UNIVERSIDAD EMPRESARIAL SIGLO 21

PROMETEO 2

"Aprovechamiento de Metadatos para la Generación de Procedimientos Almacenados y Clases de Datos"

Por: Nelson David López.

Tesis para obtener el grado de Licenciado en Informática.

Profesores Supervisores:

Jorge Cassi, Fernando Frias.

AGRADECIMIENTOS

Dedico el presente trabajo a DIOS quien hace que todo sea posible, a mi MADRE por los consejos recibidos, su apoyo incondicional y su motivación constante, a mi HERMANA la cual nunca deja de creer en mí y a mi PADRE por estar siempre a mi lado acompañándome y dándome fuerza desde lo espiritual.

Un agradecimiento especial para mis compañeros de trabajo “Tarjeta Naranja”, el cual aportan a mi desarrollo profesional y humano.-

INDICE

Capitulo VI:

Capitulo VII:

Capitulo I

INTRODUCCIÓN

El desarrollo del software y la programación es uno de los pilares fundamentales de la informática y al cual se dedican muchas horas de esfuerzos en empresas, colegios, academias y universidades.

Conforme a la tecnología va avanzando, van apareciendo nuevas soluciones, nuevas formas de programación, nuevos lenguajes y un sinfín de herramientas que intentan realizar el trabajo del desarrollador un poco más fácil1.

En el proceso de desarrollo de un sistema software debemos pasar por varias etapas, las cuales consumen tiempo y recursos.

Si disminuyéramos el tiempo de alguna de estas etapas podríamos utilizarlo en otras etapas más críticas para el éxito del proceso- .

Por lo mencionado anteriormente se propone liberar al programador de tareas rutinarias y propensas a errores de tipeo, como lo son la creación de procedimientos almacenados (Alta, Baja, Modificación y Consultas) y las clases que utilizan estos procedimientos; ofreciendo una herramienta que realice estas tareas de forma automática, proporcionando una notable disminución en el tiempo de desarrollo, debido a que el desarrollador solo debe enfocarse en funcionalidades mas especificas del sistema, dejando a la herramienta las tareas rutinarias antes mencionadas y permitiendo aumentar la productividad y eliminar errores de tipeo que en muchas ocasiones son difíciles de detectar.

Si bien se ha enfocado a la etapa de programación, las ventajas a obtener influyen en el total del proceso de desarrollo de software, posibilitando la disminución de tiempo y recursos que influyen en forma directa en los costos del proceso. Los recursos y el tiempo que no se utilicen en la programación podrían ser utilizados en otras etapas del desarrollo.

El punto de partida de esta idea surge del Proyecto “Prometeo 2”, que consiste en el Desarrollo de una Metodología de Aprovechamiento de Metadatos de los Diccionarios de Datos de Bases de Datos Relaciónales para lograr un Generador de Sentencias SQL.

[1] http://www.microsoft.com/spanish/MSDN/estudiantes/desarrollo/default.mspx -junio/2008

[2] ACC- Agencia Córdoba Ciencia. –Desarrollo de una metodología de aprovechamiento de Metadatos de los diccionarios de Datos de Bases de Datos Relaciónales para lograr un generador de sentencias SQL designado PROMETEO. Ref: 5680 Director: Martínez Spessot, Cesar. http://www.agenciacordobaciencia.cba.gov.ar/proyectos2005/financiados.htm - junio/2008

Del proyecto anteriormente nombrado no solo obtenemos la idea sino aportaciones que ayudaran a la conclusión de esta propuesta.

Prometeo esta desarrollado en una primera etapa, para las tecnologías vigentes PHP y SQL PostgreSQL, teniendo como objetivo propósito ampliarse hacia otros diccionarios de base de datos Comerciales como lo son Oracle 10g y SQL Server 2005.

Para el desarrollo de la herramienta se ha visto la necesidad de expandir los horizontes; en este momento a ASP.NET Y SQL SERVER, basado en el avance de la tecnología correspondiente a la plataforma Web y al creciente uso en el mercado de estas tecnologías. Por lo cual podemos incidir en que el proyecto a desarrollar es una actualización y continuación de “Prometeo”.

ANTECEDENTES

El siguiente trabajo toma como antecedente a Prometeo presentado en el ASIS [3] 2005, conocido como: "TecnoDB: Desarrollo de una Metodología de Aprovechamiento de Metadatos de los Diccionarios de Datos de Bases de Datos Relaciónales para Lograr un Generador de Sentencias SQL – PROMETEO", fue consolidándose en el avance de la construcción del software, con el propósito de aplicar una Heurística presentada en ese trabajo, y así generar la mayor cantidad de sentencias SQL.

Prometeo hace un análisis de los metadatos y los traduce en asistencia al usuario para construir sentencias SQL y construye por si mismo estas sentencias almacenándolas en la base de datos en un paso posterior a la construcción.

Como un primer propósito esta herramienta fue creada para trabajar con bases de datos PostgreSQL, teniendo como ambición expandirse hacia otras tecnologías, lo cual motivo al desarrollo de Prometeo 2.

[3] ASIS: Simposio Argentino de Sistemas de Información

JUSTIFICACIÓN

Justificación del objetivo del producto:

Hoy en día, cualquier organización o unidad organizacional de nuestro medio necesita un producto software como herramienta que ayude y optimice la gestión de los eventos de negocio que justifican su existencia.

Para ello, existen organizaciones denominadas SOFTWARE FACTORY cuyo fin es la realización de estos productos de software, brindando predicibilidad y robustez en la administración de los procesos de negocios de otras organizaciones, ayudando a los gestores de las mismas cumplir con el objetivo planteado por ellas.

Atendiendo el papel que cumplen estas organizaciones, nos hemos propuesto el desarrollo de una herramienta que ayude en sus actividades, para unificar (estandarizar) e industrializar el producto de software, y con ello, incrementar la productividad, automatización y control de sus proyectos informáticos en forma ágil y segura.

Si analizamos una aplicación software que administra los procesos de negocios de una organización, veremos que aproximadamente, el cincuenta - sesenta por ciento de su funcionalidad radica en la gestión de los parámetros o tablas ambientales de dicha aplicación

Y es aquí donde radican los cimientos de nuestra herramienta PROMETEO 2, permitiendo generar automáticamente esta funcionalidad en solo una fracción del tiempo, que si bien es necesaria, no es un elemento esencial para definir el producto final de software.

De esta automatización, nos permite con ello liberar y utilizar recursos físicos, tiempos y capital humano en pos de desarrollo de la funcionalidad que dará valor agregado al negocio al cual estará destinado.

Como mencionamos anteriormente, la idea con este proyecto es ayudar en la industrialización del desarrollo de un producto de software, brindando la sistematización de actividades que son rutinarias, que no hacen a la esencia de un producto final de software y de este modo, reducir tiempos, recursos físicos, capital humano, costos, falencias del SW; por ende contribuir al desarrollo de un producto de software robusto y confiable.

Justificación de la Tecnología empleada:

Ahora bien, como mencionamos anteriormente, esta herramienta provee la automatización del desarrollo de gestión de las tablas ambientales o para métricas en un producto de software, esto es, crear los Stores Procedures quienes realizaran las Alta / Baja / Modificación / Consulta / Consulta Compleja; como así también las clases quienes ejecutaran dichos SP en la capa de servicio de la aplicación.

Para ambas, nos hemos basado en la tecnología Microsoft. Por un lado, en la herramienta SQL SERVER para crear los SP en la semántica y sintaxis de TRANSACT SQL, y por otro lado, MS Visual Studio 2003 – 2005, en particular, en el lenguaje Visual Basic de este IDE de desarrollo.

Nos orientamos en esta herramienta, básicamente, por la masividad que ambas representan. Por su historia y su uso, su rapidez en la gestión y dinamismo en la configuración.

Cuando hablamos de masividad, nos referimos a la popularidad de ellas, traducida en la cantidad de personas y empresas que la manejan y conocen, que conocen su sintaxis y semántica como así también su correspondiente IDE. De hecho, la orientación del mercado de acción de este producto, está centrado en aquellos desarrolladores independientes, pequeñas y medianas empresas dedicadas a diseñar y desarrollar productos software que se sirven de estos productos para llegar a cumplir su objetivo.

De ahí radica su poder, y por ende, el aprovechamiento de las mismas para producir todos los productos resultantes de la manipulación de Prometeo 2.

Algunas ventajas de SQL SERVER:

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- Además permite administrar información de otros servidores de datos.

Ventajas del Framework de VB.NET:

- Código administrado: controla los recusos del sistema para que la aplicación se ejecute correctamente.
- Compilación just-in-time: El compilador JIT incluido en el Framework compila el código intermedio (MSIL) generando el código máquina propio de la plataforma.
- Garbage collector: sistema automático de administración de memoria denominado recolector de basura (garbage collector).
- Seguridad de acceso al código: Se puede especificar que una pieza de código tenga permisos de lectura de archivos pero no de escritura.
- Despliegue: Por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas.

OBJETIVO

Desarrollo de una herramienta, que basado en el aprovechamiento de los metadatos del diccionario de datos de SQL SERVER [4], genere de forma automática los procedimientos almacenados (Alta, Baja y Modificación), como así también Consultas Simples con "Select", Consultas multitablas de las tablas relacionadas con Constraints Referenciales y las Clases que utilizan estos procedimientos almacenados.

Objetivos Específicos

- Disminuir el tiempo desarrollo.
- Eliminar errores de programación en procedimientos almacenados y de clases que los acceden.

LIMITE

Desde la conexión a la base de datos hasta la generación de los archivos .sql y .vb que administrarán las tablas parametricas o ambientales de un dominio en particular.

ALCANCE

- Conexión a la Base de datos.
- Visualización de la estructura de las tablas, tipos de datos y constrains de la base de datos seleccionada.
- Generación automática de Consultas Simples.
- Generación automática de Consultas Multitablas
- Generación automática de Procedimientos Almacenados.
    - Alta
    - Baja
    - Modificación

[4] [4] El proyecto contemplara las siguientes versiones de SQL SERVER: 2000 y 2005.

- Generación automática de Clases que acceden a los Procedimientos Almacenados.
- Almacenamiento automático de los Procedimientos Almacenados en la Base de Datos seleccionada.

BENEFICIOS DEL PROYECTO

Beneficios Económicos

Este beneficio es muy claro ya que al automatizar parte del trabajo en el desarrollo de cualquier producto software, ahorramos recursos tanto en tiempo como mano de obra, y reducimos así los costos en la resolución de todo desarrollo.

Beneficios Tecnológicos

Este al igual que el económico es el beneficio mas claro de la ejecución del proyecto, puesto que permite insertar al mercado una herramienta mas acotada, robusta y de fácil uso, pero con características mas puntuales y directas que las que se encuentran hoy en día. Estas características la hacen mucho más económica que sus principales competidoras.

Capitulo II

MARCO REFERENCIAL

Antes de introducirnos en los aspectos a tener en cuenta en el desarrollo del proyecto, se hará una breve introducción sobre conceptos, terminologías y aspectos generales para poder de esta manera ambientarnos en el tema en cuestión y eliminar dudas que puedan surgir, para luego enfocarnos en conocimientos específicos que hacen al proyecto.

El Marco Referencial estará constituido por una breve explicación de que es una Base de Datos?, para luego enfocarnos en SQL SERVER (SGBD Seleccionado5), sobre todo hablaremos de los Objetos que lo componen y una breve explicación de cada unos de ellos, por ultimo nos introduciremos en el Diccionario de Datos del cual obtendremos los Metadatos usado para nuestra construcción.

5 Sistema de Gestión de Base de Datos.

BASE DE DATOS

En la mayoría de las aplicaciones software es indispensable el almacenamiento de la información para posteriormente ser manipulada.

Una base de datos es más que un lugar donde se almacenan los datos, sino que también podemos almacenar una serie de objetos, procedimientos y reglas, que garantizan la fiabilidad e integridad de los datos.

Las operaciones de acceso a datos mas frecuentes pueden ser almacenadas en la base de datos, como procedimientos almacenados, por lo que no solamente almacenamos datos sino que también almacenamos mecanismos requeridos para trabajar esta información.

En **SQLSERVER** las credenciales del usuario también son almacenadas en la base de datos, estas credenciales realizan la tarea de gestionar los permisos de los usuarios que pueden tener sobre los datos. Las aplicaciones software que realizamos no trabajan directamente con nuestros datos, sino que necesitan de un enlace, que reciban las órdenes de la aplicación y trabajen sobre estos datos, este enlace o conexión recibe el nombre de Servidor de Datos.

Cuando creamos una base de datos, podemos nombrar a la misma con el nombre que deseemos, y el lugar de destino que queramos. SQL SERVER se encargara de crear la estructura lógica con los archivos necesarios, estos son el **Archivo de Datos** y el **Archivo de Transacciones** (Data.MDF, Log.LDF).

- El Archivo de Datos: tiene almacenada la información y una serie de objetos que trabajan con esta información
- El Archivo de Transacciones: Este archivo garantiza la integridad de la base de datos, ya que almacena las modificaciones debido a la explotación de la base de datos.

Sistemas administradores de bases de datos relacionales

Un Sistema Administrador de Bases de Datos (SABD o también conocido en inglés como DBMS de Database Management System) es un conjunto de programas que permiten a los usuarios crear y mantener bases de datos, entre las principales características de un SABD se tienen: control de redundancia, restricción de los accesos no autorizados, suministro de interfaces para los usuarios, representación de vínculos o relaciones entre los datos, cumplimiento de las restricciones de integridad, respaldo y recuperación, entre otros.

Los usuarios de las bases de datos utilizan los componentes lógicos del manejador (comúnmente llamados objetos de bases de datos), dejando de lado la representación física, la cual es transparente para ellos. Entre los componentes lógicos que manejan los usuarios se encuentran las tablas, los procedimientos almacenados, las vistas, los disparadores, entre otros que pasaremos a nombrar y describir mas adelante.

SQL SERVER

Microsoft SQL SERVER es un sistema de gestión de base de datos basado en el lenguaje Transact-SQL (lenguaje de programación que proporciona SQL Server para ampliar SQL con los elementos característicos de los lenguajes de programación: variables, sentencias de control de flujo, bucles, etc.), el cual se pasara a detallar mas adelante a lo largo del Marco Referencial. Para hacer mas claro esta definición debemos saber que SQL SERVER es un sistema de manejo de base de datos relacional6 producido por Microsoft.

SQL SERVER es un conjunto de objetos eficientemente almacenados. Los objetos donde se almacena la información se denominan tablas, y éstas a su vez están compuestas de filas y columnas. En el centro de SQL Server está el motor de SQL Server, el cual procesa los comandos de la base de datos. Los procesos se ejecutan dentro del sistema operativo y entienden únicamente de conexiones y de sentencias SQL.

