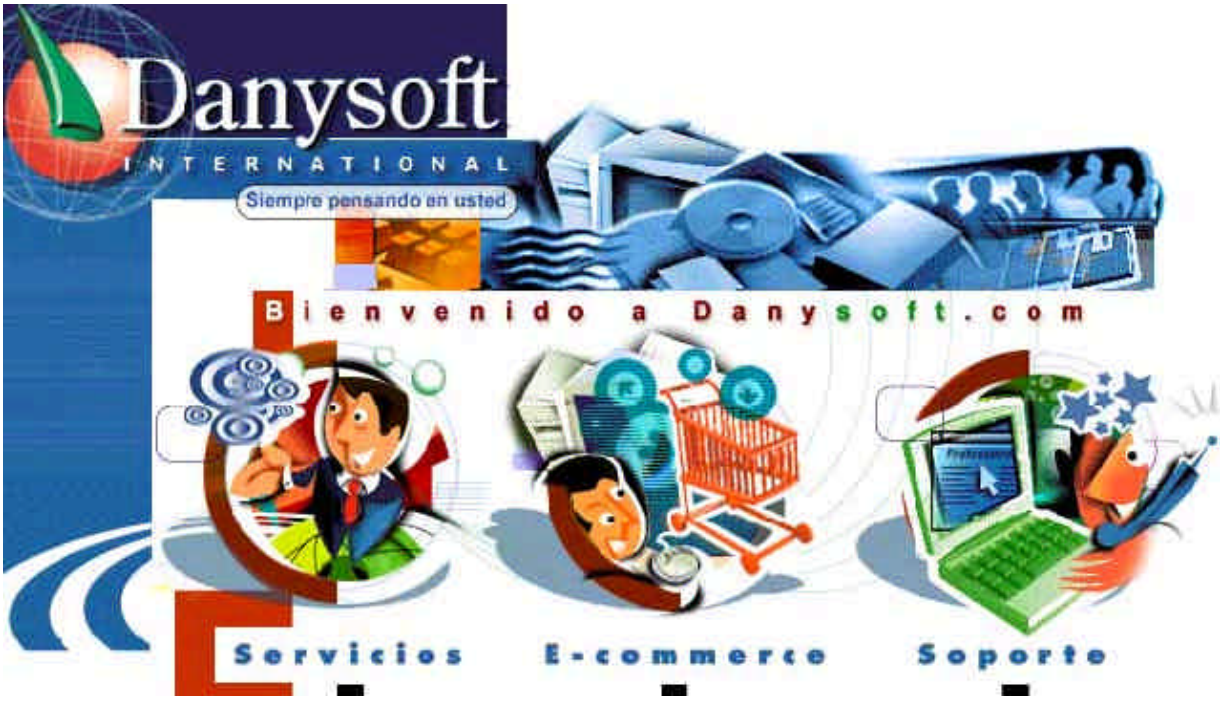
Artículos técnicos Grupo Danysoft:

Introducción al SQL de InterBase: DDL y DML



Por Pablo Reyes – Equipo Grupo Danysoft
abril de 2002 - (902) 123146
www.danysoft.com

Este documento se ha realizado utilizando Doc-To-Help®, distribuido por :

Danysoft Internacional
Avda de España 17
28100 Alcobendas – Madrid

Tfno. 902.123146
Fax. 902.123145

http://www.danysoft.com

http://www.danyshop.com

danysoft@danysoft.com

Introducción al SQL de InterBase: DDL y DML

Este artículo es el segundo de una serie dedicados al uso de InterBase con Delphi. Debido a su extensión, el tema de este artículo será dividido en dos:

- un primer artículo dedicado a las herramientas visuales de InterBase y a sentencias DDL,
- un segundo artículo dedicado a sentencias DML.

Objetivo

El objetivo de este artículo es presentar las sentencias fundamentales del lenguaje SQL de InterBase. Para ello seguiremos desde el principio los pasos necesarios para crear una bases de datos y almacenar algunos datos en ella. Lo que haremos concretamente será migrar las tablas Paradox del alias DBDemos a una base de datos de InterBase.

