

8 views

1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

[Full description](#)



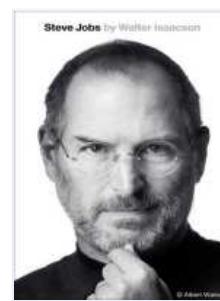
Save

Embed

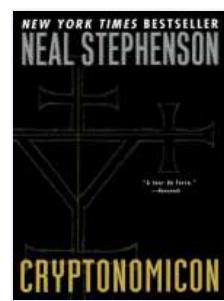
Share

Print

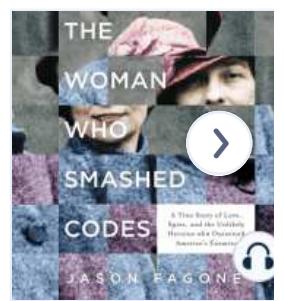
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A Tru



8 views

1 0

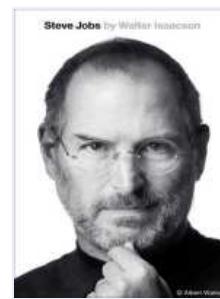
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

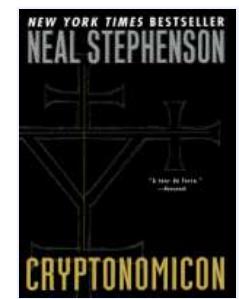
[Full description](#)

   
Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who
Smashed
Codes: A Tru

8 views

1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

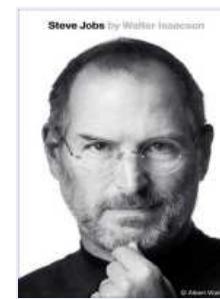
Uploaded by [anon_61286589](#)

[Full description](#)

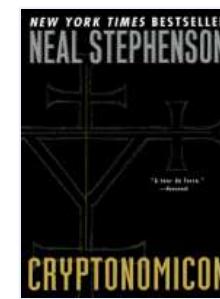


Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Autor: Manuel Angel Torres Remon

© Derecho de autor reservado
Empresa Editora Macro E.I.R.L.

© Derecho de edición, arte gráfico y diagramación reservados
Empresa Editora Macro E.I.R.L.

Edición a cargo de:
Empresa Editora Macro E.I.R.L.
Av. Paseo de la República 5613 - Miraflores
Lima - Perú
📞 (511) 719-9700
✉️ ventas@editorialmacro.com
<http://www.editorialmacro.com>

Primera edición: Octubre 2012 - 1000 ejemplares

Impreso en los Talleres Gráficos de
Empresa Editora Macro E.I.R.L.
Lima - Perú

ISBN Nº 978-612-304-084-0
Hecho el Depósito Legal en la Biblioteca Nacional del Perú Nº 2012-12379

Prohibida la reproducción parcial o total, por cualquier medio o método de este libro sin p
autorización de la Empresa Editora Macro E.I.R.L.

8 views

1 0

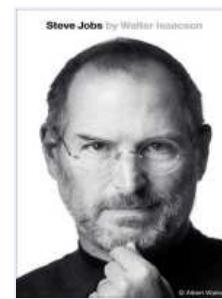
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

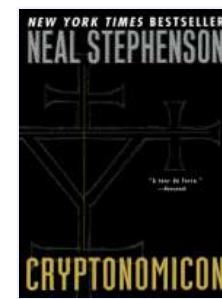
[Full description](#)

 Save  Embed  Share  Print

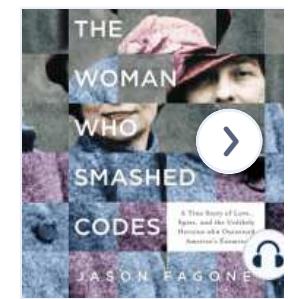
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

MANUEL ANGEL TORRES REMON

Manuel Torres es un consultor dedicado a la docencia de cursos de tecnología como Microsoft SQL Server; ha brindado capacitación en las instituciones más importantes de Lima.

Recibió su formación tecnológica en el Instituto Superior Tecnológico Público Manuel Arevalo Cáceres y también en la Universidad Alas Peruanas de la República del Perú. En ambas instituciones recibió una buena formación profesional, demostrada en las diferentes instituciones en que laboró.

Actualmente se desempeña como consultor tecnológico de grandes empresas como Topware Solutions y como docente en instituciones educativas como el instituto Manuel Arevalo Cáceres y Unimaster de la Universidad Nacional de Ingeniería en todas ellas impartiendo cursos de especialidad en Análisis, Programación y Base de Datos.

Ha publicado el libro Programación VBA con Excel donde expone que los usuarios de Office pueden programar de manera profesional sin necesidad de ser expertos en programación y también la Guía Práctica de Programación VBA con Excel.

Se le puede localizar al email manuel.torresr@hotmail.com o a los teléfonos (511)-982360996396023

8 views

1 0

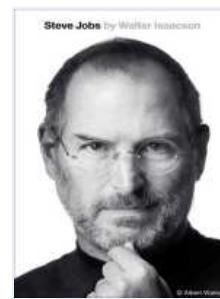
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

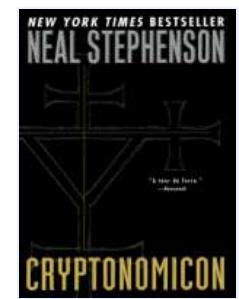
[Full description](#)

   
Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

8 views

1 0

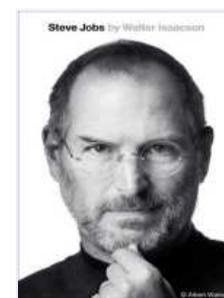
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

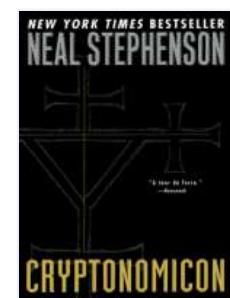
[Full description](#)

 Save  Embed  Share  Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

Agradecimientos

Cada libro desarrollado es una labor donde el autor sacrifica muchas veces la familia, horas de descanso y otras acti- tienen su recompensa cuando el mate- realidad. No cabe duda que me complac con ustedes los casos expuestos ya que en el desempeño de mis labores como c eso este agradecimiento en primer luga MACRO por confiar nuevamente en mi p