Transact-SQL es el lenguaje que utiliza SQL Server para poder enviar peticiones tanto de consultas, inserciones, modificaciones, y de borrado a las tablas, así como otras peticiones que el usuario necesite sobre los datos. En definitiva, es un lenguaje que utiliza SQL Server para poder gestionar los datos que contienen las tablas.

Transact-SQL envía peticiones entre el cliente y el servidor. Es un lenguaje exclusivo de SQL Server, pero basado en el lenguaje SQL estándar, utilizado por casi todos los tipos de bases de datos relacionales que existen. SQL Server otorga a los administradores una herramienta potencialmente robusta, provista de las herramientas suficientes que le permiten mantener un óptimo nivel de seguridad en la utilización de los recursos del sistema y de la base de datos.-

6 Una base de datos relacional es una base de datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores, y en donde todas las operaciones de la base de datos operan sobre estas tablas. Estas bases de datos son percibidas por los usuarios como una colección de relaciones normalizadas de diversos grados que varían con el tiempo

OBJETOS DE SQL SERVER

A continuación se exponen los objetos principales que componen la estructura lógica de la base de datos SQL SERVER, para tener un conocimiento general de cómo se gestionan los datos.

Los Objetos que contiene la base de datos son los siguientes:

Tablas: Las bases de datos están formadas por bloques de información básicos, estos bloques reciben el nombre de tablas, lo que antiguamente se denominaban ficheros o archivos.

Una tabla, es un conjunto de información con características comunes. Es decir, almacena información sobre un concepto en común. Estas tablas (Imagen 1), están compuestas de registros. Un registro es cada uno de los elementos de información de la tabla. A su vez, cada registro esta formado por una unidades fundamentales denominadas campos. Un campo es la unidad de información que interesa almacenar para cada registro.

Las principales tareas o actividades que se realizan sobre las tablas son las siguientes:

- Añadir información.
- Eliminar información.
- Modificar y actualizar la información.
- Recoger información y mostrarla.

Generalmente para la realización de estas tareas se utiliza el lenguaje SQL (Structured Query Language), que es el lenguaje estándar para gestionar bases de datos.

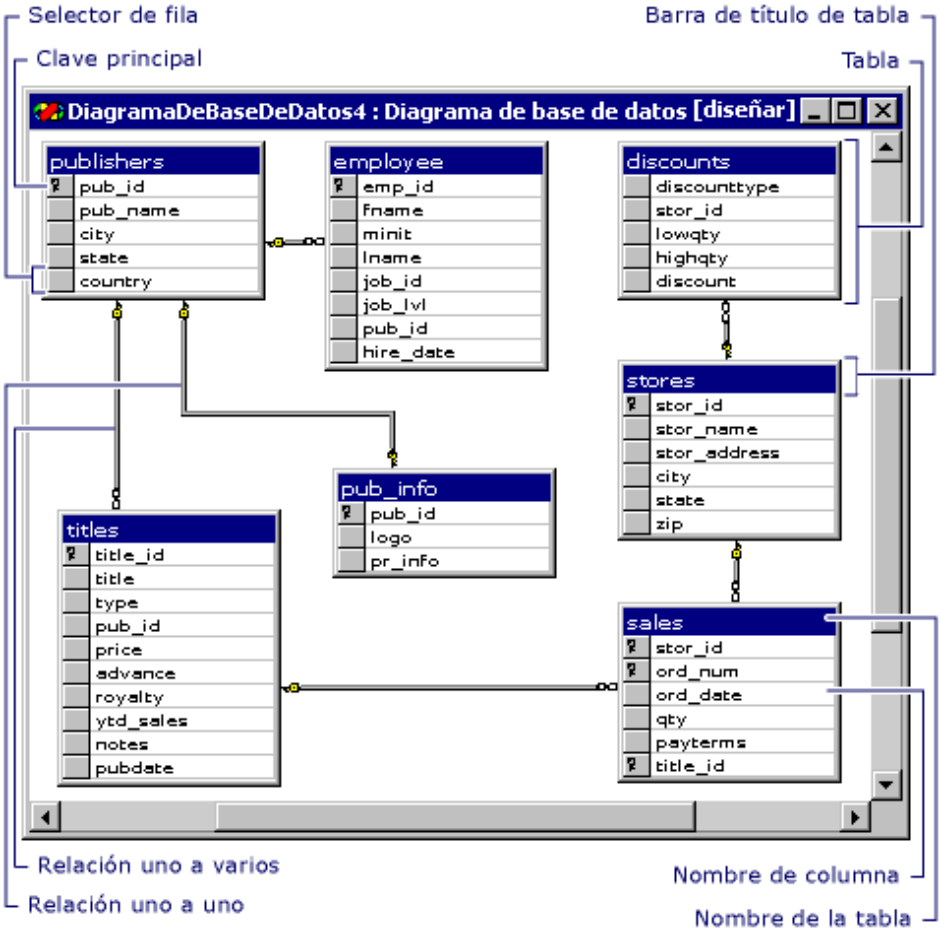


Imagen 1: Diagrama de las tablas que componen una Base de Datos.

Vistas: Son una especie de tablas virtuales; es decir no existen físicamente sino que forman mediante la selección y/o filtrado de los componentes de otras tablas, una vista puede ser definida en base a una lista previa. Esto significa que pueden crearse dependencia entre las vistas. Cuando una vista es definida en base a otra, se dice que es dependiente de esta por lo tanto, se suprimirá automáticamente la vista dependiente si se suprime la vista original. La eliminación de una tabla provoca también la eliminación automática de todas las vistas que se hayan definido haciendo referencia a ella.

Las vistas son una forma lógica de ver los datos físicos ubicados en tablas. Cuando creamos una vista, seleccionamos un formato que incluye datos que pueden ser tomados de una o más tablas. La vista queda almacenada en forma permanente, si bien los datos grabados permanecen inalterados en las tablas correspondientes. Una vista sólo es una ventana a los datos almacenados.

En resumen la “vista” es un objeto definido por una consulta. Similar a tabla, la vista muestra un conjunto de columnas y filas de datos con un nombre, sin embargo, en la vista no existen datos, estos son obtenidos desde las tablas subyacentes a la consulta. De esta forma si la información cambia en las tablas,

estos cambios también serán observados desde la vista. Fundamental emplean para mostrar la información relevante para el usuario y ocultar la complejidad de las consultas.

Las vistas permiten:

- ocultar información: permitiendo el acceso a algunos datos y manteniendo oculto el resto de la información que no se incluye en la vista. El usuario opera con los datos de una vista como si se tratara de una tabla, pudiendo modificar tales datos.
- simplificar la administración de los permisos de usuario: se pueden dar al usuario permisos para que solamente pueda acceder a los datos a través de vistas, en lugar de concederle permisos para acceder a ciertos campos, así se protegen las tablas base de cambios en su estructura.
- mejorar el rendimiento: se puede evitar tipear instrucciones repetidamente almacenando en una vista el resultado de una consulta compleja que incluya información de varias tablas.

Podemos crear vistas con: un subconjunto de registros y campos de una tabla; una unión de varias tablas; una combinación de varias tablas; un resumen estadístico de una tabla; un subconjunto de otra vista, combinación de vistas y tablas.

Una vista se define usando un "select".

La sintaxis básica parcial para crear una vista es la siguiente:

```
create view NOMBREVISTA as
  SENTENCIASSELECT
   from TABLA;
```

El contenido de una vista se muestra con un "select":

```
 select *from NOMBREVISTA;
```

Índices: Es fácil encontrar bases de datos con tablas cuyo tamaño aumenta con facilidad pudiendo contener hasta millones de registros. En estos casos el acceso a determinada información, puede ser lento y costoso. Habrá operaciones de consulta que obligue a recorrer la tabla entera, desde su primer registro hasta el último, esto repercute directamente en nuestra aplicación, convirtiéndola en una aplicación lenta y pesada. De ahí la importancia del diseño de la base de datos.

Para acelerar el acceso a la información contamos con los índices, un índice almacena una serie de claves que permite al servidor acelerar sus consultas. El índice no sólo sirve para aumentar la velocidad, sirve además para fijar un orden en nuestros registros, tener registros únicos, de modo que cuando se modifica el contenido de una tabla en la cual afecta algún índice, SQL SERVER debe modificar los datos de la tabla y los índices que afectan a la misma.

Los índices de SQL Server son similares a los índices de un libro que nos permiten llegar rápidamente a las páginas deseadas sin necesidad de pasar hoja por hoja, de forma similar los índices de una tabla nos permitirán buscar información rápidamente sin necesidad de recorrer registro por registro por toda la tabla. Un índice contiene valores y punteros a las filas donde estos valores se encuentran.

Constraints o Restricciones: Mediante las restricciones ponemos limitaciones a los datos que se van a introducir en la base de datos. Determinamos que datos son válidos para insertar en la columna de una tabla.

Tenemos las restricciones Unique, Default y Check que fuerzan la integridad de identidad, dominio y la marcada por usuario. Y por otro lado contamos con las restricciones Primary Key y Foreign Key para garantizar la integridad referencial en las relaciones.

Unique constraint

Una constraint unique protege a una o más columnas de una tabla, asegurando que no hay dos filas que contengan información duplicada en las columnas aseguradas.

Check constraints

Este tipo de constraints son usadas para asegurar reglas simples de negocio sobre el contenido de los datos en las tablas.

Los check pueden referenciar a otras columnas en la fila que está siendo chequeada, pero no pueden referenciar a otras filas o a otras tablas.

Constraint NOT NULL

Se aplica a una columna y requiere valores para la columna que protege. Por defecto, SQL pone a NULL las columnas no introducidas de la tabla.

Primary Key

La cláusula PRIMARY KEY se utiliza para definir la columna como clave principal de la tabla. Esto supone que la columna no puede contener valores nulos ni pueden haber valores duplicados en esa columna, es decir que dos filas no pueden tener el mismo valor en esa columna.

En una tabla no puede haber varias claves principales, por lo que no podemos incluir la cláusula PRIMARY KEY más de una vez, en caso contrario la sentencia da un error. No hay que confundir la definición de varias claves principales con la definición de una clave principal compuesta por varias columnas, esto último sí está permitido y se define con una restricción de tipo 2.

Foreign keys

Una foreign key protege una o más columnas de una tabla, asegurando que cada valor de la fila es, o bien nulo en su conjunto, o bien apunta a un valor de una clave única o primaria. Mencionan dos términos: parent table o tabla maestra y child table o tabla detalle.

Tabla maestra. Tabla referenciada. Es la que tiene la primary o la unique

Tabla detalle. La tabla donde se encuentra la referencia. L a que es chequeada para garantizar que sus valores se corresponden con los de la tabla maestra.

Tipo de Integridad	Tipo de Constraint
Controlar Rango de valores sobre las Columnas	DEFAULT
	CHECK
Unicidad de registros y valores únicos	PRIMARY KEY
	UNIQUE
Referencial	FOREIGN KEY
	CHECK

Triggers: Llamado también Disparador o Desencadenador es un Proceso que se dispara cuando se provoca un determinado evento. Un Desencadenador es un Procedimiento Almacenado especial el cual se invoca automáticamente ante una operación de Insert, Update o Delete sobre una tabla. Un Desencadenador puede consultar otras tablas y puede incluir complejas instrucciones SQL, se emplean para mantener la integridad referencial, preservando las relaciones definidas entre las tablas cuando se ingresa o borra registros de aquellas tablas. Los Valores Predeterminados especifican el valor que SQL Server insertará en una columna cuando el usuario no ingresa un dato específico. Por ejemplo, si se desconoce el apellido materno de un empleado SQL Server podría incluir automáticamente la cadena NN para identificar este campo.

Son usados para mejorar la administración de la Base de datos, sin necesidad de contar con que el usuario ejecute la sentencia de SQL. Además, pueden generar valores de columnas, previene errores de datos, sincroniza tablas, modifica valores de una vista, etc.

Permite implementar programas basados en paradigma lógico (sistemas expertos, deducción).

La estructura básica de un *trigger* es:[7]

- *Llamada de activación:* es la sentencia que permite "disparar" el código a ejecutar.
- *Restricción:* es la condición necesaria para realizar el código. Esta restricción puede ser de tipo condicional o de tipo nulidad.
- *Acción a ejecutar:* es la secuencia de instrucciones a ejecutar una vez que se han cumplido las condiciones iniciales.

Un sencillo ejemplo sería crear un trigger para insertar un pedido de algún producto cuando la cantidad de éste en nuestro almacén sea inferior a un valor dado:

```
BEFORE UPDATE ON tabla_almacen
FOR ALL records
   IF :NEW.producto < 100 THEN
      INSERT INTO tabla_pedidos(producto) VALUES ('1000');
   END IF;
SELECT DBO.POLVE.TEST
 END
```

Usuarios: En las bases de datos podemos añadir tantos usuarios como necesitemos. Y otorgarle los permisos que deseemos. De este modo, podemos limitar a un usuario para que sólo tenga acceso a unas determinadas tablas, y fijar las operaciones que puede realizar sobre las mismas.

Una de las tareas comunes al administrar SQL Server es permitir el acceso a bases de datos y la asignación de permisos o restricciones sobre los objetos que conforman una base de datos.

SQL Server 2000 permite trabajar a nivel de **Roles y Usuarios**.

Un ***rol*** es un conjunto de derechos asignados, los cuales se convierten en una gran alternativa para agrupar un conjunto de permisos, de tal forma que cuando se incorpore un nuevo usuario a la base de datos, ya no se le tiene que dar permiso por permiso por cada uno de los objetos que requiera emplear, sino mas bien su cuenta de usuario es agregada al rol, y si al rol tiene que asignársele acceso sobre un nuevo elemento automáticamente el permiso o la restricción afectará a los usuarios que pertenezcan a un rol.

Los ***usuarios*** representan los usuarios que tienen acceso a la base de datos y están mapeados a un Inicio de sesión, aunque pueden tener diferente identificador, por ejemplo el Inicio de sesión puede tener como nombre Dlopez pero al definir un Usuario podemos usar David. Después de que se crearon los Inicios de sesión para conectarse a SQL Server, se deben definir los accesos a las bases de datos requeridas, para

7 http://es.wikipedia.org/wiki/Disparador_(base_de_datos) /abril- 2009

ello es necesario definir los roles de los usuarios en cada BD, estos usuarios permitirán controlar el acceso a los distintos objetos incluyendo los datos que estos contienen.

Además de los Inicios de sesión y usuarios SQL Server brinda un conjunto de roles por servidor y por base de datos que son derechos predefinidos que podrán especificarse por cada usuario de ser necesario. También es posible crear roles personalizados.