SQL (Structured Query Language - Lenguaje Estructurado de Consulta) es un lenguaje estándar utilizado para crear, modificar, mantener y consultar una base de datos relacional. Si bien existe una especificación estándar que es soportada por la mayoría de los fabricantes (yo diría todos) cada uno ha desarrollado sus propias extensiones haciendo que la ambicionada compatibilidad, teóricamente garantizada por la adopción de un estándar, se perdiera en el camino. Hoy en día, aunque todos dicen soportar el estándar SQL ANSI 92, no hay dos fabricantes que posean un lenguaje SQL 100% compatible entre si.

Este artículo está dedicado al SQL de InterBase que, aunque soporta el estándar SQL ANSI 92 nivel de entrada (SQL ANSI 92 entry level), tiene sus propias extensiones.

Vayamos a lo nuestro. Las sentencias SQL pueden ser clasificadas en dos grupos:

- **DDL (Data Definition Languaje - lenguaje de definición de datos):** las sentencias DDL son aquellas utilizadas para la creación de una base de datos y todos sus componentes: tablas, índices, relaciones, disparadores (triggers), procedimientos almacenados, etc.
- **DML (Data Manipulation Languaje - lenguaje de manipulación de datos):** las sentencias DML son aquellas utilizadas para insertar, borrar, modificar y consultar los datos de una base de datos.

A lo largo de este artículo haremos uso de sentencias SQL de ambos grupos aunque trataré de explicar más detalladamente las del segundo por tratarse de las más utilizadas desde una aplicación Delphi.

Herramientas disponibles

InterBase provee un conjunto de herramientas visuales y de línea de comando para realizar una amplia gama de tareas. En Windows estamos acostumbrados a las herramientas visuales por lo que las herramientas de línea de comando no serán incluidas en este artículo. De las herramientas visuales, hay dos que están especialmente relacionadas con el contenido de este artículo: IBConsole e Interactive SQL.

IBConsole

IBConsole provee herramientas administrativas, una herramienta llamada Interactive SQL para ejecutar sentencias SQL de manera interactiva y herramientas para verificar las condiciones de comunicación de la red de trabajo.

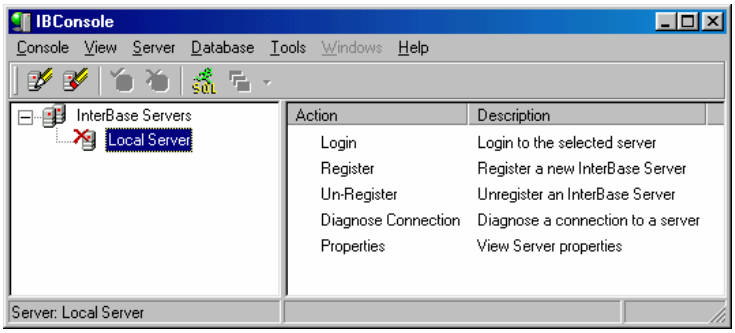


Imagen de Pantalla 1 - IBConsole

El primer paso para utilizar IBConsole es registrar el servidor InterBase. IBConsole puede ser utilizado para conectarse a servidores locales y remotos. Para registrar un servidor seleccionar del menú **Server | Register ...** lo que mostrará la ventana de diálogo para registrar y / o conectarse a un servidor local o remoto.

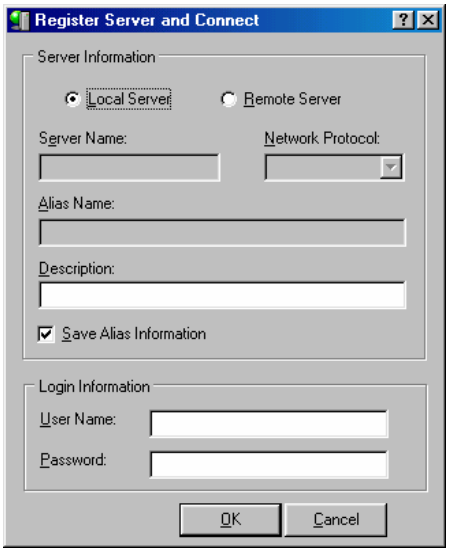


Imagen de Pantalla 2 - Registrar / Conectarse a un Servidor

Completar los campos según corresponda a su propia configuración de acuerdo a la siguiente explicación de cada uno de ellos:

- Indicar si el servidor es local o remoto. En el caso que sea remoto, indicar el nombre del servidor y el protocolo de red a utilizar. Si el protocolo seleccionado es TCP/IP entonces se puede indicar la dirección IP en lugar del nombre.
- Si se quiere registrar el servidor...
    - Indicar un nombre de alias. En el caso de que el servidor sea local, el nombre de alias es automático.

