANT SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES ON "DBA"."pbcatedt" TO
"PUBLIC"
go
setuser "DBA"
go
CREATE TABLE "DBA"."configuracion"
(
      "idcfg"                 varchar(30) NOT NULL,
      "value"                 varchar(30) NOT NULL,
      "descripcion"           varchar(255) NULL
)
go
CREATE TABLE "DBA"."scripts"
(
      "idscr"                 varchar(30) NOT NULL,
      "descripcion"           varchar(255) NULL,
      "script"                T_text NULL
)
go
CREATE TABLE "DBA"."definiciones"
(
      "iddef"                 integer NOT NULL DEFAULT autoincrement,
      "idc"                   integer NOT NULL,
      "filename"              varchar(255) NOT NULL
)
go
CREATE TABLE "DBA"."defscripts"
(
      "idds"                  integer NOT NULL DEFAULT autoincrement,
      "iddef"                 integer NOT NULL,
      "idscr"                 varchar(30) NULL,
      "seccion"               smallint NOT NULL DEFAULT 1,
      "ord"                   smallint NOT NULL DEFAULT 0,
      "script"                long varchar NULL
)
go
CREATE TABLE "DBA"."pbcattbl"
(
      "pbt_tnam"              char(129) NOT NULL,
      "pbt_tid"               integer NULL,
      "pbt_ownr"              char(129) NOT NULL,
      "pbd_fhgt"              smallint NULL,
      "pbd_fwgt"              smallint NULL,
      "pbd_fitl"              char(1) NULL,
      "pbd_funl"              char(1) NULL,
      "pbd_fchr"              smallint NULL,
      "pbd_fptc"              smallint NULL,
      "pbd_ffce"              char(18) NULL,
      "pbh_fhgt"              smallint NULL,
      "pbh_fwgt"              smallint NULL,
      "pbh_fitl"              char(1) NULL,
      "pbh_funl"              char(1) NULL,
```

```
	"pbh_fchr"		smallint NULL,
	"pbh_fptc"		smallint NULL,
	"pbh_ffce"		char(18) NULL,
	"pbl_fhgt"		smallint NULL,
	"pbl_fwgt"		smallint NULL,
	"pbl_fitl"		char(1) NULL,
	"pbl_funl"		char(1) NULL,
	"pbl_fchr"		smallint NULL,
	"pbl_fptc"		smallint NULL,
	"pbl_ffce"		char(18) NULL,
	"pbt_cmnt"		char(254) NULL
)
go
GRANT SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES ON "DBA"."pbcattbl" TO
"PUBLIC"
go
CREATE TABLE "DBA"."pbcatcol"
(
	"pbc_tnam"		char(129) NOT NULL,
	"pbc_tid"		integer NULL,
	"pbc_ownr"		char(129) NOT NULL,
	"pbc_cnam"		char(129) NOT NULL,
	"pbc_cid"		smallint NULL,
	"pbc_labl"		char(254) NULL,
	"pbc_lpos"		smallint NULL,
	"pbc_hdr"		char(254) NULL,
	"pbc_hpos"		smallint NULL,
	"pbc_jtfy"		smallint NULL,
	"pbc_mask"		char(31) NULL,
	"pbc_case"		smallint NULL,
	"pbc_hght"		smallint NULL,
	"pbc_wdth"		smallint NULL,
	"pbc_ptrn"		char(31) NULL,
	"pbc_bmap"		char(1) NULL,
	"pbc_init"		char(254) NULL,
	"pbc_cmnt"		char(254) NULL,
	"pbc_edit"		char(31) NULL,
	"pbc_tag"		char(254) NULL
)
go
GRANT SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES ON "DBA"."pbcatcol" TO
"PUBLIC"
go
CREATE TABLE "DBA"."pbcatfmt"
(
	"pbf_name"		char(30) NOT NULL,
	"pbf_frmt"		char(254) NULL,
	"pbf_type"		smallint NULL,
	"pbf_cntr"		integer NULL
)
go
GRANT SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES ON "DBA"."pbcatfmt" TO
"PUBLIC"
go
CREATE TABLE "DBA"."pbcatvld"
(
	"pbv_name"		char(30) NOT NULL,
	"pbv_vald"		char(254) NULL,
```