Los roles son los siguientes8:

Roles por Servidor	
Rol	**Descripción**
Dbcreator	Crea y modifica bases de datos.
Diskadmin	Administra los archivos de datos.
Processadmin	Administra los procesos de SQL Server.
Serveradmin	Opciones de configuración del servidor.
SecurityAdmin	Administra los Inicios de sesión.
Roles por Base de Datos	
Rol	**Descripción**
Public	Mantiene los permisos En forma predeterminada para todos los usuarios.
db_owner	Realiza cualquier actividad en la BD
db_accessadmin	Agrega o retira usuarios y/o roles
db_ddladmin	Agrega, modifica o elimina objetos
db_backupoperator	Backup y Restore de la base de datos
db_datareader	Lee información desde cualquier tabla
db_denydatareader	No puede leer la información

8 Manual de SQL 2000 Server.pdf

Esquemas: Desde SQL Server 2005, cada objeto pertenece a un esquema de base de datos. Un esquema de base de datos es un espacio de nombres separado de un usuario de base de datos.

Para entender mejor este concepto, vamos a revisar primero SQL Server 2000. Cuando se trata de eliminar un usuario que posee objetos de base de datos, es necesario asignar todos los objetos pertenecientes a ese usuario a otro utilizando sp_changeobjctowner. Si no reasignar los objetos en primer lugar, tendrá que soltar todos los objetos que pertenecen al usuario antes de que pueda soltar el usuario.

En SQL Server 2000, los objetos están estrechamente vinculados a los usuarios. Esto significa que los usuarios pueden tener dos objetos con el mismo nombre, que puede dar lugar a confusión en el entorno de desarrollo.

SQL Server 2005 ofrece una solución para este problema: un método llamado un esquema. Pensar en un esquema como un contenedor que tiene límites. Este contenedor tiene objetos. En lugar de acceder a la tabla por objeto propietario, como lo hizo en SQL Server 2000, puede acceder a ella por el esquema: Un esquema se puede considerar como un contenedor de objetos. Los esquemas se pueden crear y modificar en una base de datos, y a los usuarios se les puede conceder acceso a un esquema.

Los usuarios pueden crear esquemas y puede poseer y pertenecen a los esquemas. Los beneficios de los esquemas de SQL Server 2005 son las siguientes:

- Se pueden administrar los permisos sobre esquemas y sobre elementos que se pueden proteger con mayor precisión que en las versiones anteriores.
- La propiedad de los esquemas y de los elementos que se pueden proteger con ámbito de esquema es transferible.
- Es posible mover objetos entre esquemas.
- Varios usuarios de base de datos pueden compartir un mismo esquema predeterminado.
- Cualquier entidad de seguridad de base de datos puede ser propietaria de un esquema. Esto incluye funciones y funciones de aplicación.
- Es posible eliminar un usuario de base de datos sin necesidad de eliminar objetos en un esquema correspondiente.
- Compartir por defecto esquemas permiten a los desarrolladores almacenar objetos compartidos en un esquema creado específicamente para una aplicación específica, más que en el esquema de DBO, que es la práctica general en SQL Server 2000.
- Permisos en los esquemas y los objetos que figuran en los esquemas se pueden manejar con un mayor grado de granularidad[9] que en versiones anteriores de SQL Server.

A partir de SQL Server 2005, los esquemas son entidades explícitas reflejadas en los metadatos. En consecuencia, los esquemas sólo pueden tener un propietario, pero un solo usuario puede tener uno o

9 Granularidad es el nivel de detalle al cual se identifican los componentes de un documento o una estructura de una base de datos

varios esquemas. Esta compleja relación no se refleja en las tablas del sistema de SQL Server 2000, por lo que SQL Server 2005 introduce nuevas vistas de catálogo que reflejan los nuevos metadatos con precisión

Valores Predeterminados: Especifican el valor que SQL Server insertará en una columna cuando el usuario no ingresa un dato específico. Por ejemplo, si se desconoce el apellido materno de un empleado SQL Server podría incluir automáticamente la cadena NN para identificar este campo.

Reglas: Son objetos que especifican los valores aceptables que pueden ser ingresados dentro de una columna particular. Las Reglas son asociadas a una columna o a un tipo de dato definido por el usuario. Una columna o un Tipo de dato puede tener solamente una Regla asociada con el.

Procedimientos Almacenados:

Un servidor de base de datos no sólo puede manipular información de sus tablas, sino que tiene la capacidad de interpretar código en un lenguaje SQL (Sctrutured Query Language) para la realización de una serie de funciones u operaciones. Este código SQL se puede encapsular en un procedimiento o varios y se almacena en la propia base de datos. Por lo tanto podemos escribir procedimientos en SQL para realizar las tareas deseadas sobre nuestra base de datos y almacenarlos, para más adelante con una sencilla llamada se ejecuten nuestras instrucciones.

Estos procedimientos son llamados, **stored procedures** y son una colección de sentencias del Transact-SQL las cuales organizadas lógicamente resuelven algunas de las operaciones transaccionales que requieren los usuarios, estos procedimientos se almacenan en la base de datos. Los procedimientos almacenados soportan el empleo de variables declaradas por el usuario, sentencias para toma de decisiones entre otras características.

En SQL Server existen 5 tipos de procedimientos almacenados:

- **Procedimientos del sistema**, son los que se encuentran almacenados en la base de datos **master** y algunas en las bases de datos de usuario, estos procedimientos almacenados brindan información acerca de los datos y características del servidor. En el nombre usan como prefijo **sp_.**

- **Procedimientos locales**, son los procedimientos almacenados en una base de datos.

- **Procedimientos temporales**, son procedimientos locales y sus nombres empiezan con los prefijos # o ##, dependiendo si se desea que sea un procedimiento global a todas las conexiones o local a la conexión que lo define.

• **Procedimientos remotos**, son procedimientos almacenados en servidores distribuidos.

• **Procedimientos extendidos**, son aquellos que nos permiten aprovechar las funcionalidades de otras librerías externas a SQL Server. Estos procedimientos usan el prefijo **xp_** y se encuentran en la base de datos master.

Entre las principales características de un procedimiento almacenado podemos mencionar:

• Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al lote o al procedimiento que realiza la llamada.

• Contener instrucciones de programación que realicen operaciones en la base de datos, incluidas las llamadas a otros procedimientos.

• Devolver un valor de estado que indica si la operación se ha realizado correctamente o habido un error (y el motivo del mismo).

• Permiten una ejecución más rápida, ya que los procedimientos son analizados y optimizados en el momento de su creación, y es posible utilizar una versión del procedimiento que se encuentra en la memoria después de que se ejecute por primera vez.

• Pueden reducir el tráfico de red.

• Pueden utilizarse como mecanismo de seguridad, ya que se puede conceder permisos a los usuarios para ejecutar un procedimiento almacenado, incluso si no cuentan con permiso para ejecutar directamente las instrucciones del procedimiento.

Estructura simple de un procedimiento almacenado:

```
CREATE PROC   procedure_name
[                    {@parameter data_type}
]
AS sql_statement
```

1. Instrucción de creación del procedimiento.
2. nombre del procedimiento.
3. Parámetros del procedimiento.
4. Instrucciones SQL

Uno de los principales motivos por el cual se guarda información, es por que posteriormente la vamos a consultar o manipular, una de las principales razones por las cuales las bases de datos relacionales lograron gran aceptación fue por la forma tan sencilla de lograr acceder a los datos. Y como parte de estas facilidades para poder realizar consultas, encontramos a la sentencia SELECT.

Select

Recupera información de la Base de Datos y permite la selección de una o más filas o columnas de una o muchas tablas. La sintaxis completa de la instrucción SELECT es compleja, sus cláusulas principales pueden ser resumidas de la siguiente manera.

```
SELECT lista_cols
FROM tabla_origen
[WHERE condición]
```

1. SELECT Recupera o muestra las columnas.
2. FROM Determina la tabla o tablas de donde se muestra la información.
3. WHERE Establece un criterio de selección de filas

Insert

Utilice la sentencia INSERT para agregar registros a una tabla.

La sintaxis reducida puede ser:

```
INSERT [INTO] <Nombre de la Tabla> VALUES (Valor1, ....)
GO
```

Es importante recordar que si el valor que intenta agregar a una de las columnas no cumple con alguno de los constraints establecidos la operación abortará inmediatamente.

También es posible agregar múltiples filas a través del siguiente formato:

```
INSERT [INTO] <Nombre de la Tabla>
SELECT <lista de campos> FROM <Tabla>
```

Update

Esta sentencia nos permite modificar la información de las tablas.

La sintaxis reducida puede ser:

```
UPDATE <Nombre de la Tabla>
SET <columna> = <Nuevo Valor>
[WHERE <condición>]
GO
```

Delete

Las instrucciones DELETE y TRUNCATE TABLE remueven filas de una tabla.

La sintaxis de DELETE puede ser:

```
DELETE <Nombre de la tabla>
[WHERE <Condición>]
```

J o i n s

Para muchas de las consultas que los usuarios realizan sobre los datos almacenados en nuestra base de datos es necesario extraer información de más de una tabla, para ello es necesario emplear los **JOINS** que representan una operación producir un conjunto de resultados que incorporen filas y columnas de las tablas referidas en la consulta, esto lo hace basándose en columnas comunes a las tablas. Cuando se ejecutan los **JOIN**, SQL Server compara los valores de las columnas especificadas fila por fila entonces usa los resultados de la comparación para combinar los valores que califican como nuevas filas.

```
SELECT <lista de columnas>
FROM <tabla o vista >
[INNER JOIN
<Tabla o Vista > ON <condición>
```

La lista de columnas puede incluir campos de diferentes tablas.

JOIN especifica las tablas involucradas en la consulta. **ON**, establece la condición de unión de las tablas, a través de campos comunes.

Es bueno aclarar el porque es beneficioso el uso de los procedimientos almacenados en aplicaciones que procesan datos. Las razones son las siguientes:

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. Los procedimientos almacenados son analizados y optimizados en el momento de su creación, a diferencia de las instrucciones Transact-SQL, que deben ser analizadas, compiladas y optimizadas cada vez que son enviadas por el cliente. Además, el motor de SQL Server es capaz de reutilizar el plan de ejecución del procedimiento almacenado que se encuentra en la memoria (caché de procedimientos) después de haberse ejecutado una primera vez.

Potencia: el lenguaje para procedimientos almacenados es muy potente. Permiten ejecutar operaciones complejas en pocos pasos ya que poseen un conjunto de instrucciones avanzadas.

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.

Reducción del trafico de red: Una sentencia formada por decenas, cientos o incluso miles de líneas de código Transact-SQL puede escribirse como un procedimiento almacenado en el servidor y ejecutarse simplemente mediante el nombre de dicho procedimiento, en lugar de enviar todas las líneas de código por la red desde el cliente hasta el servidor (ésta reducción del tráfico de red será especialmente significativa en redes no muy veloces, como por ejemplo, algunas redes WAN).

Seguridad: Los procedimientos almacenados facilitan algunas tareas de administración de seguridad y asignación de permisos. Por ejemplo, se puede conceder permiso a un usuario para ejecutar un determinado procedimiento almacenado, aunque el usuario no disponga de los permisos necesarios sobre los objetos afectados por las acciones individuales de dicho procedimiento.

Encapsulación: Los procedimientos almacenados encapsulan gran parte de la lógica de los datos a las aplicaciones que los utilizan. Por ejemplo, una aplicación puede llamar al procedimiento almacenado spEliminarProveedor sin conocer cómo funciona internamente éste proceso (transacciones e instrucciones Transact-SQL utilizadas, tablas afectadas, etc.)

TRANSACT – SQL

SQL es un lenguaje de consulta para los sistemas de bases de datos relaciónales, pero que no posee la potencia de los lenguajes de programación. No permite el uso de variables, estructuras de control de flujo, bucles y demás elementos característicos de la programación. No es de extrañar, SQL es un lenguaje de consulta, no un lenguaje de programación.

Sin embargo, SQL es la herramienta ideal para trabajar con bases de datos. Cuando se desea realizar una aplicación completa para el manejo de una base de datos relacional, resulta necesario utilizar alguna herramienta que soporte la capacidad de consulta del SQL y la versatilidad de los lenguajes de programación tradicionales. Transact SQL es el lenguaje de programación que proporciona Microsoft SQL Server para extender el SQL estándar con otro tipo de instrucciones y elementos propios de los lenguajes de programación.

Con Transact SQL vamos a poder programar las unidades de programa de la base de datos SQL Server, están son:

- Procedimientos almacenados
- Funciones
- Triggers
- Scripts

Transact-SQL fue diseñado para aumentar la potencia de SQL y para minimizar, si no eliminar, las ocasiones en las que los usuarios deben recurrir a un lenguaje de programación para llevar a cabo las tareas deseadas. Transact-SQL va más allá de las normas ISO y de las diversas versiones comerciales de SQL.

Transact-SQL (T-SQL) es el lenguaje de programación del SQL Sever, a través de el podemos realizar muchas operaciones relacionadas con el SQL sin tener que volver a pasar por código ASP o VB, esto simplificará vuestro código y ganará en rapidez dado que el T-SQL se ejecuta dentro del SQL Sever y es código compilado, se compila la primera vez que se ejecuta el Stored Procedure.

El T-SQL se puede utilizar desde multitud de aplicaciones y desde diferentes

Lenguajes de programación:

- Desde Visual Basic.
- Desde Visual C++
- Desde Active Server Page (ASP), Etc.

Transact –SQL no solo sirve para realizar consultas, inserts, updates y deletes.

Una muestra de T-SQL y de SP seria:

```
Alter Procedure UE21_Listar_Alumnos
As
Select * From Alumnos
```

Este SP devuelve un conjunto de resultados de la base de datos UE21, devuelve todos los registros de la tabla "Alumnos".

T-SQL como lenguaje de programación, como comentamos anteriormente no solo sirve para hacer consultas o insert, Transact - SQL es un potente lenguaje de programación orientado a SQL Server y como tal tiene:

- Instrucciones para el control de flujo.
- Variables.
- Tipos de Datos.
- Funciones matemática, de tratamiento de cadenas, de fecha y hora.
- Pero además incluye funciones propias del SQL Sever para trabajar con las bases de datos.

Elementos y Conceptos Básicos del Lenguaje

- Transact SQL no es CASE-SENSITIVE, es decir, no diferencia mayúsculas de minúsculas como otros lenguajes de programación como C o Java.
- Un comentario es una aclaración que el programador incluye en el código. Son soportados 2 estilos de comentarios, el de línea simple y de multilínea, para lo cual son empleados ciertos caracteres especiales como son:

```
-- Para un comentario de línea simple
/* ... */ Para un comentario de varias líneas
```

- Un literal es un valor fijo de tipo numérico, carácter, cadena o lógico no representado por un identificador (es un valor explícito).
- Una variable es un valor identificado por un nombre (identificador) sobre el que podemos realizar modificaciones. En Transact SQL los identificadores de variables deben comenzar por el carácter **@**, es decir, el nombre de una variable debe comenzar por **@**.Para declarar
- variables en Transact SQL debemos utilizar la palabra clave **declare**, seguido del identificador y tipo de datos de la variable.