    - Indicar una descripción para el servidor.
    - Indicar que la información de alias debe ser almacenada.
- Si se quiere conectar al servidor...
    - Indicar el nombre de usuario (SYSDBA)
    - Indicar la clave (masterkey)

Una vez registrado y conectados al servidor crearemos la base de datos. En InterBase cada base de datos es almacenada en un archivo independiente. Para crear una base de datos sólo debemos indicar un nombre de archivo. La extensión de las bases de datos de InterBase es .GDB. Seleccionar del menú **Database | Create Database ...** lo que mostrará la ventana de diálogo para crear una base de datos.

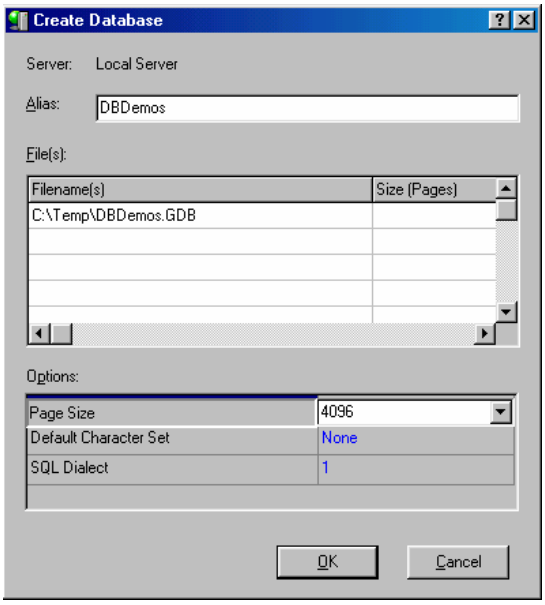


Imagen de Pantalla 3 - Crear una base de datos

Completar los campos según corresponda a su propia configuración de acuerdo a la siguiente explicación de cada uno de ellos:

- Indicar el alias de la base de datos.
- Indicar el nombre de archivo incluyendo la ruta de acceso y la extensión .GDB.

Una vez creada la base de datos utilizaremos la herramienta Interactive SQL para crear las tablas. Si bien esta herramienta es una aplicación independiente a partir de la versión 6 de InterBase no está disponible desde el menú de Windows sino desde IBConsole.

Interactive SQL

Interactive SQL es una herramienta que permite ejecutar sentencias SQL de manera interactiva. Está disponible desde IBConsole. Seleccionar del menú **Tools | Interactive SQL ...** lo que iniciará Interactive SQL.

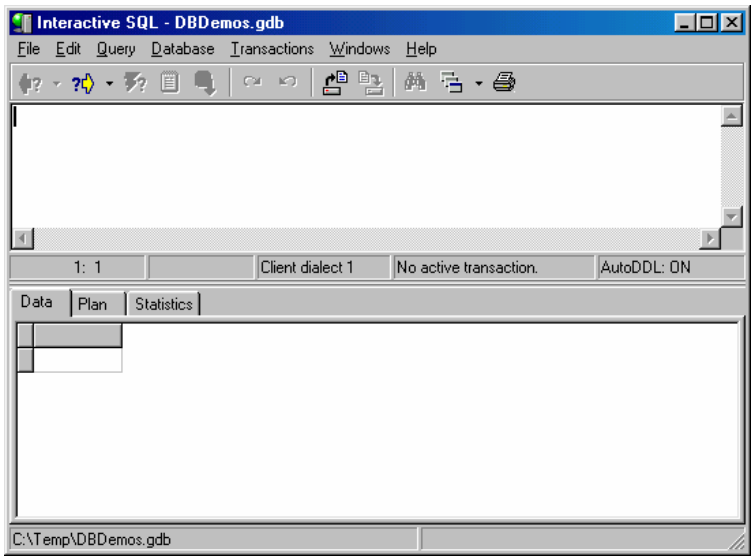


Imagen de Pantalla 4 - Interactive SQL

La ventana de Interactive SQL está dividida en dos partes. La parte superior es donde escribimos las sentencias SQL mientras que en la parte inferior vemos el resultado de ejecutarlas.