```
      "pbv_type"              smallint NULL,
      "pbv_cntr"              integer NULL,
      "pbv_msg"               char(254) NULL
)
go
GRANT SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES ON "DBA"."pbcatvld" TO
"PUBLIC"
go
CREATE TABLE "DBA"."pbcatedt"
(
      "pbe_name"              char(30) NOT NULL,
      "pbe_edit"              char(254) NULL,
      "pbe_type"              smallint NULL,
      "pbe_cntr"              integer NULL,
      "pbe_seqn"              smallint NOT NULL,
      "pbe_flag"              integer NULL,
      "pbe_work"              char(32) NULL
)
go
GRANT SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES ON "DBA"."pbcatedt" TO
"PUBLIC"
go
commit work
go


------------------------------------------------
--   Reload data
------------------------------------------------

commit work
go


------------------------------------------------
--   Add foreign key definitions
------------------------------------------------


CREATE UNIQUE INDEX "tmptablas_pk" ON "DBA"."tmptablas"
(
      "nombre" ASC
)
go

CREATE UNIQUE INDEX "tablas_pk" ON "DBA"."tablas"
(
      "id" ASC
)
go

CREATE UNIQUE INDEX "tablas_ak" ON "DBA"."tablas"
(
      "nombre" ASC
)
go

CREATE UNIQUE INDEX "casos_pk" ON "DBA"."casos"
```

```
(
      "idc" ASC
)
go

CREATE INDEX "casosxtabla_fk1" ON "DBA"."casosxtabla"
(
      "id" ASC
)
go

CREATE INDEX "casosxtabla_fk2" ON "DBA"."casosxtabla"
(
      "idc" ASC
)
go

CREATE UNIQUE INDEX "pbcatt_x" ON "DBA"."pbcattbl"
(
      "pbt_tnam" ASC,
      "pbt_ownr" ASC
)
go

CREATE UNIQUE INDEX "pbcatc_x" ON "DBA"."pbcatcol"
(
      "pbc_tnam" ASC,
      "pbc_ownr" ASC,
      "pbc_cnam" ASC
)
go

CREATE UNIQUE INDEX "pbcatf_x" ON "DBA"."pbcatfmt"
(
      "pbf_name" ASC
)
go

CREATE UNIQUE INDEX "pbcatv_x" ON "DBA"."pbcatvld"
(
      "pbv_name" ASC
)
go

CREATE UNIQUE INDEX "pbcate_x" ON "DBA"."pbcatedt"
(
      "pbe_name" ASC,
      "pbe_seqn" ASC
)
go

CREATE UNIQUE INDEX "configuracion_pk" ON "DBA"."configuracion"
(
      "idcfg" ASC
)
go

CREATE UNIQUE INDEX "scripts_pk" ON "DBA"."scripts"
```

```
(
      "idscr" ASC
)
go

CREATE UNIQUE INDEX "definiciones_pk" ON "DBA"."definiciones"
(
      "iddef" ASC
)
go

CREATE INDEX "definiciones_fk1" ON "DBA"."definiciones"
(
      "idc" ASC
)
go

CREATE UNIQUE INDEX "defscripts_pk" ON "DBA"."defscripts"
(
      "idds" ASC
)
go

CREATE INDEX "defscripts_fk1" ON "DBA"."defscripts"
(
      "iddef" ASC
)
go

CREATE INDEX "defscripts_fk2" ON "DBA"."defscripts"
(
      "idscr" ASC
)
go

CREATE UNIQUE INDEX "pbcatt_x" ON "DBA"."pbcattbl"
(
      "pbt_tnam" ASC,
      "pbt_ownr" ASC
)
go

CREATE UNIQUE INDEX "pbcatc_x" ON "DBA"."pbcatcol"
(
      "pbc_tnam" ASC,
      "pbc_ownr" ASC,
      "pbc_cnam" ASC
)
go

CREATE UNIQUE INDEX "pbcatf_x" ON "DBA"."pbcatfmt"
(
      "pbf_name" ASC
)
go

CREATE UNIQUE INDEX "pbcatv_x" ON "DBA"."pbcatvld"
(
```

```
      "pbv_name" ASC
)
go

CREATE UNIQUE INDEX "pbcate_x" ON "DBA"."pbcatedt"
(
      "pbe_name" ASC,
      "pbe_seqn" ASC
)
go
commit work
go