Scripts y lotes

- Un script de Transact SQL es un conjunto de sentencias de Transact SQL en formato de texto plano que se ejecutan en un servidor de SQL Server.

- Un script está compuesto por uno o varios lotes. Un lote delimita el alcance de las variables y sentencias del script. Dentro de un mismo script se diferencian los diferentes lotes a través de la instrucción **GO**.

Los comandos de SQL y TRANSACT-SQL se dividen en tres categorías:

- Lenguaje de manipulación de datos (DML): este incluye los procedimientos comunes de SQL, estos son: SELECT, INSERT, UPDATE y DELETE. DML es a veces erróneamente denominado Lenguaje de Modificación de Datos, lo que es engañoso, ya que SELECT no modifica datos. Sin embargo, manipula los datos devueltos.
- Lenguaje de Definición de datos (DDL): Comandos que crean y manejan Tablas de datos, los Constraints y otros objetos de la base de datos.
- Lenguaje de Control de datos (DCL): Son Comando de seguridad tales como: grant, revoke and deny.

DICCIONARIO DE DATOS

Para Lograr el objetivo de este proyecto, nos basaremos en la utilización y aprovechamiento de los metadatos del Diccionario de Datos de SQLSERVER, de este modo estaremos recuperando la información almacenada que nos servirá para la creación de las clases como así también de los objetos de BD (Stores Procedures) que administraran las entidades persistentes (tablas) de información, para ser luego utilizados en la creación de Sistemas Transaccionales basados en SGDB relacionales, tomando como foco de estudio el Administrador SQL SERVER 2000 – SQL SERVER 2005.

Dada la importancia de los Metadatos en este proyecto, realizaremos un estudio de los mismos.

Todo buen sistema de gestión de base de datos tiene algún tipo de Diccionario de Datos o Metadatos y SQL SERVER no es la excepción.

El Diccionario de Datos almacena información acerca de la estructura de la base de datos, y la información de autorización, y datos acerca de las relaciones [Korth y Silberschatz]. Es el lugar donde se almacena toda la información acerca de los datos como las descripciones, orígenes, relaciones, privilegios de los usuarios, estadísticas (cuantos registros tiene cada tabla, índices, etc.) Generalmente, un diccionario de datos almacena:

- Nombre, tipo y tamaño de los datos.
- Nombre de las relaciones entre los datos.
- Restricciones de integridad sobre los datos.
- Nombre de los usuarios autorizados a acceder a la base de datos.
- Esquema externo, conceptual e interno, y correspondencia entre los esquemas.
- Estadísticas de utilización, tales como la frecuencia de las transacciones y el número de accesos realizados a los objetos de la base de datos.

Algunos de los beneficios que reporta el diccionario de datos son los siguientes:

- La información sobre los datos se puede almacenar de un modo centralizado. Esto ayuda a mantener el control sobre los datos, como un recurso que son.
- El significado de los datos se puede definir, lo que ayudará a los usuarios a entender el propósito de los mismos.
- La comunicación se simplifica ya que se almacena el significado exacto. El diccionario de datos también puede identificar al usuario o usuarios que poseen los datos o que los acceden.
- Las redundancias y las inconsistencias se pueden identificar más fácilmente ya que los datos están centralizados.
- Se puede tener un historial de los cambios realizados sobre la base de datos.

- El impacto que puede producir un cambio se puede determinar antes de que sea implementado, ya que el diccionario de datos mantiene información sobre cada tipo de dato, todas sus relaciones y todos sus usuarios.
- Se puede hacer respetar la seguridad.
- Se puede garantizar la integridad.
- Se puede proporcionar información para auditorías.

Los Diccionarios de Datos (DDD) de los sistemas de base de datos no son iguales, aunque mantiene los mismos lineamientos o las mismas características.

EL "DDD" contienen metadatos, una definición para estos seria: "Datos que describen datos, es este caso son datos sobre la base de datos en particular.

Los metadatos describen la estructura y el significado de los datos, así como la estructura y el significado de las aplicaciones y procesos. Es importante recordar que los metadatos son abstractos, tienen contexto y pueden utilizarse en un entorno de desarrollo con diferentes propósitos.

La información usada por SQL Server y sus componentes son almacenadas en tablas especiales denominadas como tablas del sistema. Estas tablas no deben alterarse directamente por el usuario

Si desea obtener información almacenada en las tablas del sistema debe usar:

- Información de la vista esquema (schema view).
- Procedimientos Almacenados de sistema.
- Instrucciones Transact-SQL y funciones.
- SQL-DMO.
- Catálogo de funciones API.

Las tablas del sistema almacenan información, llamada Metadata, acerca del sistema y de los objetos de las bases de datos. Todas las tablas del sistema comienzan con el prefijo SYS Ejemplo: SELECT * FROM SYSUSUARIOS.

Como comentamos anteriormente SQL Server soporta bases de datos del sistema y bases de datos del usuario.

Las bases de datos del sistema, almacenan información que permite operar y administrar el sistema, mientras que las de usuario almacenan los datos requeridos por las operaciones del cliente.

Las bases de datos del sistema son:

Master

La base de datos master se compone de las tablas de sistema que realizan el seguimiento de la instalación del servidor y de todas las bases de datos que se creen posteriormente. Asimismo controla las asignaciones de archivos, los parámetros de configuración que afectan al sistema, las cuentas de inicio de sesión. Esta base de datos es crítica para el sistema, así que es bueno tener siempre una copia de seguridad actualizada.

tempdb

Es una base de datos temporal, fundamentalmente un espacio de trabajo, es diferente a las demás bases de datos, puesto que se regenera cada vez que arranca SQL Server. Se emplea para las tablas temporales creadas explícitamente por los usuarios, para las tablas de trabajo intermedias de SQL Server durante el procesamiento y la ordenación de las consultas.

model

Se utiliza como plantilla para todas las bases de datos creadas en un sistema. Cuando se emite una instrucción CREATE DATABASE, la primera parte de la base de datos se crea copiando el contenido de la base de datos "*model*", el resto de la nueva base de datos se llena con páginas vacías.

msdb

Es empleada por el servicio SQL Server Agent para guardar información con respecto a tareas de automatización como por ejemplo copias de seguridad y tareas de duplicación, asimismo solución a problemas.

La información contenida en las tablas que contiene esta base de datos, es fácilmente accedida desde el Administrador Empresarial, así que se debe tener cuidado de modificar esta información directamente a menos que se conozca muy bien lo que se esta haciendo.

Distribution

Almacena toda la información referente a la distribución de datos basada en un proceso de replicación.

Existen dos métodos para obtener los Metadatos, uno son los *procedimientos almacenados* del sistema y el otro método son las *vistas de esquema de información*[10].

Los procedimientos almacenados del sistema son guardados en la base de datos Master y son típicamente identificados por el prefijo sp_. Ellos realizan una amplia variedad de tareas para soportar las funciones del SQL Server soportando: llamadas de aplicaciones externas para datos de las tablas del sistema, procedimientos generales para administración de las bases de datos, y funciones de administración de seguridad.

Ejemplo de un procedimiento almacenado del sistema:

sp_column_privileges Devuelve información acerca de los privilegios de columna de una tabla del entorno actual.

Puede que consultar directamente las tablas del sistema no proporcione información precisa, si las tablas del sistema cambian en futuras versiones.

Las Vistas de esquema de información proporcionan una vista interna e independiente de las tablas del sistema de los metadatos de SQL Server. Las vistas de esquema de información permiten que las aplicaciones funcionen correctamente aunque se hayan realizados cambios significativos en las tablas del sistema. Estas Vistas se definen en un esquema especial llamado INFORMATION_SCHEMA, las cuales nos devolverán un alto nivel de este tipo de información. Cada INFORMATION_ SCHEMA contiene metadatos para todos los objetos de datos almacenados en esa base de datos en particular.

INFORMATION_SCHEMA ofrece muchas vistas. Estas son las vistas disponibles:

Constraint_Column_Usage, Column_Domain_Usage, Domain_Cosntraints,
Column_Privileges,Check_Connstraints, Columns, Constraint_Table_Usage,
View_Table_Usage, Domains key_Column_Usage, Schemata, Views,
Parameters, Tables, Referencial_Constraints, Routines, Routine_Columns,
Table_Constraints, Table_Privileges, View_Column_Usage.

Por ejemplo, la vista **Columns** de INFORMATION_SCHEMA devuelve información de las columnas de una determinada tabla, en contraste, la vista **Tables** con previa definición para columna **Table_Type** a un valor **View** en la cláusula **Where** devolverá información acerca de las vistas.

[10] Uso de las vistas de Esquemas de Información:
http://www.elguille.info/colabora/NET2005/Percynet_vistas_informacion_esquema_sqlserver.htm - Abril/2009

Nombre de la base de datos Nombre del esquema

```
SELECT * FROM Northwind.INFORMATION_SCHEMA.COLUMNS

WHERE TABLE_NAME=  products
```

Nombre de la vista

RELEVAMIENTO DE PRODUCTOS

En esta sección, realizaremos un análisis comparativo de las herramientas que se encuentran en el mercado. Obviamente de aquellas que presentan una funcionalidad similar a nuestro proyecto y que ostentan un reconocimiento por parte de los desarrolladores / diseñadores / grupos de desarrollos / organizaciones de software.

Esta lista está lejos de ser exhaustiva, pero si queremos exponer los beneficios de nuestra herramienta, a continuación, se provee una lista de los 5 principales productos competidores en nuestro medio:

Id.	Producto	Organización	Link
1	Clarión 6	UniSolutions	www.unisolutions.com.ar
2	AjGenesis	ajlopez	www.ajlopez/ajgenesis
3	Codesmith	CodesmithTools	www.codesmithtools.com
4	Altova UModel 2008	Altova	www.altova.com/umodel.html
5	Iron Speed	Iron Speed Inc	www.ironspeed.com

Si deberíamos realizar una observación, la misma seria que todas de ellas se basan en la producción de código automatizado a partir de plantillas predefinidas para un leguaje especifico, así es que, para describir la capa de negocio, se podrá especificar un template en particular que defina un lenguaje de programación y realizar el producto resultante a partir de la misma, permitiendo ser totalmente configurable y estandarizado. De igual forma, para el código que se establecerá en el motor de base de datos que se designe.

Como mencionamos, el uso de estas plantillas los hacen totalmente configurables pero incrementa significativamente la complejidad de la herramienta, teniendo que prácticamente aprender una nueva heurística de desarrollo.

Si bien estas herramientas son muy buenas, también tienen un costo (no así AjGenesis, puesto que es Open Source con toda ideología que ello representa) y aunque algunas de ellas proveen su correspondiente versión Free (CodeSmith - IronSpeed), las limitaciones que esta representan, llevan a adquirir el producto final.

En el caso de necesitar una aplicación mas empresarial (por así decirlo) en la cual se necesite una distribución orientada a servicios, o quizá los requerimientos del usuario sean cambiantes, Clarion 6,

Altova UModel 2008, CodeSmith están a la altura, aunque estas herramientas necesitan licencias realmente costosas, y que generalmente poseen aspectos funcionales que muy rara vez se utilicen y de hecho ya vienen soportadas por su propio IDE de, que en algunos casos, es demasiado complejo para configurar y administrar.

Nuestra herramienta provee una ventaja sobre estos productos, que si bien son muy buenos, Promoteo2 se orienta básicamente a una acotación del desarrollo de un producto de SW orientado a la sintaxis y semántica de SQL SERVER 2000 – 2005, como así también código VB de MS VP 2003- 2005; proveyendo una interfaz ágil, intuitiva y fácil de utilizar, dejando como resultado de su utilización los productos .sql y .vb para la administración de los primeros.

Podemos mencionar que nuestra producto posee, a diferencia de las listadas anteriormente, la capacidad de realizar consultas multitablas relacionadas entre si por medio de CONSTRAINTS brindando una base confiable para la creación de reportes estadísticos y consultas de mayor nivel de detalle.

Al momento de realizar el relevamiento de las licencias de estos productos, nos encontramos que el costo de las mismas ascienden, versiones con funcionalidad acotada desde los U$S 5.000 hasta monto final de U$S 15.000 por licencia.

Capitulo IV

REQUERIMIENTOS

Funcionales:

- ✓ Generación automática de los stores procedures ABMC y consulta compleja de los objetos de base de datos seleccionados a partir de una base de datos / esquema seleccionado (utilizando autentificación integrada o por medio de usuario)
- ✓ Generación automática de las clases administradores de los objetos de base datos.
- ✓ Ejecución en la base de datos seleccionada de los SP ABMC y consulta compleja (multitabla)
- ✓ Permitir seleccionar base de datos de red
- ✓ Visualizar las tablas de la base de datos / esquema seleccionado, mostrando: nombre, columnas, tipo, constraints.
- ✓ Leer la información de las bases de datos SQL SERVER 2000 y SQL SERVER 2005

No Funcionales

- ✓ Aplicativo WEB utilizando ASP.NET
- ✓ Generación de una vista que provee la información de las tablas de la BD / esquema seleccionado.

Capitulo V

PLAN DEL PROYECTO

Introducción

Este Plan de Desarrollo del Software es una versión preliminar preparada para ser incluida en la propuesta elaborada como respuesta al proyecto para la empresa de transporte de cargas “Jumbo”. Este documento provee una visión global del enfoque de desarrollo propuesto.

El proyecto esta basado en la metodología de Rational Unified Process, dividiendo en 4 fases el desarrollo del proyecto y detallando cada una de estas, las actividades a realizar y los artefactos (entregables) que serán generados para dar una visión global de todo proceso.

Este documento es a su vez uno de los artefactos de RUP.

Propósito

El propósito del Plan de Desarrollo de Software es proporcionar la información necesaria para controlar el proyecto. En él se describe el enfoque de desarrollo del software.

Entregables del proyecto

A continuación se indican y describen cada uno de los artefactos que serán generados y utilizados por el proyecto y que constituyen los entregables. Esta lista constituye la configuración de RUP desde la perspectiva de artefactos, y que proponemos para este proyecto.

Es preciso destacar que de acuerdo a la filosofía de RUP (y de todo proceso iterativo e incremental), todos los artefactos son objeto de modificaciones a lo largo del proceso de desarrollo, con lo cual, sólo al término del proceso podríamos tener una versión definitiva y completa de cada uno de ellos. Sin embargo, el resultado de cada iteración y los hitos del proyecto están enfocados a conseguir un cierto grado de completitud y estabilidad de los artefactos.

1) Plan de Desarrollo

Es el presente documento donde además se incluye una introducción a la problemática, justificación del proyecto, Objetivos Generales y Específicos, Alcance y Límite del proyecto y Marco teórico.