8 views

1 0

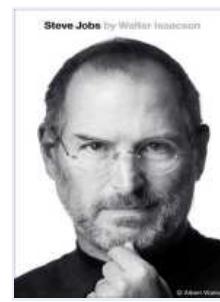
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

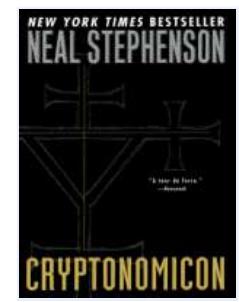
[Full description](#)

   
Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who
Smashed
Codes: A Tru

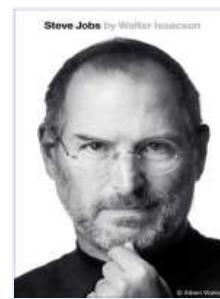
8 views

1 0

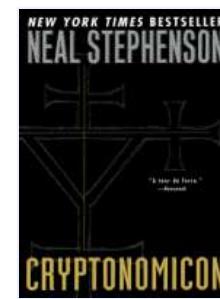
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

Introducción

Server 2012 es un sistema integrado de gestión de base de datos; hoy en día las organizaciones necesitan tener un control automatizado de sus archivos, es decir, una simple acción necesariamente tendrá que ser física en la actualidad se la puede enviar por email; por ejemplo Perú mediante las leyes 26612 y 681, se aprueba el uso de las imágenes como medio de prueba legal, considerándose las mismas con el mismo valor legal que el documento original, por lo tanto pueden dejar de lado la papelería para pasar a los archivos digitales; SQL puede tener el control de sus archivos administrándolos de manera eficaz, rápida y segura.

Programación Transact SQL propone realizar procesos de manera profesional por medio de lenguajes que se ejecutarán tanto en el cliente como en el servidor y que será de gran utilidad. Los lenguajes de Programación siembran la cultura de la programación nativa mientras que el SQL Server propone instrucciones o sentencias para la obtención de resultados, Transact SQL es un lenguaje que no tiene su propia sintaxis ni su propio esquema y le quita un poco de protagonismo a los lenguajes de programación usando sus estructuras que son entendidas en el motor de base de datos de SQL Server 2012.

Transact SQL Server 2012 tiene como fundamento primordial gestionar la información de una forma más eficiente y segura en una base de datos sin dejar toda la responsabilidad de la gestión a los lenguajes de programación, más bien usa de ellos sus estructuras como el If o While para procesar reglas de negocio.

La versión de SQL Server 2012 no presenta grandes cambios en el trabajo de programación Transact-SQL así es que si no tiene esta versión podrá ejecutar los casos desarrollados en SQL Server 2008.

8 views

1 0

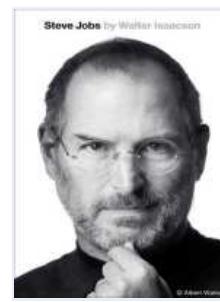
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

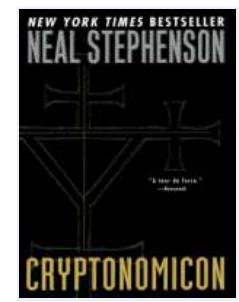
[Full description](#)

   
Save Embed Share Print

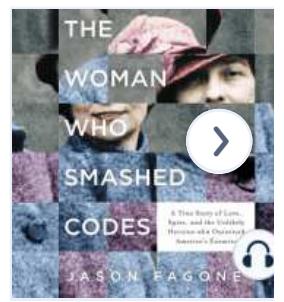
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

8 views

1 like

0 comments

eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

[Full description](#)



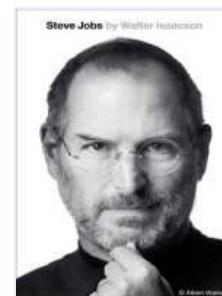
Save

Embed

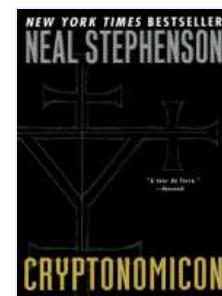
Share

Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

A quién está dirigido este libro

Los administradores de base de datos ya tienen definida su labor frente a una base de datos. Este material propone ver más allá de la administración y prepararlos al desarrollo, no se impone que solo se utilice una aplicación como lo hace Visual Studio o Java pero se podrá programar consultas e implementar de manera profesional los procedimientos almacenados, las funciones, los cursor y mucho más que verá en este material.

Un pre-requisito para este material es tener un conocimiento básico de los comandos e instrucciones de SQL Server, además de conocer el objetivo de las estructuras selectivas o repetitivas suficiente para que este material le sea útil, mejor dicho es el siguiente nivel de un administrador que administra una base de datos.

Este material no pretende ser una introducción a la programación Transact más bien se trata de una programación avanzada de SQL Server exigiendo al motor de base de datos la responsabilidad de compilar una porción de código Transact.

8 views

1 0

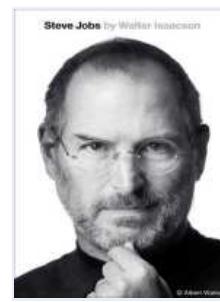
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

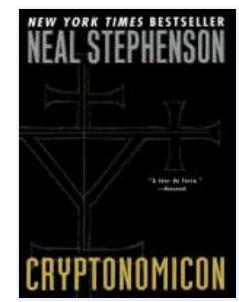
[Full description](#)

   
Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

8 views

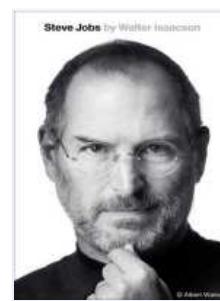
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

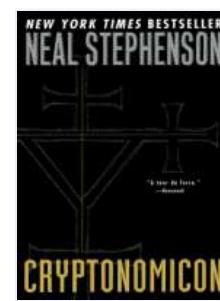
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A Tru