------------------------------------------------
--   Create functions
------------------------------------------------

setuser "DBA"
go
commit work
go


------------------------------------------------
--   Create views
------------------------------------------------


create view
  %% ============================================================
  %%   View: defscriptsxcaso
  %% ============================================================
  "DBA".defscriptsxcaso
  as select
defscripts.idds,defscripts.iddef,defscripts.idscr,defscripts.seccion,
    defscripts.ord,defscripts.script,definiciones.idc
    from DBA.casos,DBA.definiciones,DBA.defscripts
    where casos.idc=casos.idc
    and definiciones.iddef=defscripts.iddef with check option
go

create view
  %% ============================================================
  %%   View: defxcaso
  %% ============================================================
  "DBA".defxcaso
  as select definiciones.iddef,definiciones.idc,definiciones.filename
    from DBA.casos,DBA.definiciones
    where casos.idc=definiciones.idc with check option
go
commit work
go


------------------------------------------------
--   Create user messages
------------------------------------------------
```

```
commit work
go


------------------------------------------------
--   Create procedures
------------------------------------------------

setuser "DBA"
go

create procedure dba.bindtabletocase(@table varchar(30),@case varchar(30)) as
begin
  insert into casosxtabla
    select id,idc from
      tablas as a,
      casos as b where
      a.nombre = @table and
      b.nombre = @case
end
go

create procedure "DBA".copy_idc(@idc_old integer,@idc_new integer) as
begin
  declare @iddefini integer
  select num=number(*),* into #tmp_defxcaso
    from defxcaso where idc=@idc_old
  select distinct b.num,a.* into #tmp_defscriptsxcaso
    from defscriptsxcaso as a
    ,#tmp_defxcaso as b
    where a.iddef=b.iddef order by
    a.iddef asc,a.seccion asc,a.ord asc,a.idds asc
  select @iddefini="max"(iddef)
    from definiciones
  update #tmp_defxcaso set
    iddef=@iddefini+num,
    idc=@idc_new
  insert into definiciones
    select iddef,
      idc,
      filename
      from #tmp_defxcaso
  update #tmp_defscriptsxcaso set
    iddef=@iddefini+num,
    idc=@idc_new
  insert into defscripts(iddef,idscr,seccion,ord,script)
    select iddef,idscr,seccion,ord,script
      from #tmp_defscriptsxcaso
end
go
commit work
go


------------------------------------------------
--   Create triggers
------------------------------------------------
```

```
setuser "DBA"
go

create trigger %% ==========================================================
%%   Database name:  repgen
%%   DBMS name:      Sybase SQL Anywhere 5.5
%%   Created on:     9/08/99  12:34 AM
%% ==========================================================
%  After update trigger "tua_casos" for table "casos"
tua_casos after update of idc
on casos
referencing old as old_upd new as new_upd
for each row begin
  declare user_defined_exception exception for sqlstate value '99999';
  declare found integer;
  %  Modify parent code of "casos" for all children in "casosxtabla"
  if(new_upd.idc<>old_upd.idc) then
    update casosxtabla set
      idc=new_upd.idc
      where idc=old_upd.idc
  end if
  ; %  Modify parent code of "casos" for all children in "definiciones"
  if(new_upd.idc<>old_upd.idc) then
    update definiciones set
      idc=new_upd.idc
      where idc=old_upd.idc
  end if
exception
  when others then
    message 'Exception in after update trigger(tua_casos) of table casos';
    resignal
end
go

create trigger %  After delete trigger "tda_casos" for table "casos"
tda_casos after delete on casos
referencing old as old_del
for each row begin
  declare user_defined_exception exception for sqlstate value '99999';
  declare found integer;
  %  Delete all children in "casosxtabla"
  delete from casosxtabla
    where idc=old_del.idc;
  %  Delete all children in "definiciones"
  delete from definiciones
    where idc=old_del.idc
exception
  when others then
    message 'Exception in after delete trigger(tda_casos) of table casos';
    resignal
end
go

create trigger %  Before insert trigger "tib_casosxtabla" for table
"casosxtabla"
tib_casosxtabla before insert on casosxtabla
referencing new as new_ins
```