2) Relevamiento

Información a Relevar que son necesarias para la construcción del Software

3) Glosario

Es un documento que define los principales términos usados en el proyecto. Permite establecer una terminología consensuada.

4) Modelo de Casos de Uso

El modelo de Casos de Uso presenta las funciones del sistema y los actores que hacen uso de ellas. Se representa mediante Diagramas de Casos de Uso.

Si bien este modelo se inicia en anteriormente, en la búsqueda de requisitos, para esta etapa debe de estar terminado.

5) Especificaciones de Casos de Uso (Principales)

Para los casos de uso que lo requieran (cuya funcionalidad no sea evidente o que no baste con una simple descripción narrativa) se realiza una descripción detallada utilizando una plantilla de documento, donde se incluyen: precondiciones, post-condiciones, flujo de eventos, requisitos no-funcionales asociados. También, para casos de uso cuyo flujo de eventos sea complejo podrá adjuntarse una representación gráfica mediante un Diagrama de Actividad.

6) Prototipos de Interfaces de Usuario

Se trata de prototipos que permiten al usuario hacerse una idea más o menos precisa de las interfaces que proveerá el sistema y así, conseguir retroalimentación de su parte respecto a los requisitos del sistema. Estos prototipos se realizarán como: dibujos a mano en papel, dibujos con alguna herramienta gráfica o prototipos ejecutables interactivos, siguiendo ese orden de acuerdo al avance del proyecto. Sólo los de este último tipo serán entregados al final de la fase de Elaboración, los otros serán desechados. Asimismo, este artefacto, será desechado en la fase de Construcción en la medida que el resultado de las iteraciones vayan desarrollando el producto final.

7) Modelo de Datos

Previendo que la persistencia de la información del sistema será soportada por una base de datos relacional, este modelo describe la representación lógica de los datos persistentes, de acuerdo con el enfoque para modelado relacional de datos. Para expresar este modelo se utiliza un Diagrama de Clases (donde se utiliza un profile UML para Modelado de Datos, para conseguir la representación de tablas, claves, etc.).

8) Implementación del Producto

Los ficheros del producto empaquetados y almacenadas en un CD con los mecanismos apropiados para facilitar su instalación. El producto, a partir de la primera iteración de la fase de Construcción es desarrollado incremental e iterativamente, obteniéndose una nueva release al final de cada iteración.

Plan del Proyecto

El desarrollo e implementación del sistema se dividirá en cuatro fases, las mismas no representan fronteras taxativas, si no más bien que se superpondrán debido al ciclo de vida iterativo e incremental que se asumirá.

Las fases del proyecto expuestas a continuación coinciden con las fechas de entrega del trabajo para la materia Seminario Final. A continuación de las mismas, expondré un Diagrama de Gantt en el cual quedara debidamente especificado la duración de cada una de las fases en el desarrollo e implementación del sistema.

1ERA FASE:

Realizar Plan de Desarrollo

- Introducción
  - Idea del Proyecto
  - Problemática
- Objetivos
  - Generales y Particulares
- Marco Teórico

2DA FASE:

Análisis del Proyecto

- Relevamiento de SQL SERVER Relevamiento
- Investigación y Análisis del Diccionario de Datos
- Establecimiento de los requerimientos

3RA FASE:

Diseño

- Modelado de Casos de Usos
- Modelado de Datos del Diccionario de Datos
- Realización de Interfaz de Usuario

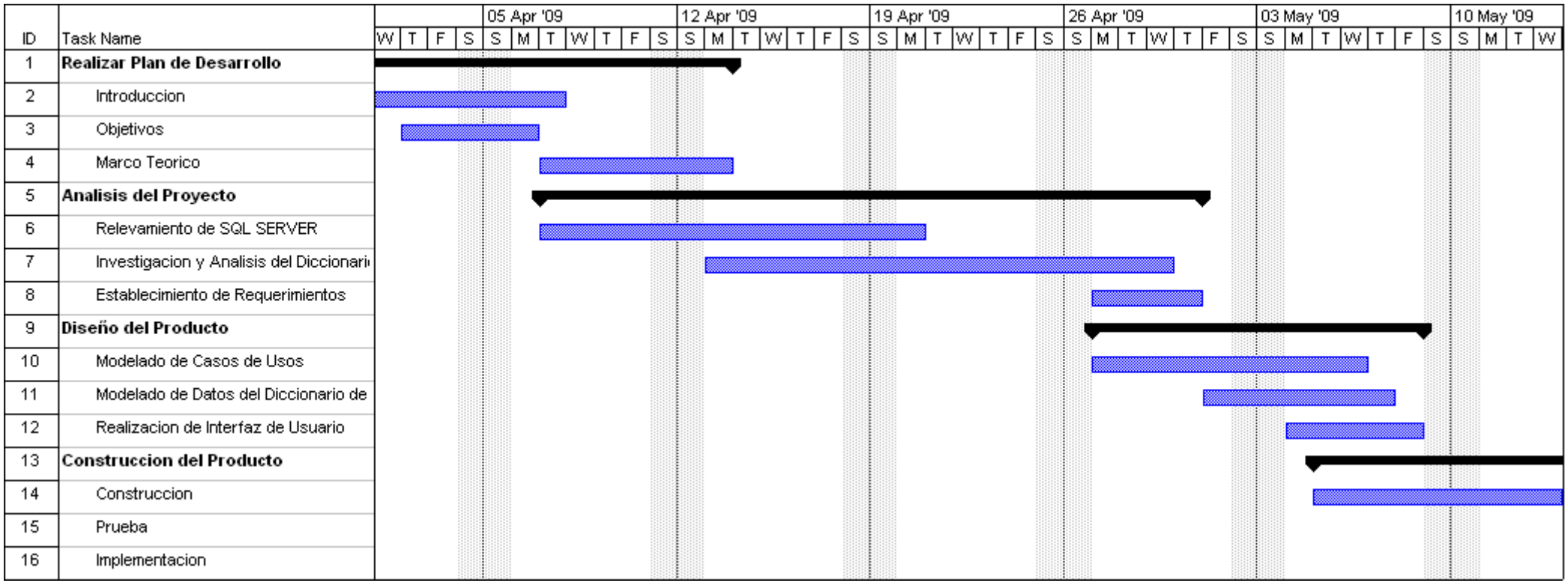
4TA FASE:

Construcción del Producto

- Construcción
- Prueba
- Implementación

ID	Nombre	Duración	Comienzo	Finalización
1	**Realizar Plan de Desarrollo**	9 días	01/04/09	13/04/09
2	Introducción	5 días	01/04/09	07/04/09
3	Objetivos	3 días	02/04/09	06/04/09
4	Marco Teórico	5 días	07/04/09	13/04/09
5	**Análisis del Proyecto**	18 días	07/04/09	30/04/09
6	Relevamiento de SQL SERVER	10 días	07/04/09	20/04/09
7	Investigación y Análisis del Diccionario de Datos	13 días	13/04/09	29/04/09
8	Establecimiento de Requerimientos	4 días	27/04/09	30/04/09
9	**Diseño del Producto**	10 días	27/04/09	08/05/09
10	Modelado de Casos de Usos	8 días	27/04/09	06/05/09
11	Modelado de Datos del Diccionario de Datos	5 días	01/05/09	07/05/09
12	Realización de Interfaz de Usuario	5 días	04/05/09	08/05/09
13	**Construcción del Producto**	29 días	05/05/09	12/06/09
14	Construcción	16 días	05/05/09	26/05/09
15	Prueba	10 días	25/05/09	05/06/09
16	Implementación	8 días	03/06/09	12/06/09

Diagrama de Gantt



Estimación horas de trabajo

Workflow	Fases			
	Inicio	Elaboración	construcción	Transicion
Requisitos	Implementador:	Se estima la participación en este WF por la fase de elaboración e inicio, orientandose mayormente a la construcción de prototipos. Tiempo estimado: 2 hs. 1 Persona Jr.		
	Líder de Proyecto:	Gestión de: tiempos, recursos, estimaciones, guía de actividades junto a AFU's. Participación en las 4 fases, mayormente en Inicio y elaboración. Tiempo estimado: 15 hs. 1 Persona Sr		
	Analista Funcional:	Relevamiento, especificación, elicitacion. 20 hs. 2 Personas SSr		
	Analista de Testing:	No se preveen aun participación		
	Diseñador / Arquitecto:	No se preveen aun participación		
Análisis	Implementador:	Se estima la participación en este WF por la fase de elaboración e inicio, orientandose mayormente a la construcción de prototipos. Tiempo estimado: 3hs. 1 Persona Jr.		
	Líder de Proyecto:	Gestión de: tiempos, recursos, estimaciones, guía de actividades junto a AFU's. Participación en las 4 fases, mayormente en Inicio y elaboración. Tiempo estimado: 15 hs. 1 Persona Sr		
	Analista Funcional:	Relevamiento, especificación, elicitacion. 15 hs. 2 Personas SSr		
	Analista de Testing:	No se preveen aun participación		
	Diseñador / Arquitecto:	No se preveen aun participación		
Diseño	Implementador:	Se prevee la participación como líder referente técnico. 1 Persona Sr. 5 hs.		
	Líder de Proyecto:	Gestión de las actividades de los Diseñadores con los AFU's. 1 Persona Sr. 5 hs.		
	Analista Funcional:	Actividad centrada en la fase elaboración y construcción en colaboración con los Diseñadores / Arquitectos. 5 hs. 1 persona Sr.		
	Analista de Testing:	Diseño, junto con los Diseñadores de los casos de pruebas. 1 Persona SSr. 3 hs.		
	Diseñador / Arquitecto:	Actividad centrada en la fase elaboración y construcción. 5 hs. 1 persona Sr.		
Implementación	Implementador:	Participación en las dos ultimas fases. Desarrollo del producto final. 3 personas SSr. 40hs Totales		
	Líder de Proyecto:	Gestión de: tiempos, recursos, estimaciones, guía de actividades junto a AFU's - Implementadores - Diseñadores - Analistas de Testing. Participación en las 4 fases, mayormente en construcción Tiempo estimado: 5 hs. 1 Persona Sr		
	Analista Funcional:	Coordinación con los desarrolladores. 1 Persona SSr- 5 hs.		
	Analista de Testing:	Validación junto con los Diseñadores de los casos de pruebas. 1 Persona SSr. 5 hs.		
	Diseñador / Arquitecto:	No se preveen aun participación		

Prueba	Implementador:	No se preveen aun participación
	Líder de Proyecto:	Coordinación con los Analistas de testing sobre los casos de pruebas, probar y establecer hitos de control y nuevas reléase. Nuevas iteraciones. 5 hs. 1 Persona Sr
	Analista Funcional:	Coordinación con los analistas de testing. 1 Persona SSr. 2 hs.
	Analista de Testing:	Ejecución de los casos de pruebas. Análisis y retroalimentación. 10 hs.
	Diseñador / Arquitecto:	No se preveen aun participación

ESTIMACION DE COSTOS

Costo en base a la estimación anterior y de acuerdo las tarifas manejadas por el consejo de ciencias informáticas de la provincia de Córdoba, realizamos el siguiente cuadro con el costo total del proyecto:

Roles	Horas	Tarifa	Precio
Implementados:	50,00	$ 40,00	$ 2.000,00
Líder de Proyecto:	50,00	$ 70,00	$ 3.500,00
Analista Funcional:	47,00	$ 50,00	$ 2.350,00
Analista de Testing:	18,00	$ 30,00	$ 540,00
Diseñador / Arquitecto:	5,00	$ 50,00	$ 250,00
			$ 8.640,00

Capitulo VI

PRODUCTO

Cosas que hay que Construir

Puesto que el objetivo es automatizar la tarea del programador, por lo tanto el producto final de la ejecución de la herramienta serán una serie de archivos divididos en dos categorías, a saber:

* Clases que se utilizaran para administrar los objetos TABLES de BD, con sus correspondientes métodos SET – GET. Dichas clases se almacenarán en archivos físicos con extensión VB.NET 2005 (.vb) y su sintaxis será compatible con el lenguaje mencionado. La descripción del archivo creado será el mismo de la tabla a la cual administrara. Siguiendo con el ejemplo de de la BD Northwind, la clase para la tabla **catergories** será categories.vb

* Archivos físicos .SQL que se crearan con el objetivo de mantener la referencia a los SP que realizaran las tareas de INSERT / DELETE / UPDATE / SELECT sobre los objetos TABLES de bases de datos a los cuales harán referencia las clases creadas anteriormente. Estos archivos se crearan respetando la sintaxis TRANSACT-SQL para ser ejecutados en SQL SERVER. La descripción del archivo creado será: ***prc_[objetivo]_[nombreTabla].sql.*** Ejemplo: ***prc_alta_categories.sql***

Se destaca que la aplicación brinda la posibilidad de editar dichas clases en la ejecución de la misma.

Para facilitar las tareas, el aplicativo visualizara un esquema con la estructura de tablas de la BD, permitiendo una visualización ágil de las mismas por parte del operador.