Índice

Capítulo 1

Microsoft SQL Server 2012
1.1. Definición de Microsoft SQL Server 2012
1.2. Versiones SQL Server 2012
1.3. Características de SQL Server 2012
1.4. Preparando la instalación de SQL Server 2012
1.5. Pre-requisitos para la instalación de SQL server 2012.....
1.6. Pantalla inicial de autorun del CD de instalación.....
1.7. Acesso al SQL Server 2012
1.8. Configuración de fuente para el entorno de Trabajo.....
1.9. Lenguaje de Definición de Datos (LDD)
1.10. Sentencia CREATE
1.11. Sentencia ALTER
1.12. Sentencia DROP

Capítulo 2

Gestión de base de datos
2.1. Qué es una Base de Datos
2.2. Objetivos de los sistemas de base de datos
2.3. Las bases de datos en SQL Server
2.4. Estructura de una Base de Datos
2.5. Archivos y grupos Físicos de la Base de Datos
2.6. Motor de Base de Datos
2.7. Crear una base de datos
2.8. Enunciado: Reserva de vuelos
2.9. Separar y Adjuntar una Base de Datos
2.10. Procedimiento almacenado sp_detach_db

8 views

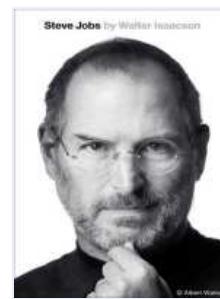
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

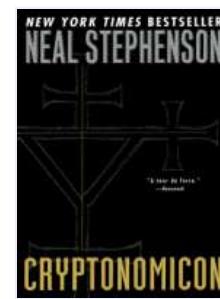
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

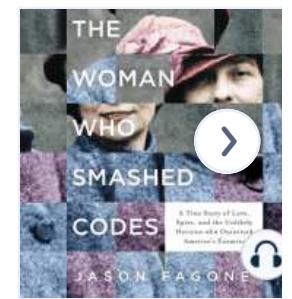
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

- [2.16. Implementación de Tablas con Propietario DBO](#)
- [2.17. Implementación de tablas con esquemas](#)
- [2.18. Definición de las llaves primarias y foráneas](#)
- [2.19. Restricciones de los campos: unique, check y default.....](#)
- [2.20. Esquema de la base de datos AGENCIA para el uso de los casos desarrollados..](#)

Capítulo 3

Lenguaje de manipulación de datos (DML)

- [3.1. Introducción a la manipulación de datos](#)
- [3.2. Insertar registros con INSERT INTO](#)
- [3.3. Modificación y actualización de datos de una tabla con UPDATE](#)
- [3.4. Eliminación de registros de una tabla con DELETE](#)
- [3.5. Declaración general del comando SELECT para la recuperación de registros](#)
- [3.6. Los operadores en SQL Server 2012](#)
- [3.7. Combinación de tablas Join, Left Join, Right Join](#)
- [3.8. Recuperación de datos agrupados Group By, Having y las funciones agregadas SUM, COUNT, MAX, MIN y AVG](#)
- [3.9. Funciones Agregadas](#)
- [3.10. Agregar conjunto de resultados: UNION](#)
- [3.11. Resumen de datos: Operador Cube y ROLLUP](#)
- [3.12. Declaración MERGE](#)

Capítulo 4

Programación Transact SQL.....

- [4.1. Introducción](#)
- [4.2. Fundamentos de Programación Transact SQL](#)
- [4.3. Variables, Identificadores](#)
- [4.4. Funciones CAST y CONVERT](#)
- [4.5. Estructuras de Control](#)

8 views

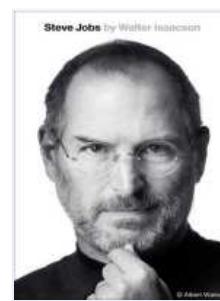
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

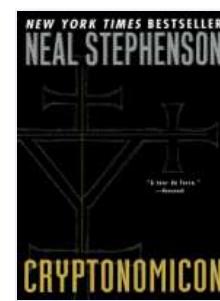
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True

- 4.12. Función Raiserror
- 4.13. Implementación de cursos
- 4.14. Funciones
- 4.15. Funciones del Sistema
- 4.16. Funciones definidas por el usuario
- 4.17. Procedimientos Almacenados
- 4.18. Procedimientos Almacenados del sistema
- 4.19. Instrucción EXECUTE y SP_EXECUTESQL
- 4.20. Procedimientos Almacenados definidos por el usuario
- 4.21. Procedimientos almacenados con parámetros de entrada
- 4.22. Procedimientos almacenados con parámetros de entrada y salida
- 4.23. Modificar la implementación de un procedimiento almacenado
- 4.24. Eliminar procedimientos almacenados
- 4.26. Visualizar la implementación de un procedimiento almacenado
- 4.27. Procedimientos almacenados y cursos
- 4.28. Transacciones en Transact SQL
- 4.29. Triggers
- 4.30. Casos desarrollados para Triggers DML
- 4.31. Casos desarrollados para Triggers DDL.....

Capítulo 5

- XML con SQL**
 - 5.1. Introducción
 - 5.2. Modelo de datos relacionales o XML
 - 5.3. Ventajas de almacenar valores en XML
 - 5.4. Elección de la tecnología XML
 - 5.5. Tipo de dato XML
 - 5.6. Columnas y Variables XML.....
 - 5.7. FOR XML y OPENXML

8 views

1 0

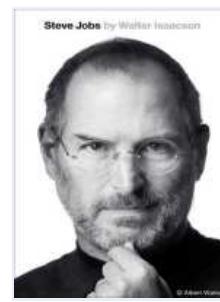
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

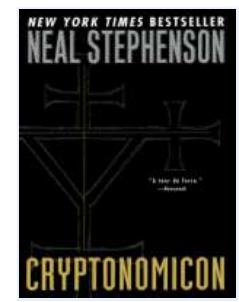
[Full description](#)

   
Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

8 views

 1  0

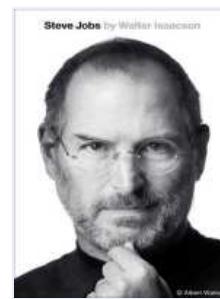
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by **anon_61286589**

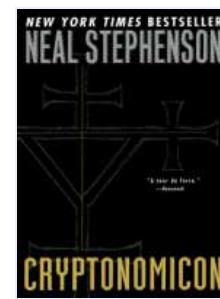
Full description



RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A Tri

Microsoft SQL Server 2012

CAPACIDAD:

El lector podrá reconocer las diferencias entre las versiones de SQL Server, además de con casos desarrollados las sentencias pertenecientes al lenguaje de definición de datos.