```
for each row begin
  declare user_defined_exception exception for sqlstate value '99999';
  declare insert_child_parent_exist_exception exception for sqlstate value
'99991';
  declare found integer;
  % Parent "tablas" must exist when inserting a child in "casosxtabla"
  if(new_ins.id is not null) then
    set found=0;
    select 1 into found
      from dummy
      where exists(select 1
        from tablas
        where id=new_ins.id);
    if found<>1 then
      signal insert_child_parent_exist_exception
    end if
  end if
  ;
  % Parent "casos" must exist when inserting a child in "casosxtabla"
  if(new_ins.idc is not null) then
    set found=0;
    select 1 into found
      from dummy
      where exists(select 1
        from casos
        where idc=new_ins.idc);
    if found<>1 then
      signal insert_child_parent_exist_exception
    end if
  end if
exception
  when insert_child_parent_exist_exception
  then
    message 'Error: Trigger(tib_casosxtabla) of table casosxtabla';
    message '        Parent code must exist when inserting a child!';
    signal user_defined_exception
  when others then
    message 'Exception in before insert trigger(tib_casosxtabla) of  table
casosxtabla';
    resignal
end
go

create trigger %  Before update trigger "tub_casosxtabla" for table
"casosxtabla"
tub_casosxtabla before update of id,
idc on casosxtabla
referencing old as old_upd new as new_upd
for each row begin
  declare user_defined_exception exception for sqlstate value '99999';
  declare update_change_column_exception exception for sqlstate value '99991';
  declare update_child_change_parent_exception exception for sqlstate value
'99992';
  declare update_child_parent_exist_exception exception for sqlstate value
'99993';
  declare update_parent_restrict_exception exception for sqlstate value '99994';
  declare found integer;
  % Parent "tablas" must exist when updating a child in "casosxtabla"
```

```
   if(new_upd.id is not null
     and((old_upd.id is null)
     or(new_upd.id<>old_upd.id))) then
     set found=0;
     select 1 into found
       from dummy
       where exists(select 1
         from tablas
         where id=new_upd.id);
     if found<>1 then
       signal update_child_parent_exist_exception
     end if
   end if
   ;
   %  Parent "casos" must exist when updating a child in "casosxtabla"
   if(new_upd.idc is not null
     and((old_upd.idc is null)
     or(new_upd.idc<>old_upd.idc))) then
     set found=0;
     select 1 into found
       from dummy
       where exists(select 1
         from casos
         where idc=new_upd.idc);
     if found<>1 then
       signal update_child_parent_exist_exception
     end if
   end if
 exception
   when update_change_column_exception
   then
     message 'Error: Trigger(tub_casosxtabla) of table casosxtabla';
     message '        Non modifiable column cannot be modified!';
     signal user_defined_exception
   when update_child_change_parent_exception then
     message 'Error: Trigger(tub_casosxtabla) of table casosxtabla';
     message '        Cannot modify parent code in child!';
     signal user_defined_exception
   when update_child_parent_exist_exception then
     message 'Error: Trigger(tub_casosxtabla) of table casosxtabla';
     message '        Parent must exist when updating a child!';
     signal user_defined_exception
   when update_parent_restrict_exception then
     message 'Error: Trigger(tub_casosxtabla) of table casosxtabla';
     message '        Cannot modify parent code if children still exist!';
     signal user_defined_exception
   when others then
     message 'Exception in before update trigger(tub_casosxtabla) of table
 casosxtabla';
     resignal
 end
 go

 create trigger ;  Before insert trigger "tib_definiciones" for table
 "definiciones"
 tib_definiciones before insert on definiciones
 referencing new as new_ins
 for each row begin
```

```
  declare user_defined_exception exception for sqlstate value '99999';
  declare insert_child_parent_exist_exception exception for sqlstate value
'99991';
  declare found integer;
  % Parent "casos" must exist when inserting a child in "definiciones"
  if(new_ins.idc is not null) then
    set found=0;
    select 1 into found
      from dummy
      where exists(select 1
        from casos
        where idc=new_ins.idc);
    if found<>1 then
      signal insert_child_parent_exist_exception
    end if
  end if
exception
  when insert_child_parent_exist_exception
  then
    message 'Error: Trigger(tib_definiciones) of table definiciones';
    message '        Parent code must exist when inserting a child!';
    signal user_defined_exception
  when others then
    message 'Exception in before insert trigger(tib_definiciones) of  table
definiciones';
    resignal
end
go