Almacenamiento de lo Obtenido

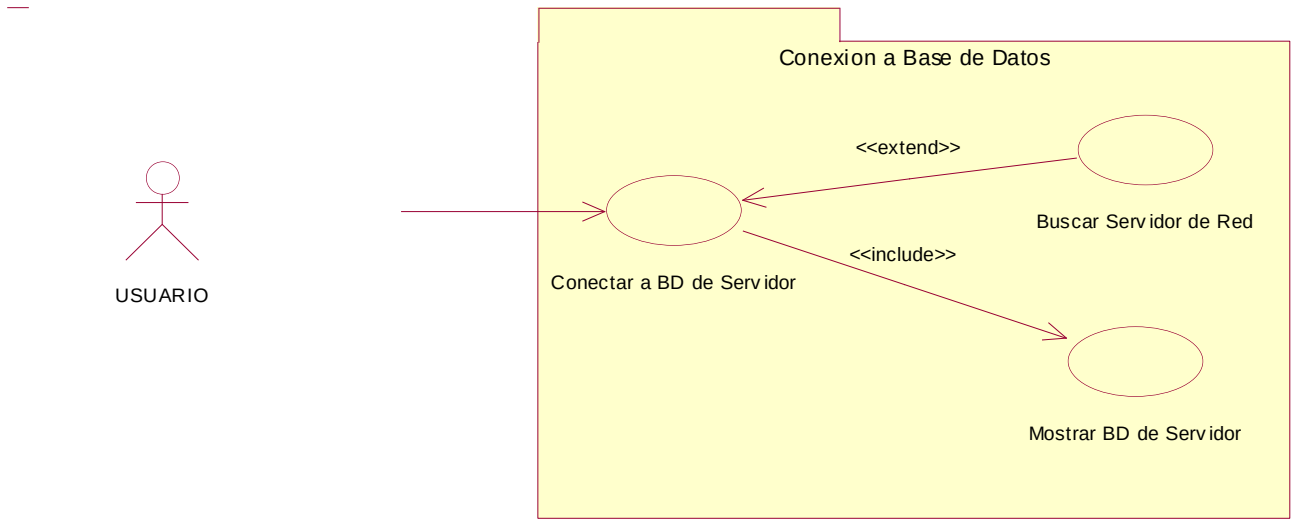
Dados los tipos de productos a obtener, la aplicación almacenara los mismos en dos ubicaciones diferentes (dentro de la raíz de C:\Inetpub\wwwroot\Promoteo2\Products):

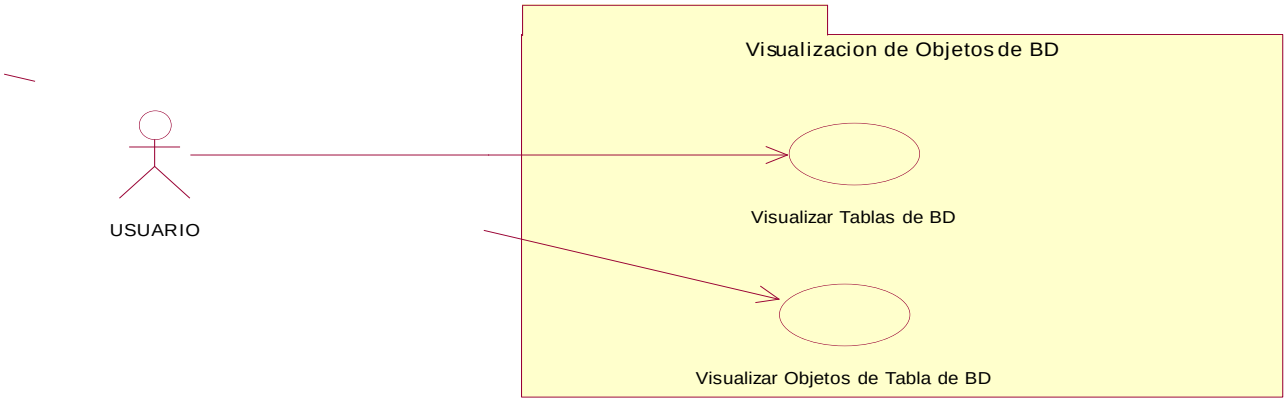
* Carpeta “CLASS”: donde se almacenarán los archivos .vb dentro de una carpeta creada con el nombre de la BD de la cual se ha obtenido la información. Ejemplo: C:\Inetpub\wwwroot\Promoteo2\Products\Class\Northwind\categories.vb

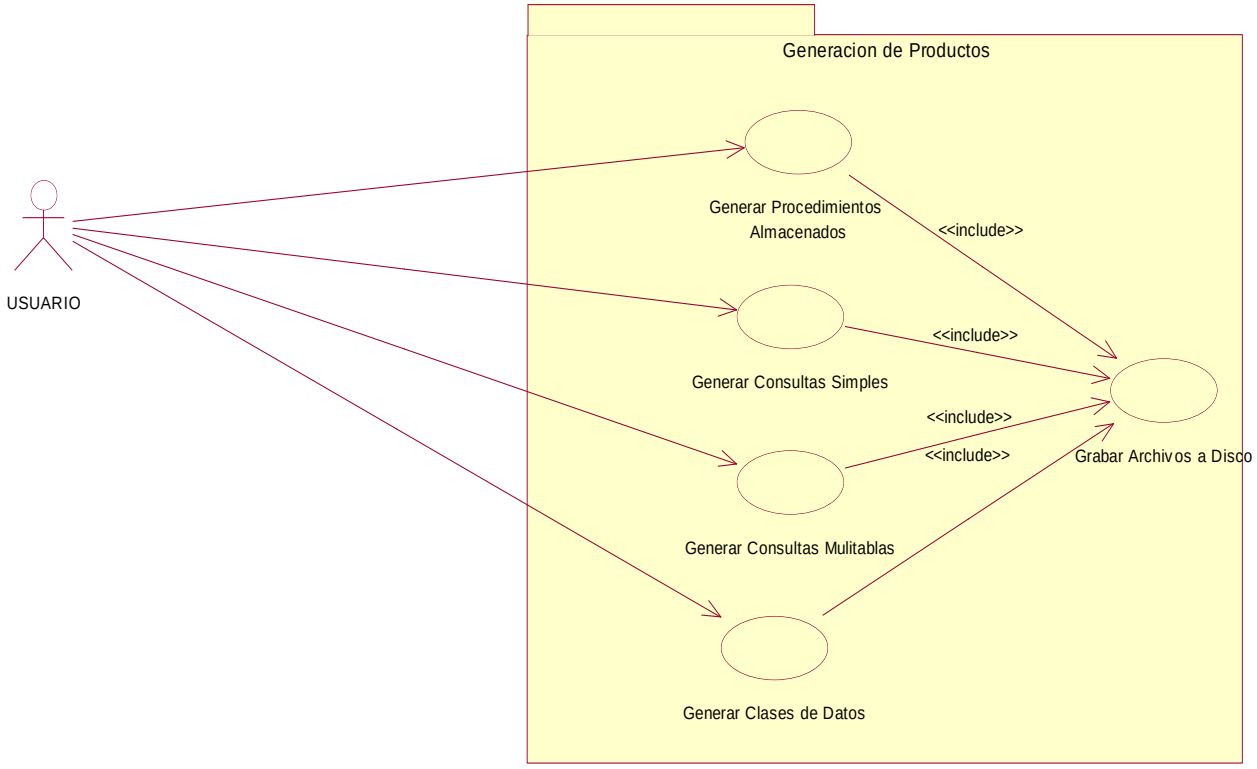
* Carpeta “StrPdr”: donde se almacenaran los archivos .sql dentro de una carpeta creada con el nombre de la BD de la cual se ha obtenido la información. Ejemplo: C:\Inetpub\wwwroot\Promoteo2\Products\StrPdr\Northwind\prc_alta_categories.sql

CASOS DE USOS

Diagrama de Casos de Usos






Generacion de Productos
USUARIO
Generar Procedimientos Almacenados
<<include>>
Generar Consultas Simples
<<include>>
<<include>>
<<include>>
Grabar Archivos a Disco
Generar Consultas Mulitablas
Generar Clases de Datos

Listado de Casos de Usos

Listado de Casos de Uso								
Paquete	**Nº CU**	**Nombre del Caso de Uso**	**Objetivo del Caso de Uso**	**Complejidad**	**Prioridad**	**Categoría**	**Significativo para la Arquitectura**	**Cantidad de Actores**
Conexión a BD	1	Conectar a Servidor de Base de Datos	Realizar la conexión exitosa a una BD seleccionada por el usuario	Simple	Alta	Esencial	Significativo para la Arquitectura	1,00
Visualización de Objetos de Base Datos	2	Visualizar Tablas de la Base de Datos	Recuperar las tablas la BD Seleccionada	Simple	Alta	Esencial	Significativo para la Arquitectura	1,00
Visualización de Objetos de Base Datos	3	Visualizar elementos de la tabla de Base de Datos	Recuperar los objetos asociados una tabla seleccionada.	Simple	Alta	Esencial	Significativo para la Arquitectura	1,00
Generación de Productos	4	Generar Procedimientos Almacenados (ABM)	Generar automáticamente los Insert, Update y Delete de todas las tablas o de las seleccionadas.-	Compleja	Alta	Esencial	Significativo para la Arquitectura	1,00
Generación de Productos	5	Generar Consultas Simples	Generar los Select simples de todas las tablas o de las seleccionadas	Compleja	Alta	Esencial	Significativo para la Arquitectura	1,00
Generación de Productos	6	Generar Consultas Multitablas	Generar consultas multitablas de todas las tablas seleccionadas	Compleja	Alta	Esencial	Significativo para la Arquitectura	1,00
Generación de Productos	7	Generar Clase de Datos	Generar las Clases de los Procedimientos Almacenados que se han Generados	Compleja	Alta	Esencial	Significativo para la Arquitectura	

Generación de Productos	8	Grabar archivo a disco	Grabar el archivo generado en disco de acuerdo a una ubicación	Simple	Media	No esencial	No significativo para la Arquitectura	1,00
Conexión a BD	9	Buscar Servidor de Red	Buscar los servidores de red activos disponibles	Simple	Media	No esencial	No significativo para la Arquitectura	1,00
Conexión a BD	10	Visualizar BD del servidor	Mostrar las BD activas de un servidor seleccionada	Simple	Alta	Esencial	Significativo para la Arquitectura	1,00

Descripción de casos de usos

Nombre del Use Case: Generar Procedimientos Almacenados		**Nro. de Orden:** 04
Complejidad: ☐ Alta	☒ Media	☐ Baja
Actor Principal: Usuario (U)		**Actor Secundario:** no aplica
Tipo de Use Case: ☒ Concreto		☐ Abstracto
Objetivo: Generar automáticamente los Insert, Update y Delete de todas las tablas seleccionadas.-		
Precondiciones: Conectado a una base de datos seleccionada, visualización de las tablas de una base de datos seleccionada.		
Post-Condiciones: El sistema ejecuta exitosamente los SP creados. Se cancela el caso de uso si: no se ha seleccionado ninguna tabla, o si no se ha seleccionado al menos una opción, o si el UC "Grabar Archivo a Disco" no se ejecuta correctamente o si el UC "Generar Clases de datos" no se ejecuta correctamente		
Curso Normal		**Alternativas**
1 – El Usuario (U) ingresa al modulo de Generación de Productos.		
2- El U selecciona una o varias tablas del servidor seleccionado.		
3- El sistema muestra las opciones de "Alta", "Baja", "Modificación", "Todos"		
4 – El U selecciona alguna de las opciones visualizadas:"Alta", "Baja", "Modificación", "Todos"		
5- El U selecciona la opción de "Generar"; el sistema verifica que se haya seleccionado por lo menos una tabla y una opción, y existe al menos una tabla seleccionada y una opción seleccionada.		5.A- No existe la selección de al menos una tabla. 5.A.1 El sistema muestra el mensaje "No existen tablas seleccionadas". 5.A.2 FUC 5.B No existe la selección de la menos una opción. 5.B.1. El sistema muestra el mensaje "No existen opciones seleccionadas" 5.B.2 FUC
6- El sistema arma el código TRANSACT SQL de las opciones seleccionadas.		
7- El sistema ejecuta el código confeccionado y lo ejecuta en la BD seleccionada.		
8- Se llama al UC "Grabar Archivo a Disco". Y se ejecuta correctamente		8.A- El UC "Grabar Archivo a Disco" no se ejecuta correctamente. 8.A.1- El sistema muestra un mensaje del error producido. 8.A.2 FUC
9.- Se llama al Caso de Uso "Generar Clases de datos" y se ejecuta correctamente.		9.A- El UC "Generar Clases de datos" no se ejecuta correctamente.

	9.A.1- El sistema muestra un mensaje del error producido. 9.A.2 FUC
10- El sistema muestra el mensaje “Creación exitosa de SP en Base de Datos”	
11- FUC	
Asociaciones de Extensión: no aplica	
Asociaciones de Inclusión: Grabar Archivo a Disco	
Use Case donde se incluye: no aplica	
Use Case al que extiende: no aplica	
Use Case de Generalización: no aplica	
Autor: López Nelson	**Fecha Creación**: 01/07/2009
Autor Última Modificación: López Nelson	**Fecha Última Modificación**: 01/07/2009

Nombre del Use Case: Generar Clase de Datos		**Nro. de Orden:** 07
Complejidad: ☐ Alta	☒ Media	☐ Baja
Actor Principal: no aplica		**Actor Secundario:** no aplica
Tipo de Use Case: ☐ Concreto		☒ Abstracto
Objetivo: Generar las Clases de los Procedimientos Almacenados que se han Generados.-		
Precondiciones: el usuario haya generado la menos un SP		
Post-Condiciones: El sistema muestra exitosamente las clases generadas. Se cancela el caso de uso si: no se ha seleccionado ninguna tabla, o si no se ha seleccionado al menos una opción, o si el UC "Grabar Archivo a Disco" no se ejecuta correctamente.		
Curso Normal		**Alternativas**
1 – El UC comienza cuando se ha generado un SP seleccionado por el usuario.		
2- El sistema arma el script en código VB.NET para ejecutar el SP creado.		
3. El sistema muestra el código generado.		
4- Se llama al UC "Grabar Archivo a Disco". Y se ejecuta correctamente		4.A- El UC "Grabar Archivo a Disco" no se ejecuta correctamente. 4.A.1- El sistema muestra un mensaje del error producido. 4.A.2 FUC
5- El sistema muestra el mensaje "Creación exitosa de la Clases"		
6- FUC		
Asociaciones de Extensión: no aplica		
Asociaciones de Inclusión: Grabar Archivo a Disco		
Use Case donde se incluye: Generar Procedimientos Almacenados – Generar Consultas Simples – Generar Consultas Multitablas.		
Use Case al que extiende: no aplica		
Use Case de Generalización: no aplica		
Autor: López Nelson		**Fecha Creación**: 01/07/2009
Autor Última Modificación: López Nelson		**Fecha Última Modificación**: 01/07/2009

Nombre del Use Case: Generar Consultas Simples	**Nro. de Orden:** 05
Complejidad: ☐ Alta ☒ Media ☐ Baja	
Actor Principal: Usuario (U)	**Actor Secundario:** no aplica
Tipo de Use Case: ☒ Concreto ☐ Abstracto	
Objetivo: Generar los Select simples de todas las tablas o de las seleccionadas.	
Precondiciones: Conectado a una base de datos seleccionada, visualización de las tablas de una base de datos seleccionada.	
Post-Condiciones: El sistema ejecuta exitosamente las Consultas creadas. Se cancela el caso de uso si: no se ha seleccionado ninguna table, o si no se ha seleccionado al menos una opción, o si el UC "Grabar Archivo a Disco" no se ejecuta correctamente o si el UC "Generar Clases de datos" no se ejecuta correctamente	
Curso Normal	**Alternativas**
1 – El Usuario (U) ingresa al modulo de Generación de Productos.	
2- El U selecciona una o varias tablas del servidor seleccionado.	
3- El sistema muestra la opción de "Generar Consulta Simple".	
4 – El U selecciona esta opción.	
5- El U selecciona la opción de "Generar"; el sistema verifica que se haya seleccionado por lo menos una tabla y la opción, y existe al menos una tabla seleccionada y la opción seleccionada.	5.A- No existe la selección de al menos una tabla. 5.A.1 El sistema muestra el mensaje "No existen tablas seleccionadas". 5.A.2 FUC 5.B- El U no selecciono la opción Consulta Simple 5.B.1. El sistema muestra el mensaje "No existen opciones seleccionadas" 5.B.2 FUC
6- El sistema arma el código TRANSACT SQL de las Consultas Simples.	
7- El sistema ejecuta el código confeccionado y lo ejecuta en la BD seleccionada.	
8- Se llama al UC "Grabar Archivo a Disco". Y se ejecuta correctamente	8.A- El UC "Grabar Archivo a Disco" no se ejecuta correctamente. 8.A.1- El sistema muestra un mensaje del error producido. 8.A.2 FUC
9.- Se llama al Caso de Uso "Generar Clases de datos" y se ejecuta correctamente.	9.A- El UC "Generar Clases de datos" no se ejecuta correctamente. 8.A.1- El sistema muestra un mensaje del error producido. 8.A.2 FUC
10- El sistema muestra el mensaje "Creación exitosa de SP en Base de Datos"	