Migración de DBDemos

Por razones de espacio no vamos a migrar todas las tablas del alias DBDemos en este artículo. Lo que haremos será migrar sólo algunas tablas para conocer de a poco las principales opciones de las cláusulas **create table**, **create generator** y **create procedure**.

Creación de una tabla

Vamos a empezar creando una tabla simple: *country.db*. Tiene la siguiente estructura:

Tabla: country.db			
Nombre de Campo	**Tipo de Dato**	**Tamaño**	**Es Clave Primaria**
Name	Alpha	24	Si
Capital	Alpha	24	
Continent	Alpha	24	
Area	Numeric		
Population	Numeric		

Tabla 1- Estructura de la tabla country.db

El siguiente código muestra la sentencia SQL para crearla. Copiarla en Interactive SQL y presionar Ctrl+E para ejecutarla.

```
create table country
(name varchar(24) not null primary key,
capital varchar(24) not null,
continent varchar(24) not null,
area integer not null,
population integer not null)
```

Código Fuente 1 - Creación de la tabla country

La sentencia **create table** tiene muchas opciones. Basta mirar la ayuda en línea de Interactive SQL para ver que sólo la descripción de su sintaxis abarca un par de páginas. Para la creación de la tabla *country* hemos utilizado una de sus formas más simples. En primer lugar indicamos el nombre de la tabla y luego, entre paréntesis y separados por comas, los campos que la componen. Para cada campo indicamos su nombre, tipo de dato e imposiciones. Todos los campos tienen las mismas imposiciones salvo el campo *name* que, además de la imposición **not null**, tiene la imposición **primary key**. Esta última imposición merece algunos comentarios. Una tabla puede tener una y solo una clave primaria que puede estar compuesta por uno o más campos los cuales deben tener la imposición **not null** obligatoriamente. Los campos de la calve primaria no pueden tener valores nulos. InterBase crea automáticamente un índice con los campos de la clave primaria.

Con respecto a los tipos de dato no voy a hacer demasiados comentarios, salvo cuando estrictamente sea necesario. La documentación de InterBase explica detalladamente los tipos de datos soportados con lo cual, en un proceso de migración como el que estamos llevando a cabo, todo se reduce a encontrar el tipo de datos que mejor se adapte a nuestras necesidades. Si alguna vez crearon una tabla, aunque sea con Paradox, entonces ya saben de lo que estoy hablando.

Integridad referencial

Vamos a crear ahora dos tablas que están relacionadas entre si: *parts.db* y *vendors.db*. La estructura de cada una de ellas es la siguiente:

Tabla: parts.db			
Nombre de Campo	**Tipo de Dato**	**Tamaño**	**Es Clave Primaria**
PartNo	Numeric		Si
VendorNo	Numeric		
Description	Alpha	30	
OnHand	Numeric		
OnOrder	Numeric		
Cost	Money		
ListPrice	Money		

Tabla 2 - Estructura de la tabla parts.db

Tabla: vendors.db			
Nombre de Campo	**Tipo de Dato**	**Tamaño**	**Es Clave Primaria**
VendorNo	Numeric		Si
VendorName	Alpha	30	
Address1	Alpha	30	
Address2	Alpha	30	
City	Alpha	20	
State	Alpha	20	
Zip	Alpha	10	
Country	Alpha	15	
Phone	Alpha	15	
FAX	Alpha	15	

Preferred	Logical		

Tabla 3 - Estructura de la tabla vendors.db

La tabla *parts.db* está relacionada con la tabla *vendors.db* por medio del campo *VendorNo*. Esta relación mantiene la integridad de los datos de la base de datos. Un registro de la tabla *parts.db* no puede estar relacionado con registro de la tabla *vendors.db*. Cualquier modificación a los datos de alguna de las dos tablas sería cancelada si produjera que la imposición anterior fuera violada.