create trigger '  Before update trigger "tub_definiciones" for table
"definiciones"
tub_definiciones before update of iddef,
idc on definiciones
referencing old as old_upd new as new_upd
for each row begin
  declare user_defined_exception exception for sqlstate value '99999';
  declare update_change_column_exception exception for sqlstate value '99991';
  declare update_child_change_parent_exception exception for sqlstate value
'99992';
  declare update_child_parent_exist_exception exception for sqlstate value
'99993';
  declare update_parent_restrict_exception exception for sqlstate value '99994';
  declare found integer;
  %  Parent "casos" must exist when updating a child in "definiciones"
  if(new_upd.idc is not null
    and((old_upd.idc is null)
    or(new_upd.idc<>old_upd.idc))) then
    set found=0;
    select 1 into found
      from dummy
      where exists(select 1
        from casos
        where idc=new_upd.idc);
    if found<>1 then
      signal update_child_parent_exist_exception
    end if
  end if
exception
```

```
  when update_change_column_exception
  then
    message 'Error: Trigger(tub_definiciones) of table definiciones';
    message '        Non modifiable column cannot be modified!';
    signal user_defined_exception
  when update_child_change_parent_exception then
    message 'Error: Trigger(tub_definiciones) of table definiciones';
    message '        Cannot modify parent code in child!';
    signal user_defined_exception
  when update_child_parent_exist_exception then
    message 'Error: Trigger(tub_definiciones) of table definiciones';
    message '        Parent must exist when updating a child!';
    signal user_defined_exception
  when update_parent_restrict_exception then
    message 'Error: Trigger(tub_definiciones) of table definiciones';
    message '        Cannot modify parent code if children still exist!';
    signal user_defined_exception
  when others then
    message 'Exception in before update trigger(tub_definiciones) of table
definiciones';
    resignal
end
go

create trigger % After update trigger "tua_definiciones" for table
"definiciones"
tua_definiciones after update of iddef,
idc on definiciones
referencing old as old_upd new as new_upd
for each row begin
  declare user_defined_exception exception for sqlstate value '99999';
  declare found integer;
  % Modify parent code of "definiciones" for all children in "defscripts"
  if(new_upd.iddef<>old_upd.iddef) then
    update defscripts set
      iddef=new_upd.iddef
      where iddef=old_upd.iddef
  end if
exception
  when others then
    message 'Exception in after update trigger(tua_definiciones) of table
definiciones';
    resignal
end
go

create trigger % After delete trigger "tda_definiciones" for table
"definiciones"
tda_definiciones after delete on definiciones
referencing old as old_del
for each row begin
  declare user_defined_exception exception for sqlstate value '99999';
  declare found integer;
  % Delete all children in "defscripts"
  delete from defscripts
    where iddef=old_del.iddef
exception
  when others then
```

```
    message 'Exception in after delete trigger(tda_definiciones) of table
definiciones';
    resignal
end
go

create trigger ; Before insert trigger "tib_defscripts" for table "defscripts"
tib_defscripts before insert on defscripts
referencing new as new_ins
for each row begin
  declare user_defined_exception exception for sqlstate value '99999';
  declare insert_child_parent_exist_exception exception for sqlstate value
'99991';
  declare found integer;
    Parent "definiciones" must exist when inserting a child in "defscripts"
  if(new_ins.iddef is not null) then
    set found=0;
    select 1 into found
      from dummy
      where exists(select 1
        from definiciones
        where iddef=new_ins.iddef);
    if found<>1 then
      signal insert_child_parent_exist_exception
    end if
  end if
  ;
  ; Parent "scripts" must exist when inserting a child in "defscripts"
  if(new_ins.idscr is not null) then
    set found=0;
    select 1 into found
      from dummy
      where exists(select 1
        from scripts
        where idscr=new_ins.idscr);
    if found<>1 then
      signal insert_child_parent_exist_exception
    end if
  end if
exception
  when insert_child_parent_exist_exception
  then
    message 'Error: Trigger(tib_defscripts) of table defscripts';
    message '        Parent code must exist when inserting a child!';
    signal user_defined_exception
  when others then
    message 'Exception in before insert trigger(tib_defscripts) of  table
defscripts';
    resignal
end
go

create trigger . Before update trigger "tub_defscripts" for table "defscripts"
tub_defscripts before update of idds,
iddef,
idscr on defscripts
referencing old as old_upd new as new_upd
for each row begin
```