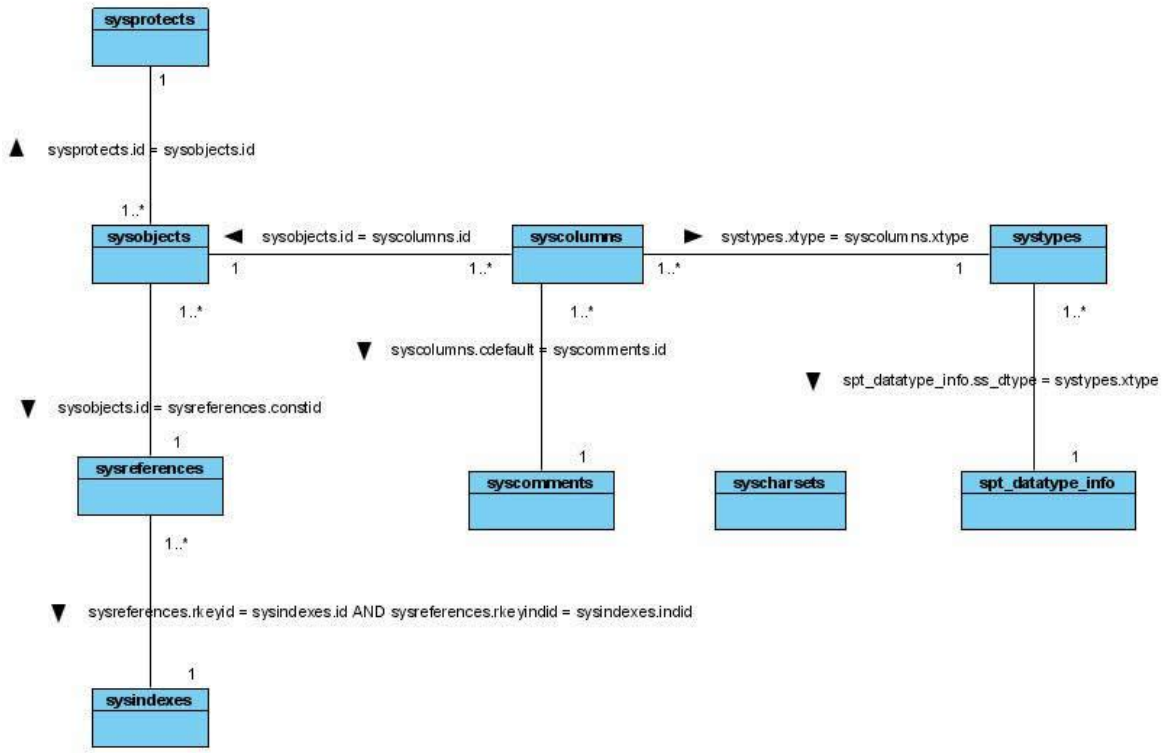
11- FUC	
Asociaciones de Extensión: no aplica	
Asociaciones de Inclusión: Grabar Archivo a Disco	
Use Case donde se incluye: no aplica	
Use Case al que extiende: no aplica	
Use Case de Generalización: no aplica	
Autor: López Nelson	**Fecha Creación**: 01/07/2009
Autor Última Modificación: López Nelson	**Fecha Última Modificación**: 01/07/2009

Nombre del Use Case: Generar Consultas Multitablas	**Nro. de Orden:** 06
Complejidad: ☐ Alta ☒ Media ☐ Baja	
Actor Principal: Usuario (U)	**Actor Secundario:** no aplica
Tipo de Use Case: ☒ Concreto ☐ Abstracto	
Objetivo: Generar consultas multitablas de todas las tablas seleccionadas.	
Precondiciones: Conectado a una base de datos seleccionada, visualización de las tablas de una base de datos seleccionada.	
Post-Condiciones: El sistema ejecuta exitosamente las Consultas creadas. Se cancela el caso de uso si: no se ha seleccionado ninguna tabla, o si no se ha seleccionado al menos una opción, o si el UC "Grabar Archivo a Disco" no se ejecuta correctamente o si el UC "Generar Clases de datos" no se ejecuta correctamente.	
Curso Normal	**Alternativas**
1 – El Usuario (U) ingresa al modulo de Generación de Productos.	
2- El U selecciona una o varias tablas del servidor seleccionado.	
3- El sistema muestra la opción de "Generar Consulta Multitablas".	
4 – El U selecciona esta opción.	
5- El Usuario selecciona una o varias Columnas de las tablas Seleccionadas	
6- El U selecciona la opción de "Generar"; el sistema verifica que se haya seleccionado por lo menos una tabla y la opción, y existe al menos una tabla seleccionada y la opción seleccionada.	5.A- No existe la selección de al menos una tabla. 5.A.1 El sistema muestra el mensaje "No existen tablas seleccionadas". 5.A.2 FUC 5.B- El U no selecciono la opción Consulta Simple 5.B.1. El sistema muestra el mensaje "No existen opciones seleccionadas" 5.B.2 FUC
6- El sistema arma el código TRANSACT SQL de las Consultas Simples.	
7- El sistema ejecuta el código confeccionado y lo ejecuta en la BD seleccionada.	
8- Se llama al UC "Grabar Archivo a Disco". Y se ejecuta correctamente	8.A- El UC "Grabar Archivo a Disco" no se ejecuta correctamente. 8.A.1- El sistema muestra un mensaje del error producido. 8.A.2 FUC
9.- Se llama al Caso de Uso "Generar Clases de datos" y se ejecuta correctamente.	9.A- El UC "Generar Clases de datos" no se ejecuta correctamente. 8.A.1- El sistema muestra un mensaje del error producido. 8.A.2 FUC

10- El sistema muestra el mensaje “Creación exitosa de SP en Base de Datos”	
11- FUC	
Asociaciones de Extensión: no aplica	
Asociaciones de Inclusión: Grabar Archivo a Disco	
Use Case donde se incluye: no aplica	
Use Case al que extiende: no aplica	
Use Case de Generalización: no aplica	
Autor: López Nelson	**Fecha Creación**: 01/07/2009
Autor Última Modificación: López Nelson	**Fecha Última Modificación**: 01/07/2009

MODELO DE DATOS DEL DICCIONARIO

Si bien Prometo 2 no posee un modelo de datos porque no requiere almacenar su configuración en un esquema de BD, si se explayara a continuación el DER de los metadatos, elemento fundamental para la aplicación. Cabe aclarar que si bien, el modelo de datos del diccionario de datos de una BD es mucho más extenso, solo nos basamos en las tablas que en el figuran por contener la información relevante para cumplir con el objetivo de este proyecto.



A continuación se especifica las columnas de estas tablas:

syscolumns	
name	sysname
id	int
xtype	tinyint
typestat	tinyint
xusertype	smallint
length	smallint
xprec	tinyint
xscale	tinyint
colid	smallint
xoffset	smallint
bitpos	tinyint
reserved	tinyint
colstat	smallint
cdefault	int
[domain]	int
number	smallint
colorder	smallint
autoval	varbinary
offset	smallint
collationid	int
[language]	int
status	tinyint
type	tinyint
usertype	smallint
printfmt	varchar
prec	smallint
scale	int
iscomputed	int
isoutparam	int
isnullable	int
[collation]	sysname
tdscollation	binary

sysobjects	
name	sysname
id	int
xtype	char
uid	smallint
info	smallint
status	int
base_schema_ver	int
replinfo	int
parent_obj	int
crdate	datetime
ftcatid	smallint
schema_ver	int
stats_schema_ver	int
type	char
userstat	smallint
sysstat	smallint
indexdel	smallint
refdate	datetime
version	int
deltrig	int
instrig	int
updtrig	int
seltrig	int
category	int
cache	smallint

syscomments	
id	int
number	smallint
colid	smallint
status	smallint
ctext	varbinary
texttype	smallint
[language]	smallint
encrypted	bit
compressed	bit
text	nvarchar

sysreferences	
constid	int
fkeyid	int
rkeyid	int
rkeyindid	smallint
keycnt	smallint
forkeys	varbinary
refkeys	varbinary
fkeydbid	smallint
rkeydbid	smallint
fkey1	smallint
fkey2	smallint
fkey3	smallint
fkey4	smallint
fkey5	smallint
fkey6	smallint
fkey7	smallint
fkey8	smallint
fkey9	smallint
fkey10	smallint
fkey11	smallint
fkey12	smallint
fkey13	smallint
fkey14	smallint
fkey15	smallint
fkey16	smallint
rkey1	smallint
rkey2	smallint
rkey3	smallint
rkey4	smallint
rkey5	smallint
rkey6	smallint
rkey7	smallint
rkey8	smallint
rkey9	smallint
rkey10	smallint
rkey11	smallint
rkey12	smallint
rkey13	smallint
rkey14	smallint
rkey15	smallint
rkey16	smallint

systypes	
name	sysname
xtype	tinyint
status	tinyint
xusertype	smallint
length	smallint
xprec	tinyint
xscale	tinyint
tdefault	int
[domain]	int
uid	smallint
reserved	smallint
collationid	int
usertype	smallint
variable	bit
allownulls	bit
type	tinyint
printfmt	varchar
prec	smallint
scale	tinyint
[collation]	sysname

syscharsets	
type	smallint
id	tinyint
csid	tinyint
status	smallint
name	sysname
description	nvarchar
binarydefinition	varbinary
definition	image

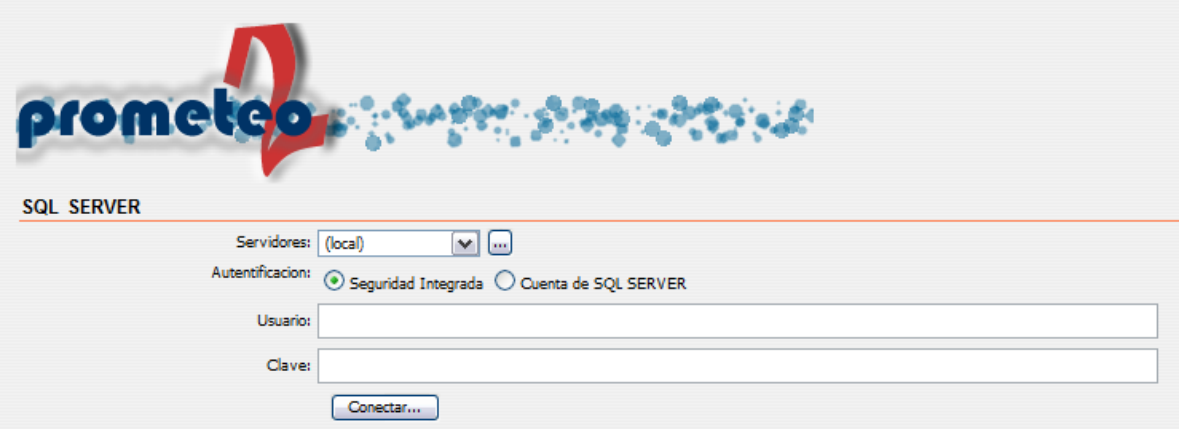
spt_datatype_info	
ss_dtype	tinyint
fixlen	int
ODBCVer	tinyint
TYPE_NAME	sysname
DATA_TYPE	smallint
data_precision	int
numeric_scale	smallint
RADIX	smallint
length	int
LITERAL_PREFIX	varchar
LITERAL_SUFFIX	varchar
CREATE_PARAMS	varchar
NULLABLE	smallint
CASE_SENSITIVE	smallint
SEARCHABLE	smallint
UNSIGNED_ATTRIBUTE	smallint
MONEY	smallint
AUTO_INCREMENT	smallint
LOCAL_TYPE_NAME	sysname
charbin	tinyint
SQL_DATA_TYPE	smallint
SQL_DATETIME_SUB	smallint

sysreprotects	
id	int
uid	smallint
[action]	tinyint
protecttype	tinyint
columns	varbinary
grantor	smallint

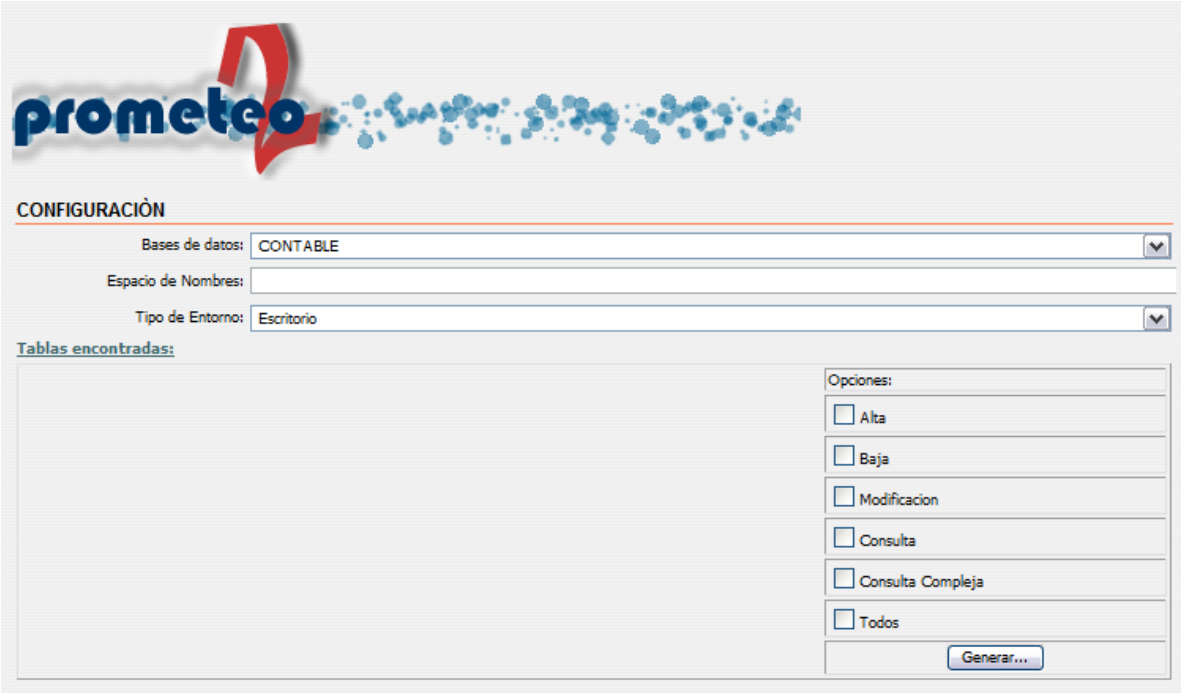
sysindexes	
id	int
status	int
[first]	binary
indid	smallint
root	binary
minlen	smallint
keycnt	smallint
groupid	smallint
dpages	int
reserved	int
used	int
rowcnt	bigint
rowmodctr	int
reserved3	tinyint
reserved4	tinyint
xmaxlen	smallint
maxirow	smallint
OrigFillFactor	tinyint
StatVersion	tinyint
reserved2	int
FirstIAM	binary
impid	smallint
lockflags	smallint
pgmodctr	int
keys	varbinary
name	sysname
statblob	image
maxlen	int
[rows]	int