Primero se reconocerá las versiones de SQL Server y luego se procederá a mostrar pasos para la instalación de SQL Server 2012. Al final del capítulo se presentarán casos desarrollados para cada sentencia del lenguaje de definición de datos.

CONTENIDO:

- Definición de Microsoft SQL Server 2012
 - Versiones SQL Server 2012
 - Características de SQL Server 2012
 - Preparando la instalación de SQL Server 2012

8 views

1 0

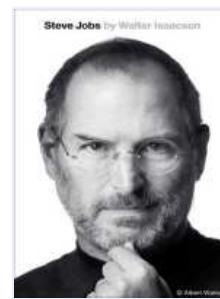
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

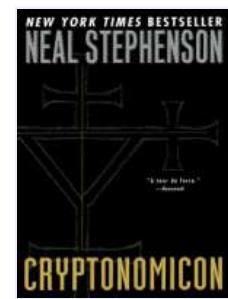
[Full description](#)

   
Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who
Smashed
Codes: A Tru

8 views

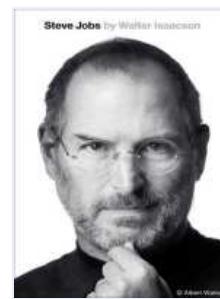
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

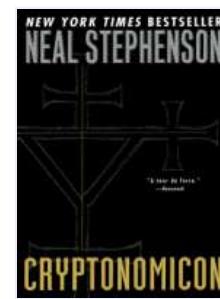
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of Mathematics

CAP. 1: MICROSOFT SQL SERVER 2012

1.1. DEFINICIÓN DE MICROSOFT SQL SERVER 2012

Microsoft SQL Server es un sistema para la gestión de bases de datos producidos por Microsoft en el modelo relacional. Sus lenguajes para consultas son Transact-SQL y ANSI SQL. Microsoft Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, PostgreSQL o MySQL.

Las principales características de Microsoft SQL Server 2012 son:

- Ofrece a los desarrolladores de base de datos un soporte potente de transacciones.
- Soporte de procedimientos almacenados.
- Todas las versiones de SQL Server presentan un entorno gráfico de administración del motor de base de datos, 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 los terminales o clientes de la red sólo acceden a la información.
- Permite la administración de información de otros servidores de datos y no necesita el mismo sistema operativo.

A partir de la versión 2005 se incluyó dentro del sistema la versión reducida que la llamaron el mismo motor de base de datos, pero orientado a proyectos más pequeños, reduciendo el espacio en disco y lo engorroso que podría ser instalar la versión completa del SQL Server, a esta le llamo SQL Express Edition, que se distribuye en forma gratuita desde la página oficial de Microsoft (<http://msdn.microsoft.com>). En la versión 2008 sale una nueva utilidad dentro de la administración de bases de datos que lleva a gestionar bases de datos distribuidas.

1.2. VERSIONES SQL SERVER 2012

Veamos una tabla de comparación Versión, Año y Nombre clave en donde veremos la evolución de SQL Server hasta la última versión 2012.

VERSIÓN	AÑO DE LANZAMIENTO	NOMBRE DEL PROYECTO
1.0	1989	SQL
4.21	1993	SEQUEL
6.0	1995	SQL95
6.5	1996	Hydra
7.0	1998	Sphinx
8.0	2000	Shiloh (SQL Server 2000)
	2003	Liberty (SQL Server 2000 64 bit)
9.0	2005	Yukon (SQL Server 2005)

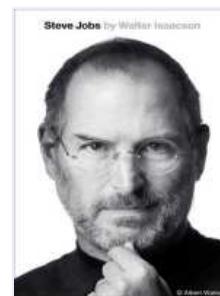
8 views

1 0

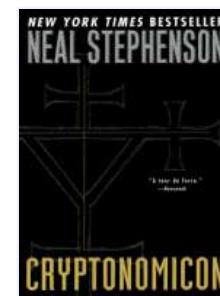
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

18 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

1.3. CARACTERÍSTICAS DE SQL SERVER 2012

Las principales características y los puntos más destacables de SQL Server 2012 son:

- Mayor disponibilidad. Alcance de los 9s exigidos de disponibilidad y el nivel de protección que requiere su organización con AlwaysOn, que ahora ofrece más funcionalidades que CTP1 y permite a los clientes disfrutar un nivel aún mayor de flexibilidad y valor de negocio.
- Los análisis más avanzados. Descubra la potencia escondida en sus datos con análisis potentes y una exploración de datos más rápida a través de toda su organización. Esto está disponible para los clientes de SQL Server por primera vez.
- Datos creíbles y consistentes. Ofrezca a sus usuarios una visión consistente de la información de los orígenes de datos muy diversos con el Modelo de Semántica de BI (BI Semantic Model). Un modelo unificado y común para las aplicaciones de Business Intelligence. La calidad de los datos no dejará de ser una tarea habitual gracias al complemento Master Data Services para nuevos Data Quality Services, que se integran con proveedores de datos externos disponibles en el Datamarket de Windows Azure Marketplace. Los clientes pueden probar ya esta función.
- Una experiencia de desarrollo productiva. Aumente la productividad de sus departamentos de desarrollo tanto en sus propias instalaciones de servidor como en la nube, con el Componente de Aplicación de Capa de Datos (DAC, Data-tier Application Component) que establece una paridad con SQL Azure y SQL Server Data Tools para lograr una experiencia de desarrollo moderna y consistente en funciones de bases de datos, Business Intelligence y en la nube. Además, los clientes de SQL Server Express Edition pueden probar una nueva versión LocalDB para instalaciones sin configuración.