La sentencia SQL para la creación de la *vendors.db* es la siguiente:

```
create table vendors
(VendorNo integer not null primary key,
VendorName varchar(30) not null,
Address1 varchar(30),
Address2 varchar(30),
City varchar(20),
State varchar(20),
Zip varchar(10),
Country varchar(15),
Phone varchar(15) not null,
FAX varchar(15),
Preferred smallint default 0 check(Preferred = 0 or Preferred = 1))
```

Código Fuente 2 - Creación de la tabla vendors

InterBase no soporta el tipo de dato *logical* o *boolean* lo cual no representa un problema mayor. Al campo *Preferred* lo creamos con el tipo de dato **smallint** y el valor por defecto 0. Además establecimos una imposición sobre el campo: su valor puede ser 0 ó 1. De esta manera tenemos el equivalente al tipo de dato *logical* o *boolean*. El valor 0 es equivalente a falso y el valor 1 es equivalente a verdadero.

La cláusula **default** permite asignar un valor por defecto a un campo en el caso de que no se provea uno. La cláusula **check** permite establecer una imposición sobre el valor de un campo. La imposición debe ser una condición lógica cuyo resultado debe ser verdadero para aceptar el valor del campo o falso para rechazarlo. En este caso cualquier intento por asignarle al campo *Preferred* un valor distinto de 0 ó 1 será rechazado.

La sentencia SQL para la creación de la tabla *parts.db* es la siguiente:

```
create table parts
(PartNo integer not null,
VendorNo integer not null,
Description varchar(30) not null,
OnHand integer,
OnOrder integer,
Cost decimal(12,2),
ListPrice decimal(12,2),
primary key (PartNo),
foreign key (VendorNo) references vendors (VendorNo))
```

Código Fuente 3 - Creación de la tabla parts

Para la creación de la clave primaria de la tabla *parts* utilizamos una sintaxis diferente. En lugar de indicar la clave primaria junto con la definición del campo, indicamos la clave primaria luego de la definición de los campos como una lista de nombres de campos. En el

caso de la tabla *parts* la clave primaria es simple, es decir, está formada por un solo campo. Pero si la clave primaria fuera compuesta, es decir, estuviera formada por más de un campo, deberíamos indicar la lista de nombres de campos que la componen separándolos con comas.

Otra novedad en esta sentencia SQL es la creación de una relación de integridad referencial por medio de la cláusula **foreign key - references**. En este caso la tabla *parts* está relacionada con la tabla *vendors* por medio del campo *VendorNo* (el campo tiene el mismo nombre en ambas tablas pero podría no tenerlo). Para que esto sea posible la clave primaria de la tabla *vendors* debe estar formada por el campo *VendorNo*. De aquí en adelante InterBase se encargará de que esta relación de integridad referencial sea respetada rechazando cualquier modificación a los datos de ambas tablas que la viole.

Generadores

Siguiendo con nuestra migración es el turno de la tabla *customer.db*. La estructura de esta tabla es la siguiente:

Tabla: customer.db			
Nombre de Campo	**Tipo de Dato**	**Tamaño**	**Es Clave Primaria**
CustNo	Numeric		Sí
Company	Alpha	30	
Addr1	Alpha	30	
Addr2	Alpha	30	
City	Alpha	15	
State	Alpha	20	
Zip	Alpha	10	
Country	Alpha	20	
Phone	Alpha	15	
FAX	Alpha	15	
TaxRate	Numeric		
Contact	Alpha	20	
LasInvoiceDate	Timestamp		

Tabla 4 - Estructura de la tabla customer.db

La sentencia SQL para crear la tabla *customer.db* es la siguiente:

```
create table customer
(CustNo integer not null primary key,
Company varchar(30) not null,
Addr1 varchar(30),
Addr2 varchar(30),
City varchar(15),
State varchar(20),
Zip varchar(10),
Country varchar(20),
Phone varchar(15) not null,
FAX varchar(15),
TaxRate numeric(4,2),
Contact varchar(30),
LasInvoiceDate timestamp)
```

Código Fuente 4 - Creación de la tabla customer

Esta tabla tiene una particularidad: utiliza la tabla *nextcust.db* para obtener el próximo número de cliente. Esta técnica es muy utilizada. Consiste en mantener una tabla (en este caso *nextcust.db*) para generar valores únicos que puedan ser utilizados como clave primaria. Es importante destacar que la correcta utilización de esta técnica implica que las claves primarias generadas no deben tener ninguno significado para las reglas de negocio. Por ejemplo, esta técnica no debería ser utilizada para generar números de pedidos si es que nos interesa mantener su correlatividad.