```
  declare user_defined_exception exception for sqlstate value '99999';
  declare update_change_column_exception exception for sqlstate value '99991';
  declare update_child_change_parent_exception exception for sqlstate value
'99992';
  declare update_child_parent_exist_exception exception for sqlstate value
'99993';
  declare update_parent_restrict_exception exception for sqlstate value '99994';
  declare found integer;
  % Parent "definiciones" must exist when updating a child in "defscripts"
  if(new_upd.iddef is not null
    and((old_upd.iddef is null)
    or(new_upd.iddef<>old_upd.iddef))) then
    set found=0;
    select 1 into found
      from dummy
      where exists(select 1
        from definiciones
        where iddef=new_upd.iddef);
    if found<>1 then
      signal update_child_parent_exist_exception
    end if
  end if
  ;
  % Parent "scripts" must exist when updating a child in "defscripts"
  if(new_upd.idscr is not null
    and((old_upd.idscr is null)
    or(new_upd.idscr<>old_upd.idscr))) then
    set found=0;
    select 1 into found
      from dummy
      where exists(select 1
        from scripts
        where idscr=new_upd.idscr);
    if found<>1 then
      signal update_child_parent_exist_exception
    end if
  end if
exception
  when update_change_column_exception
  then
    message 'Error: Trigger(tub_defscripts) of table defscripts';
    message '        Non modifiable column cannot be modified!';
    signal user_defined_exception
  when update_child_change_parent_exception then
    message 'Error: Trigger(tub_defscripts) of table defscripts';
    message '        Cannot modify parent code in child!';
    signal user_defined_exception
  when update_child_parent_exist_exception then
    message 'Error: Trigger(tub_defscripts) of table defscripts';
    message '        Parent must exist when updating a child!';
    signal user_defined_exception
  when update_parent_restrict_exception then
    message 'Error: Trigger(tub_defscripts) of table defscripts';
    message '        Cannot modify parent code if children still exist!';
    signal user_defined_exception
  when others then
    message 'Exception in before update trigger(tub_defscripts) of table
defscripts';
```

```
    resignal
end
go

create trigger % After update trigger "tua_scripts" for table "scripts"
tua_scripts after update of idscr
on scripts
referencing old as old_upd new as new_upd
for each row begin
  declare user_defined_exception exception for sqlstate value '99999';
  declare found integer;
  % Modify parent code of "scripts" for all children in "defscripts"
  if(new_upd.idscr<>old_upd.idscr) then
    update defscripts set
      idscr=new_upd.idscr
      where idscr=old_upd.idscr
  end if
exception
  when others then
    message 'Exception in after update trigger(tua_scripts) of table scripts';
    resignal
end
go

create trigger % After delete trigger "tda_scripts" for table "scripts"
tda_scripts after delete on scripts
referencing old as old_del
for each row begin
  declare user_defined_exception exception for sqlstate value '99999';
  declare found integer;
  % Delete all children in "defscripts"
  delete from defscripts
  where idscr=old_del.idscr
exception
  when others then
    message 'Exception in after delete trigger(tda_scripts) of table scripts';
    resignal
end
go

create trigger % After update trigger "tua_tablas" for table "tablas"
tua_tablas after update of id
on tablas
referencing old as old_upd new as new_upd
for each row begin
  declare user_defined_exception exception for sqlstate value '99999';
  declare found integer;
  % Modify parent code of "tablas" for all children in "casosxtabla"
  if(new_upd.id<>old_upd.id) then
    update casosxtabla set
      id=new_upd.id
      where id=old_upd.id
  end if
exception
  when others then
    message 'Exception in after update trigger(tua_tablas) of table tablas';
    resignal
end
```

```
go

create trigger 1  After delete trigger "tda_tablas" for table "tablas"
tda_tablas after delete on tablas
referencing old as old_del
for each row begin
  declare user_defined_exception exception for sqlstate value '99999';
  declare found integer;
  -- Delete all children in "casosxtabla"
  delete from casosxtabla
    where id=old_del.id
exception
  when others then
    message 'Exception in after delete trigger(tda_tablas) of table tablas';
    resignal
end
go
setuser "DBA"
go
commit work
go