1.4. PREPARANDO LA INSTALACIÓN DE SQL SERVER 2012

SQL Server presenta ediciones de 32 y 64 bits para lo cual se recomienda tener las siguientes consideraciones antes de empezar la instalación del producto:

- Se recomienda instalar el SQL Server 2012 en equipos con formatos NTFS por ser más seguros que los de formato FAT32; pero finalmente es sólo recomendación ya que también se puede instalar en este último.
- El programa de instalación de SQL Server 2012 bloqueará las instalaciones en unidades que tienen sólo lectura, asignadas o comprimidas.
- SQL Server 2012 requiere que se instale una actualización para asegurarse de que se instalará correctamente el componente de Visual Studio. El programa de instalación de SQL Server 2012 comprueba la presencia de esta actualización y, a continuación, le exige que descargue la actualización antes de continuar con la instalación de SQL Server. Para evitar la instalación de SQL Server 2012 durante la instalación de SQL Server, puede descargar e instalar la actualización antes de comenzar la instalación de SQL Server.

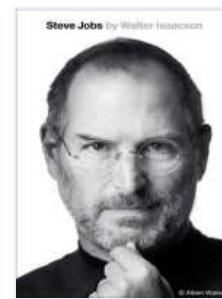
8 views

1 0

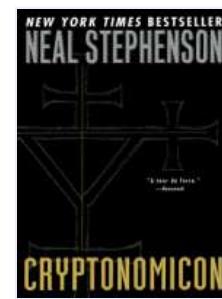
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

1.5. PRE-REQUISITOS PARA LA INSTALACIÓN DE SQL SERVER 2012

◎ HARDWARE

Memoria: recomendado

- SQL Server Express 1GB
- Todas las demás versiones 4GB

Procesador: mínimo

- | | |
|------------------|--------------|
| ● Procesador X86 | : 1GHZ |
| ● Procesador X64 | : 2GHZ a más |

Disco Duro: mínimo 6GB de espacio libre

A continuación listaremos la distribución de espacios requeridos por características de 2012:

● Motor de Base de datos	: 811MB
● Servicio de Análisis y archivo de datos	: 345MB
● Servicio de Reportes y administración de Informes	: 304MB
● Servicios de Integración	: 591MB
● Servicios de Datos Maestros	: 243MB
● Componentes de Cliente	: 1.78GB
● Libros en pantalla de SQL Server	: 375KB

◎ FRAMEWORK

.NET 3.5 SP1 es un requisito de SQL Server 2012 al seleccionar el Motor de base de dato Services, Replicación, Data Quality Services, Master Data Services o SQL Server Manager y el programa de instalación de SQL Server ya no lo instala. Si el programa de instalación en un equipo con el sistema operativo Windows Server 2008 R2 SP1, debe habilitar .NET 3.5 SP1 antes de instalar SQL Server 2012.

.NET 4.0 es un requisito para SQL Server 2012. SQL Server instala .NET 4.0 durante instalación de características. El programa de instalación de SQL Server instala los componentes de software requeridos por el producto:

- .NET Framework 4 1
- SQL Server Native Client
- Archivos auxiliares para la instalación de SQL Server

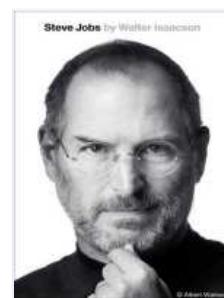
8 views

1 0

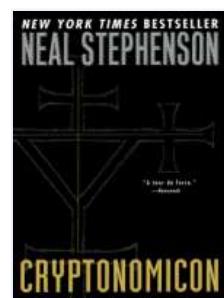
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

20

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

● RED

Los sistemas operativos admitidos para SQL Server 2012 tienen software de red integrado que admite instancias con nombre y predeterminadas de una instalación independiente. Los sistemas operativos admiten los siguientes protocolos de red:

- Memoria compartida
- Canalizaciones con nombre
- TCP/IP
- VIA

● VIRTUALIZACIÓN

SQL Server 2012 se admite en entornos de máquina virtual que se ejecuten en el rol Hyper-V. Se admite en las ediciones Standard, Enterprise y Datacenter de Windows Server 2008 SP2 y las ediciones Standard, Enterprise y Datacenter de Windows Server 2008 R2 SP1.

Además de los recursos requeridos por la partición primaria, a cada máquina virtual (secundaria) se le deben proporcionar suficientes recursos de procesador, memoria y disco para su instancia de SQL Server 2012.

En el rol Hyper-V de Windows Server 2008 SP2 y Windows Server 2008 R2 SP1, se puede asignar un máximo de cuatro procesadores virtuales a las máquinas virtuales que ejecuten las versiones 32 bits o 64 bits de Windows Server 2008 SP2 o Windows Server 2008 R2 SP1.

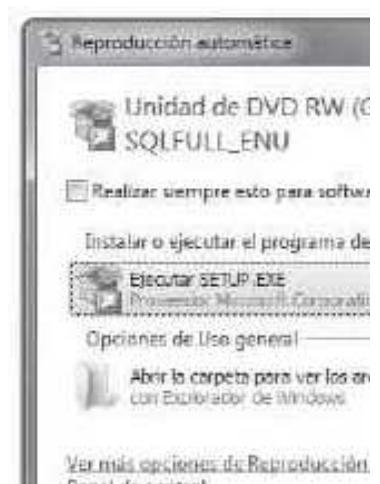
● NAVEGADOR

Se requiere Internet Explorer 7 o una versión posterior para Microsoft Management Console, las Herramientas de datos de SQL Server (SSDT), el componente Diseñador de informes de SQL Server, los servicios de Ayuda y la Ayuda HTML.

1.6. PANTALLA INICIAL DE AUTORUN DEL CD DE INSTALACIÓN

Para obtener el software puede descargar la versión de prueba desde la página oficial www.microsoft.com.