InterBase ofrece una herramienta llamada generadores (generators) para implementar esta técnica. La sentencia SQL para crear el generador *nextcust* es la siguiente:

```
create generator nextcust
```

Código Fuente 5 - Creación del generador nextcust

Un generador es un mecanismo global a la base de datos que permite generar números enteros mediante la función **gen_id**. La función **gen_id** recibe dos parámetros: el nombre del generador y el incremento o decremento que se debe aplicar para generar el próximo valor. Por ejemplo, **gen_id**(nextcust, 1) incrementa en 1 el generador *nextcust* y devuelve el resultado.

Los generadores generalmente son utilizados en procedimientos almacenados y en disparadores (triggers).

Procedimientos almacenados

Un procedimiento almacenado es un pequeño programa almacenado en la base de datos que puede ser ejecutado en cualquier momento. Los procedimientos almacenados, al igual que los disparadores, utilizan un lenguaje propietario ya que el estándar SQL ANSI 92 no especifica nada acerca de ellos. Generalmente extienden el lenguaje SQL con sentencias de control de flujo como **if...then** y sentencias para proveer funcionalidades adicionales.

Los procedimiento almacenados ofrecen ventajas importantes:

- Rendimiento: al ser ejecutados por el motor de base de datos ofrecen un rendimiento inmejorable ya que no es necesario transportar datos a ninguna parte. Cualquier proceso externo tiene una penalidad de tiempo adicional dada por el transporte de datos.
- Potencia: el lenguaje para procedimientos almacenados es muy potente. Permiten ejecutar operaciones complejas en pocos pasos ya que poseen un conjunto de instrucciones avanzado.
- Centralización: al formar parte de la base de datos los procedimientos almacenados están en un lugar centralizado y pueden ser ejecutados por cualquier aplicación que tenga acceso a la misma. Si un determinado proceso es desarrollo con una aplicación como Delphi, es posible que no esté disponible en todos los lugares que se lo necesite, por ejemplo, el sistema operativo unix. Los procedimientos almacenados están siempre disponibles.

Pero también ofrecen un desventaja importante:

- Esclavitud: los procedimientos almacenados nos esclavizan al motor de base de datos. Una base de datos con muchos procedimientos almacenados es prácticamente imposible de migrar a otro motor. Esto se debe, principalmente, a que los lenguajes de procedimientos almacenados de distintos fabricantes no son compatibles entre sí.

Teniendo en cuenta las ventajas y desventajas mi consejo es no abusar de los procedimientos almacenados y utilizarlos sólo cuando no queda otra alternativa.

Vamos a crear un procedimiento almacenado para obtener el siguiente número de cliente utilizando el generador *nextcust*.

```
set term !!
create procedure proxcliente
(incremento integer)
returns (proximocliente integer) as
begin
  proximocliente = gen_id(nextcust, incremento);
end;
!!
set term ;
```

Código Fuente 6 - Creación del procedimiento almacenado proxcliente

La cláusula **set term** modifica el identificador de fin de sentencia. Por defecto este identificador es el caracter punto y coma que es el mismo utilizado como fin de línea en procedimientos almacenados y disparadores. Es por ello que para crear el procedimiento almacenado *proxcliente* debemos modificar el identificador de fin de sentencia. Utilicé como identificador doble caracter signo de admiración final porque es el utilizado en la documentación de InterBase.

El procedimiento almacenado *proxcliente* tiene un parámetro de entrada y uno de salida, ambos de tipo **integer**. Dentro del procedimiento asignamos el resultado de la función **gen_id** al parámetro de salida *proximocliente*. La función **gen_id** recibe como argumentos el nombre del generador *nextcust* y el parámetro de entrada *incremento.*

Conclusiones

Hemos visto las herramientas visuales que provee InterBase y las utilizamos para conectarnos a un servidor local, crear una base de datos y crear algunas tablas, generadores y procedimientos almacenados.

Quedan muchas sentencias SQL del grupo DDL por explorar pero sería imposible hacerlo en un artículo de esta extensión y de esta serie. Prometo escribir más artículos sobre el SQL de InterBase para mostrarles las sentencias que no vimos.

En el próximo artículo veremos algunas sentencias del grupo DML para trabajar no ya con la creación de una base de datos sino con los datos que almacena.

Para más información

Puede dejarnos cualquier consulta o comentario sobre el artículo, enviando un email a soporte@danysoft.com, estaremos encantados de atenderle.