PANTALLAS DEL SISTEMA

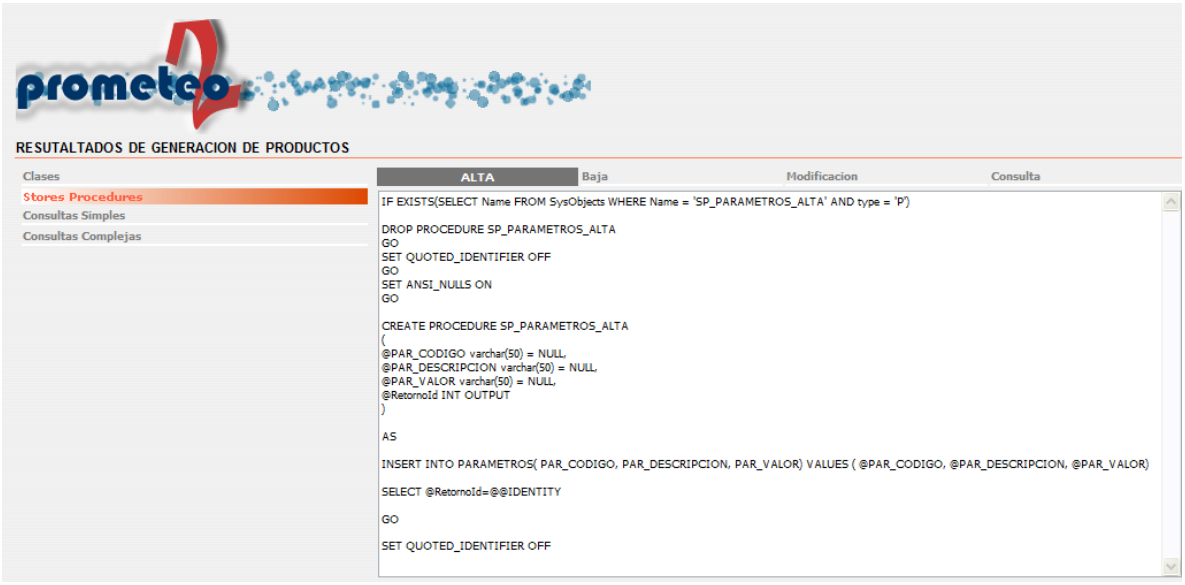
Se visualiza a continuación los prototipos correspondientes a diferentes funcionalidades del sistema en donde, en un principio establecemos a que servidor nos conectaremos y la forma de autentificación al mismo, pudiendo seleccionar un servidor local o bien uno de red:



Una vez realizada la conexión al servidor de base de datos, se visualizara cuales son las Bases de datos que del mismo, y allí se podrá configurar las opciones de generación de productos, eligiendo las tablas y las opciones de generación:



Como mencionamos, los productos generados pueden ser CLASES en el lenguaje de VB.Net y archivos .SQL escritos en lenguaje TRANSACT SQL:



MEDICION DE LA APLICACION

En el marco de este proyecto, y teniendo en cuenta en el objetivo en el cual se encuadra el mismo, necesitamos contrarrestar el uso de la aplicación en un ambiente de desarrollo genérico (una BD de datos conocida) para realizar o confeccionar de forma automática las clases que administraran las tablas de la BD, como así también los STORES PROCEDURES que realizaran las ALTAS / BAJAS / MODIFICACIONES / CONSULTAS / CONSULTAS COMPLEJAS sobre dichos objetos; contra la generación manual de los mimos.

De acuerdo a estas premisas, se decidió tomar la BD NorthWind bajo estudio. La misma se provee como ejemplo en varias aplicaciones MS (Access / SQL SERVER) y se trabaja con ella por presentarse completa en cuanto a objetos y relaciones en ellos.

Por otro lado, se tomo como objeto de prueba a dos programadores ***Semi – Senior***[11] para la generación de las clases (las clases se generaran con VB.NET 2005) y sus correspondientes objetos SP en la base que serán usados en dichas clases.

Dado este modelo de análisis, se tomo la BD y en una primera instancia las tablas paramétricas (9 tablas principales), es decir las ambientales y luego las que registran transacciones dentro del negocio. Se realizaron las clases con sus respectivos métodos (SET / GET) que administran los objetos persistentes mapeados al modelo de persistencia relacional. También, como complemento se crearon los objetos SP los cuales realizan INSERT / DELETE / UPDATE / SELECT sobre las tablas de la BD. Todo esto de formal manual, realizando las pruebas correspondientes, tanto unitarias como integrales en un pequeño aplicativo destinado para tal fin.

Para obtener valores para comparar, se realizaron las mismas tareas sobre las mismas tablas (ambientales y transaccionales), con la herramienta automatizada. Con los objetos creados, se realizaron las pruebas integrales correspondientes, con la misma aplicación usada anteriormente.

De las dos experiencias, pudimos observar que al realizar manualmente las clases con sus correspondientes métodos y objetos de BD para gestionar los registros de las tablas, nos encontramos que:

- fue muy propensas a errores básicos en las llamadas a los SP.
- La duración de construcción duro aproximadamente 7 HS.

[11] Programador de 2 a 6 años de Experiencia. Técnicamente autosuficiente. Puede desarrollar funcionalidades más complejas y ejecutar proyectos de mayor envergadura. Pero no es un crack y todavía comete errores “evitables”.

- Del resultado de las pruebas unitarios – integrales, las correcciones posteriores duro 1,5 hora aproximadamente

Con respecto a la utilización de la herramienta que automatiza la tarea:

- La utilización de la aplicación y ejecución de los procesos duro aproximadamente 5 minutos.
- Del resultado de las pruebas unitarios – integrales, las correcciones posteriores duro 15 minutos aproximadamente
- Se pudieron generar listados mas complejos (estadísticas) con la utilización de la herramienta
- Se generaron clases y objetos SP para administrar las tablas transaccionales, cuya manipulación por parte de los programadores se pudieron adaptar correctamente restricciones del negocio.

Con el objeto de tener una visión más de la aplicación en su ayuda en el desarrollo de aplicaciones empresariales, se propuso el uso del aplicativo con un grupo de programadores ***Juniors***[12].

Con ello, se presento la misma Base de Datos de estudio: NorthWind y de ella se realizaron las clases de cada uno de los objetos tablas de la base de datos, con sus respectivas declaraciones de métodos y atributos. Sumado a esto, también se incluyo la construcción de los principales SP: INSERT / DELETE / UPDATE / SELECT.

La realización de forma manual arrojo los siguientes resultados:

- Fue muy propensas a errores básicos en las llamadas a los SP, tanto lógicos como de estructura, incluyendo captura de errores, su formalización y visualización de valores consistentes y amigables a los usuarios.
- La duración de construcción (clases / SP) duro aproximadamente 13 HS.
- Del resultado de las pruebas unitarios – integrales, y correcciones posteriores duro 3 horas aproximadamente.

Con respecto a la utilización de la herramienta que automatiza la tarea:

- La utilización de la aplicación y ejecución de los procesos duro aproximadamente entre 5 y 15 minutos.
- Del resultado de las pruebas unitarios – integrales, las correcciones posteriores duro 45 minutos aproximadamente
- Se pudieron generar listados más complejos (estadísticas) con la utilización de la herramienta.

[12] Menos de dos años de experiencia. Para desempeñarse suele requerir acompañamiento. El código que genera puede presentar mayor cantidad de bugs de lo esperado

- Se generaron clases y objetos SP para administrar las tablas transaccionales, que manipulación de los programadores se pudieron adaptar correctamente restricciones del negocio.
- Como valor agregado, se destaca que la aplicación proveyó una estructura simple, entendible para los programadores y todos ellos adoptaron la sintaxis que propone aplicar el aplicativo para la instancia de cada una de las clases creadas con ella en el sistema de prueba.

CONCLUSION

Planteado los objetivos anteriormente fijados, se llego a la conclusión de haber alcanzado los mismos de forma satisfactoria, obteniendo como resultado final de este trabajo una herramienta accesible, robusta e intuitiva, que ofrece la automatización en la ejecución de tareas rutinarias en la creación de Software (SW) permitiendo al desarrollador enfocarse en tareas que generan valor agregado al producto.

Esto conlleva a disminuir el tiempo de desarrollo, el cual se expuso en la sección anterior “Medición de la Aplicación”, lo que afecta directamente en la reducción de los costos. Por ultimo destacamos la eliminación de bugs o errores de programación en procedimientos almacenados y clases que los acceden.

Teniendo en cuenta lo anterior, mencionamos que la principal característica de Prometeo 2 es la generación de código (archivos .vb y .sql) confiable e industrializado generando un aporte tecnológico orientado a Software Factory.

No solo se le proporciona un beneficio al desarrollador sino que le agrega satisfacción al no tener que ocuparse en tareas usuales que se vuelven tediosas.

GLOSARIO

Aplicación: Cualquier programa que corra en un sistema operativo y que haga una función específica para un usuario. Por ejemplo, procesadores de palabras, bases de datos, agendas electrónicas, etc.

Archivo: Archivo es el equivalente a 'file', en inglés. Es data que ha sido codificada para ser manipulada por una computadora. Los archivos de computadora pueden ser guardados en CD-ROM, DVD, disco duro o cualquier otro medio de almacenamiento. Usualmente los archivos tienen una 'extensión' después de un punto, que indica el tipo de data que contiene el archivo. Dependiendo del sistema operativo usado, se cargan los programas necesarios para manejar los archivos según su extensión. Ejemplo, panamacom.txt se refiere a un archivo de texto, imagen.jpg a una imagen JPEG, documento.odt a un archivo Open Office, y un word.doc a Word de Microsoft Office.

Atributos: estos son los datos que caracterizan al objeto. Son variables que almacenan datos relacionados al estado de un objeto.

Backup: Copia de Respaldo o Seguridad. Acción de copiar archivos o datos de forma que estén disponibles en caso de que un fallo produzca la perdida de los originales. Esta sencilla acción evita numerosos, y a veces irremediables, problemas si se realiza de forma habitual y periódica.

Base de datos: Conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente. En una base de datos, la información se organiza en campos y registros. Los datos pueden aparecer en forma de texto, números, gráficos, sonido o vídeo.

Base de Datos Relacional : es una base de datos que cumple con el modelo relacional, el cual es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos.

Bugs: es un error o un defecto en el software o hardware que hace que un programa funcione incorrectamente.

Clases: Las clases son declaraciones o abstracciones de objetos, lo que significa, que una clase es la definición de un objeto. Cuando se programa un objeto y se definen sus características y funcionalidades, realmente se programa una clase.

IDE: Entorno de desarrollo.

Métodos: Los métodos de un objeto caracterizan su comportamiento, es decir, son todas las acciones (denominadas operaciones) que el objeto puede realizar por sí mismo. Estas operaciones hacen posible que el objeto responda a las solicitudes externas (o que actúe sobre otros objetos). Además, las operaciones están estrechamente ligadas a los atributos, ya que sus acciones pueden depender de, o modificar, los valores de un atributo.

Juniors:(Programador): Menos de 2 años de experiencia. Para desempeñarse suele requerir acompañamiento. El código que genera puede presentar mayor cantidad de bugs de lo esperado. Probablemente no maneja todas las herramientas que se necesitan para la tarea.

Modelo relacional: Rara la gestión de una base de datos es un modelo de datos basado en la lógica de predicado y en la teoría de conjuntos. Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos.

Objeto: Representación detallada y particular de algo de la realidad. Todo objeto tiene un identidad o nombre, estado (características definidas generalmente en variables) y comportamiento (sus funciones o procedimientos).

Procedimiento almacenado: (stored procedure en inglés)
Es un programa (o procedimiento) el cual es almacenado físicamente en una base de datos.

Programación Orientada a Objetos: Programación Orientada a Objetos (POO) es una filosofía de programación que se basa en la utilización de objetos. El objetivo de la POO es "imponer" una serie de normas de desarrollo que aseguren y faciliten la mantenibilidad y reusabilidad del código.

Semi – senior (Programador): De 2 a 6 años de experiencia. Técnicamente autosuficiente. Puede desarrollar funcionalidades más complejas y ejecutar proyectos de mayor envergadura. Pero no es un crack y todavía comete errores “evitables”.

SQL: Structured Query Language. Es un lenguaje especializado de programación que permite realizar consultas (queries) a bases de datos. Los orígenes del SQL están ligados a los de las bases de datos relaciónales.

BIBLIOGRAFÍA

En esta sección se debe indicar la bibliografía consultada para la realización del proyecto.

Libros:

- SQL Server 2005 “Manual de Referencia” – autor: William Stanek – Mc Graw Hill.-
- Bases de datos con sql 2000. TRANSACT SQL – autor: Jorge Moratalla – Eidos.
- Biblie SQL Server 2005 – autor Paul Nielsen – Wiley Publishing.
-

Referencias en PDF:

- Prometeo_TecnoDB.Pdf.
- Database & data management: The forest and the trees: using oracle and SQL server together to teach ANSI-standard SQL-Gary B. Randolph-October 2003
- Tutorial de visual studio 2005 y sql server 2005.pdf

Referencias de fuentes electrónicas:

- “Diseño de Base de datos” - http://www.wikilearning.com/tutorial/diseno_de_bases_de_datos_en_sql/21129 Consultada: Marzo 2009.
- http://technet.microsoft.com/es-es/library/ms190940(SQL.90).aspx. Consultada: abril 2009
- http://www.elguille.info/colabora/NET2005/Percynet_vistas_informacion_esquema_sqlserver.htm - Consultada: abril/09
- “Oracle database Concept” - http:// downloadwest.oracle.com/docs/cd/B10501_01/server.920/a96524/title.htm consultada: Mayo 2009