El Asistente para instalación se ejecutara después de seleccionar el archivo SETUP.EXE



8 views

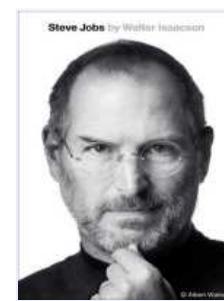
1 like

0 comments

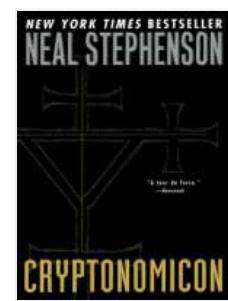
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



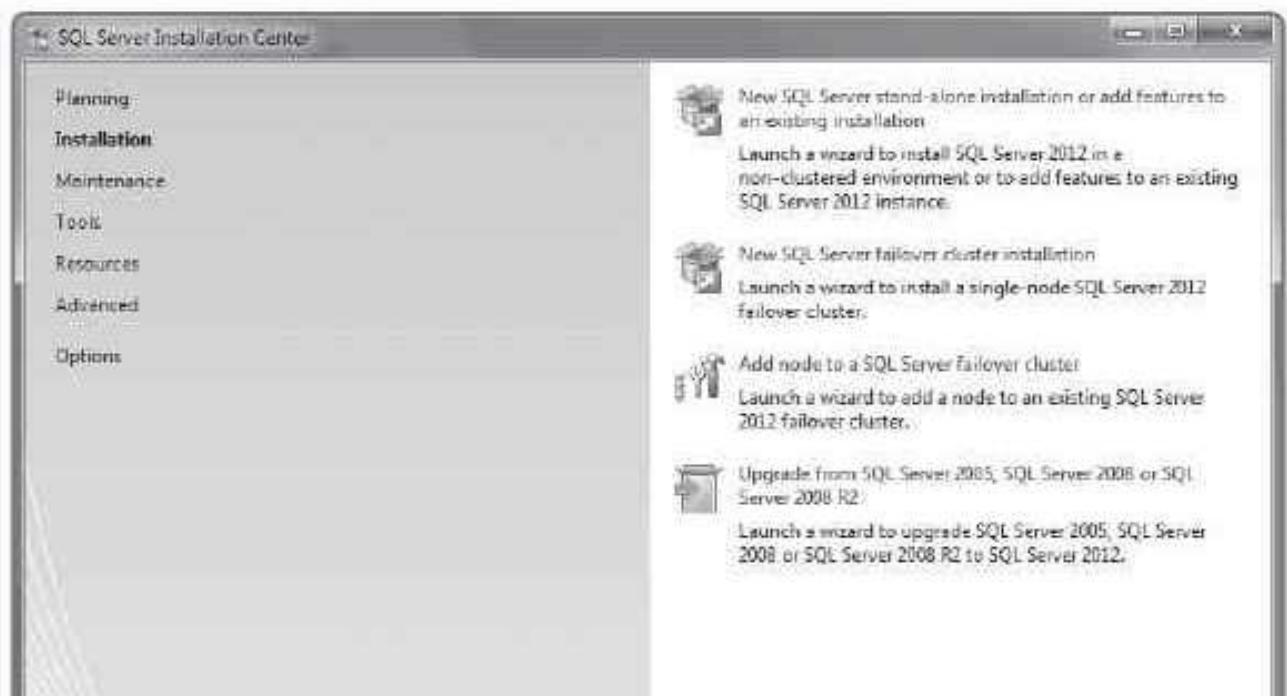
The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

CAP. 1: MICROSOFT SQL SERVER 2012

Luego se presenta el Centro de Instalación de SQL Server, a partir de aquí se tendrá que seleccionar las características propias de SQL Server 2012. Selección **Installation** de la imagen siguiente:



Seguidamente seleccione: **New SQL Server stand-alone installation or add features to installation**, para poder seleccionar una nueva instancia de la instalación de SQL Server 2012.



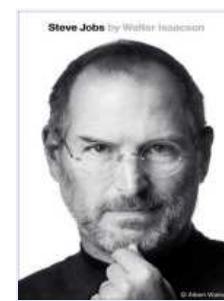
8 views

1 0

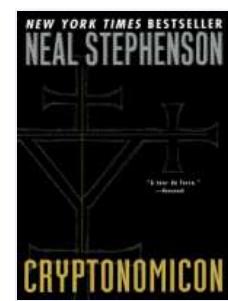
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon

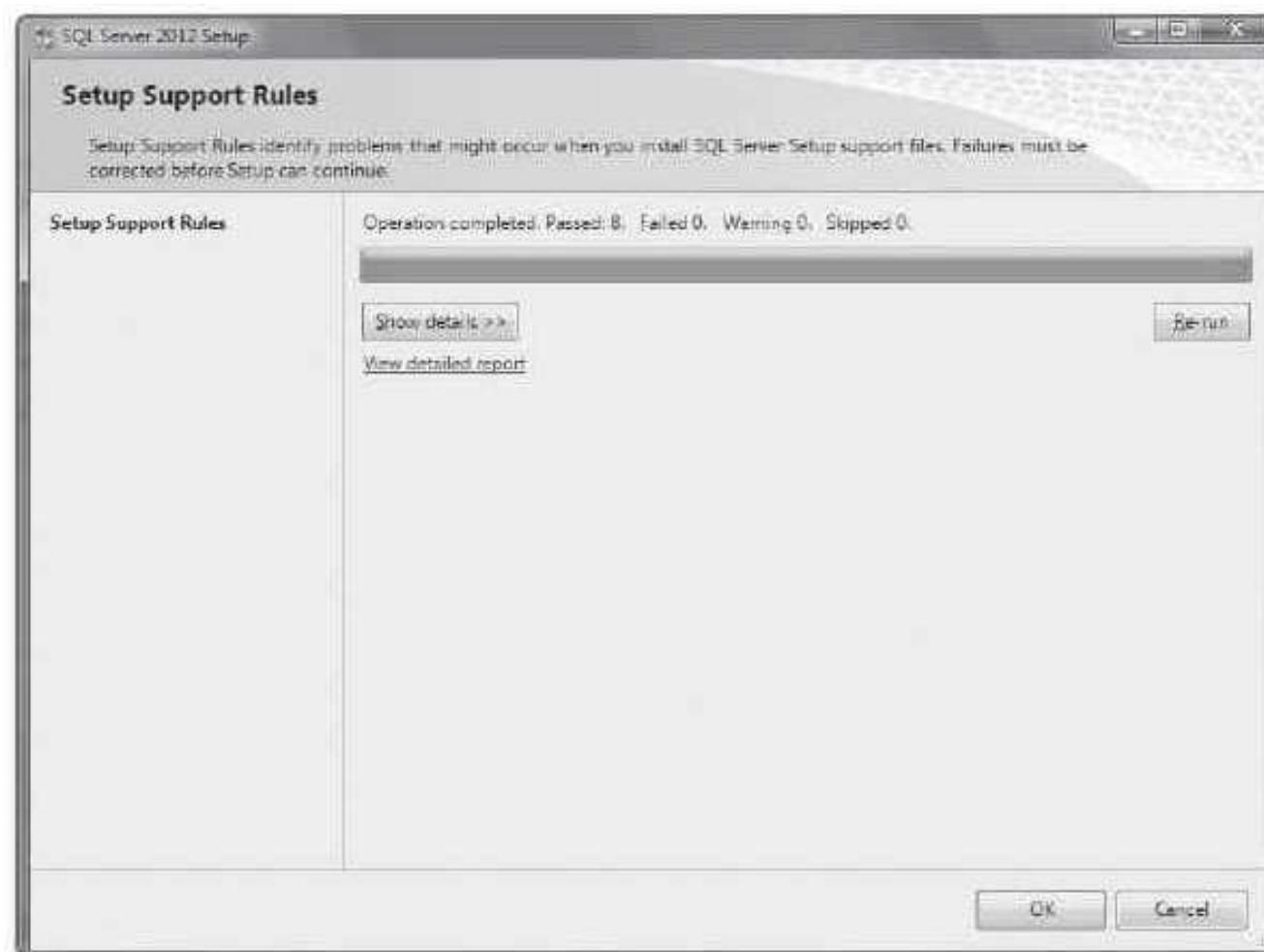


The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

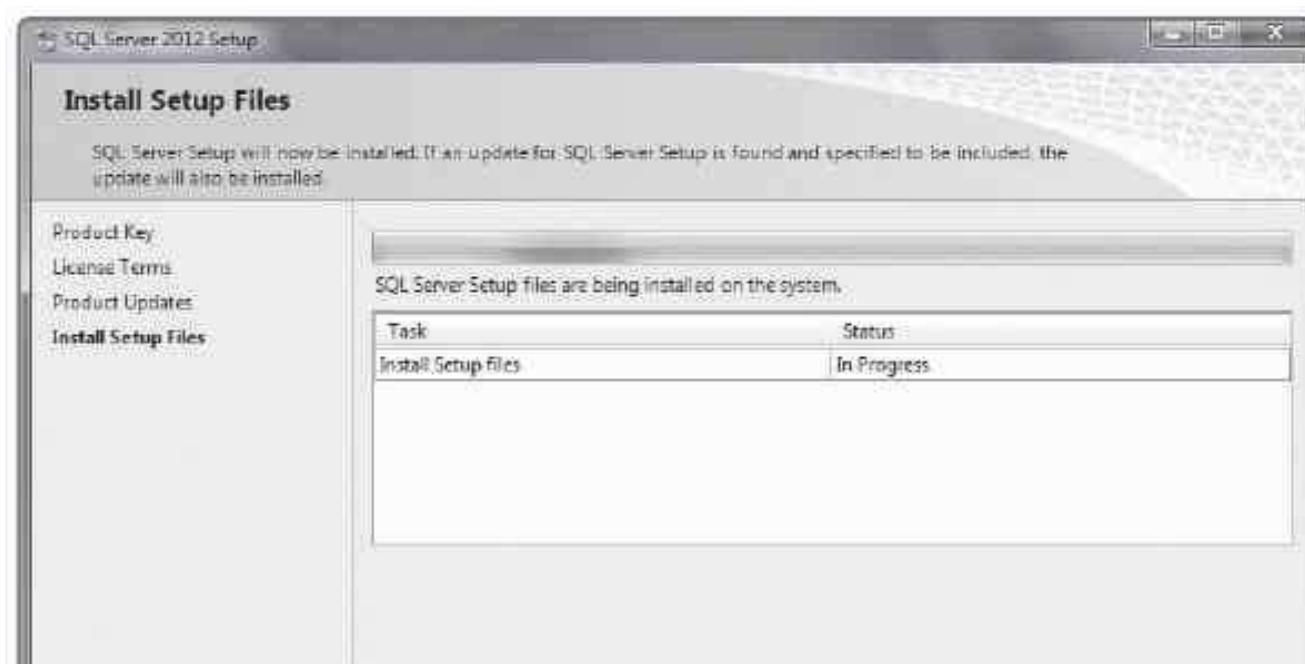
22

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Como vemos en la siguiente imagen sólo será cuestión de esperar las comprobaciones pre Server 2012, si hubiera algún problema durante esta búsqueda tendrá que subsanarla continuar con la misma, si desea ver cuáles son los errores presione [Show details>>](#).



Toda vez verificando las reglas de SQL Server 2012, el asistente comprobará si todas las especias están correctas.



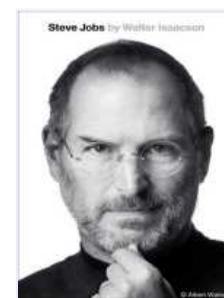
8 views

1 0

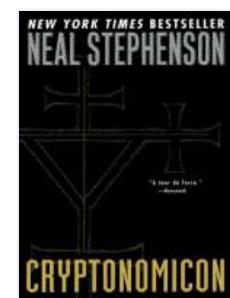
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