------------------------------------------------
--   Create SQL remote definitions
------------------------------------------------

CREATE REMOTE TYPE "FILE" ADDRESS ''
go
CREATE REMOTE TYPE "MAPI" ADDRESS ''
go
CREATE REMOTE TYPE "VIM" ADDRESS ''
go
CREATE REMOTE TYPE "SMTP" ADDRESS ''
go
commit work
go


------------------------------------------------
--   Remove SQL remote definitions
------------------------------------------------

commit work
go


------------------------------------------------
--   Repserver Commit Offsets
------------------------------------------------

commit work
go


------------------------------------------------
--   Check view definitions
------------------------------------------------
```

```
GRANT CONNECT TO "DBA" IDENTIFIED BY ENCRYPTED
'\x7B\x5E\x7E\xEA\x08\x97\xF9\x72\x95\xA8\x4B\x45\x24\xE0\xF8\x84\x12\x6F\x82\x9
E\x6F\x29\xA1\x71\x37\xC3\x75\x02\xD4\x09\x3B\xA1\xA6\xA9\x29\x19'
go
commit work
go


------------------------------------------------
--   Create integrated logins
------------------------------------------------

commit work
go


------------------------------------------------
--   Set option values
------------------------------------------------


SET OPTION Statistics =
go
SET OPTION Date_order =
go
SET OPTION describe_java_format=
go


--
--SQL Option Statements for user
--

SET OPTION "PUBLIC"."Blocking" = 'ON'
go
SET OPTION "PUBLIC"."Checkpoint_time" = '60'
go
SET OPTION "PUBLIC"."Conversion_error" = 'ON'
go
SET OPTION "PUBLIC"."Date_format" = 'YYYY-MM-DD'
go
SET OPTION "PUBLIC"."Date_order" = 'YMD'
go
SET OPTION "PUBLIC"."Isolation_level" = '0'
go
SET OPTION "PUBLIC"."Lock_rejected_rows" = 'OFF'
go
SET OPTION "PUBLIC"."Precision" = '30'
go
SET OPTION "PUBLIC"."Recovery_time" = '2'
go
SET OPTION "PUBLIC"."Replicate_all" = 'OFF'
go
SET OPTION "PUBLIC"."Row_counts" = 'OFF'
go
SET OPTION "PUBLIC"."Scale" = '6'
go
```

```
SET OPTION "PUBLIC"."Thread_count" = '0'
go
SET OPTION "PUBLIC"."Thread_stack" = '750'
go
SET OPTION "PUBLIC"."Thread_swaps" = '18'
go
SET OPTION "PUBLIC"."Timestamp_format" = 'YYYY-MM-DD HH:NN:SS.SSS'
go
SET OPTION "PUBLIC"."Time_format" = 'HH:NN:SS.SSS'
go
SET OPTION "PUBLIC"."Wait_for_commit" = 'OFF'
go
SET OPTION "PUBLIC"."Quoted_identifier" = 'ON'
go
SET OPTION "PUBLIC"."Allow_nulls_by_default" = 'ON'
go
SET OPTION "PUBLIC"."Automatic_timestamp" = 'OFF'
go
SET OPTION "PUBLIC"."Query_plan_on_open" = 'OFF'
go
SET OPTION "PUBLIC"."Cooperative_commits" = 'ON'
go
SET OPTION "PUBLIC"."Cooperative_commit_timeout" = '250'
go
SET OPTION "PUBLIC"."Delayed_commits" = 'OFF'
go
SET OPTION "PUBLIC"."Delayed_commit_timeout" = '500'
go
SET OPTION "PUBLIC"."Non_keywords" = ''
go
SET OPTION "PUBLIC"."SQL_flagger_error_level" = 'W'
go
SET OPTION "PUBLIC"."SQL_flagger_warning_level" = 'W'
go
SET OPTION "PUBLIC"."Ansi_blanks" = 'OFF'
go
SET OPTION "PUBLIC"."Ansi_integer_overflow" = 'OFF'
go
SET OPTION "PUBLIC"."String_rtruncation" = 'OFF'
go
SET OPTION "PUBLIC"."Divide_by_zero_error" = 'ON'
go
SET OPTION "PUBLIC"."Ansinull" = 'ON'
go
SET OPTION "PUBLIC"."Ansi_permissions" = 'ON'
go
SET OPTION "PUBLIC"."Close_on_endtrans" = 'ON'
go
SET OPTION "PUBLIC"."Tsql_variables" = 'OFF'
go
SET OPTION "PUBLIC"."RI_Trigger_time" = 'AFTER'
go
SET OPTION "PUBLIC"."Tsql_hex_constant" = 'ON'
go
SET OPTION "PUBLIC"."Chained" = 'ON'
go
SET OPTION "PUBLIC"."Nearest_century" = '0'
go
```

```
SET OPTION "PUBLIC"."Fire_triggers" = 'ON'
go
SET OPTION "PUBLIC"."Background_priority" = 'OFF'
go
SET OPTION "PUBLIC"."Auto_commit" = 'OFF'
go
SET OPTION "PUBLIC"."Auto_refetch" = 'ON'
go
SET OPTION "PUBLIC"."Bell" = 'ON'
go
SET OPTION "PUBLIC"."Command_delimiter" = ';'
go
SET OPTION "PUBLIC"."Commit_on_exit" = 'ON'
go
SET OPTION "PUBLIC"."Echo" = 'ON'
go
SET OPTION "PUBLIC"."Headings" = 'On'
go
SET OPTION "PUBLIC"."Input_format" = 'ASCII'
go
SET OPTION "PUBLIC"."ISQL_log" = ''
go
SET OPTION "PUBLIC"."NULLS" = '(NULL)'
go
SET OPTION "PUBLIC"."On_error" = 'PROMPT'
go
SET OPTION "PUBLIC"."Output_format" = 'ASCII'
go
SET OPTION "PUBLIC"."Output_length" = '0'
go
SET OPTION "PUBLIC"."Output_nulls" = ''
go
SET OPTION "PUBLIC"."Quiet" = 'OFF'
go
SET OPTION "PUBLIC"."Screen_format" = 'TEXT'
go
SET OPTION "PUBLIC"."SQLConnect" = ''
go
SET OPTION "PUBLIC"."SQLStart" = ''
go
SET OPTION "PUBLIC"."Statistics" = '3'
go
SET OPTION "PUBLIC"."Truncation_length" = '30'
go
SET OPTION "PUBLIC"."Verify_all_columns" = 'Off'
go
SET OPTION "PUBLIC"."Delete_old_logs" = 'Off'
go
SET OPTION "PUBLIC"."Replication_error" = ''
go
SET OPTION "PUBLIC"."Verify_threshold" = '1000'
go
commit work
go


------------------------------------------------
--    Destroy userids
```

```
-----------------------------------------------------

commit work
go
```

## BIBLIOGRAFIA

**Fundamentos y modelos de bases de datos**
Adoración de Miguel Castaño y Mario G. Piattini Velthois
2ª. Edición
Ed. Alfaomega

**Introducción a los Sistemas de Bases de Datos. Volumen I.**
C. J. Date
5ª. Edición.
Ed. Addison – Wesley Iberoamericana

**Análisis y Diseño de Sistemas de Información**
James A. Senn
2ª. Edición
Ed. Mc Graw Hill

**Comunicación de datos, redes de computadores y sistemas abiertos**
Fred Halsall
4ª. Edición
Ed. Addison Wesley

**Diseño y Administración de Bases de Datos**
Gary W. Hansen y James V. Hansen
2ª. Edición
Ed. Prentice Hall

**Data Replication**
Marie Buretta
1997
Wiley Computer Publishing

**Replication Server Administration Guide**
Sybase, Inc.
2000

**Replication Server Design Guide**
Sybase, Inc.
2000

**Replication Server Reference Manual**
Sybase, Inc.
2000

**Manuales del curso "Fast Track to Replication Server 12.0"**

Sybase, Inc.
2000

**http://my.sybase.com/detail?id=204851**

**http://www.atomica.com**

Sybase, Inc.
2000

**PowerBuilder 5.0**
Michael MacDonald
1996
SYS-CON Publications

**http://my.sybase.com/detail?id=204851**

**http://www.atomica.com**