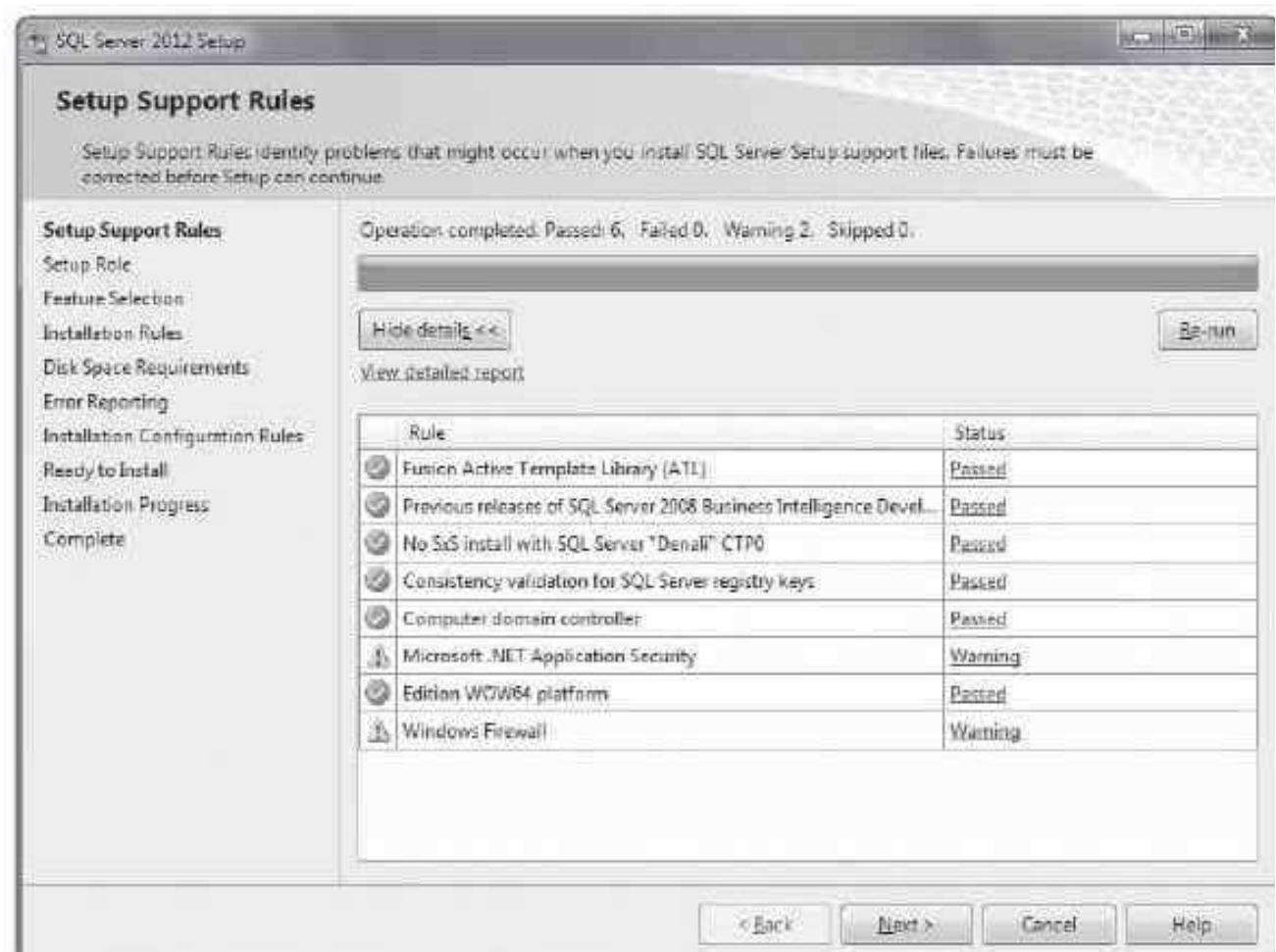
Cryptonomicon



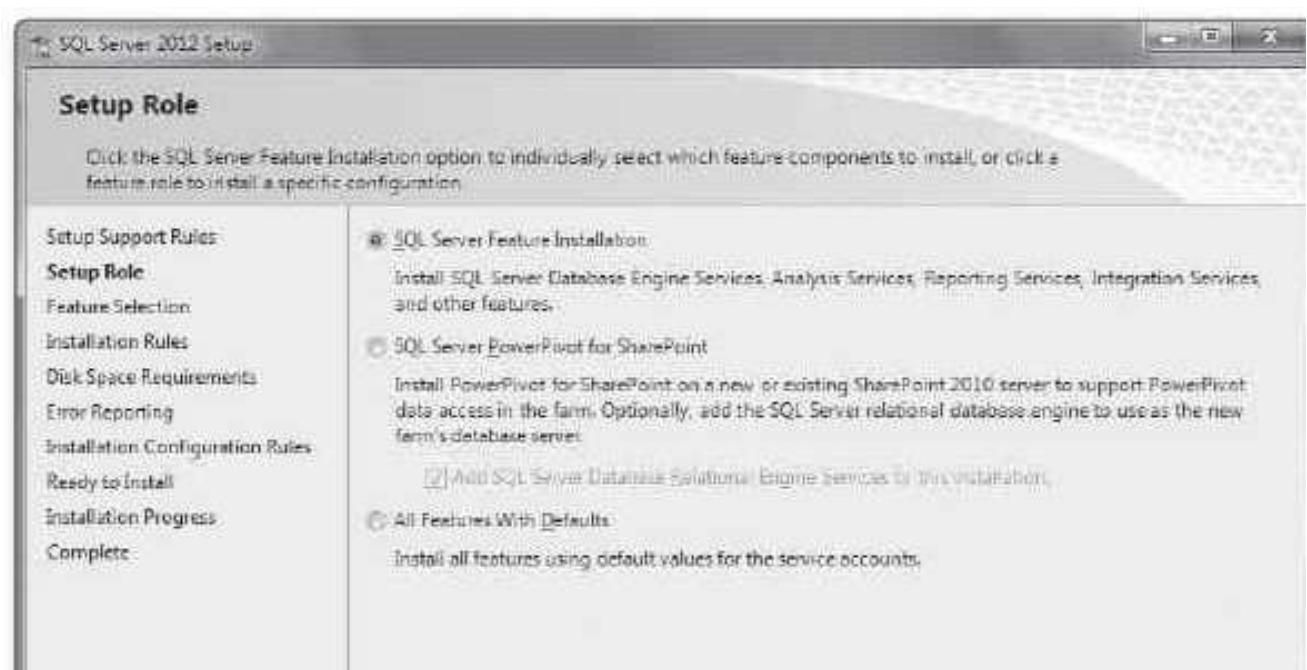
The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in History

CAP. 1: MICROSOFT SQL SERVER 20

Consideremos el caso que no tenga instalado en su computador Visual NET ni un Firewall entonces se mostrara el siguiente resumen:



En este caso estos errores de cuidado no son relevantes en la instalación de SQL Server así que continuar con la instalación presionando el botón **Next >**. Seguidamente en la ventana Se debe seleccionar **SQL Server Feature Installation**.



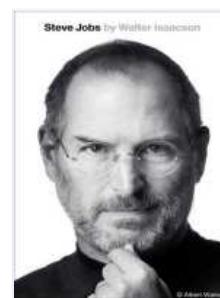
8 views

1 0

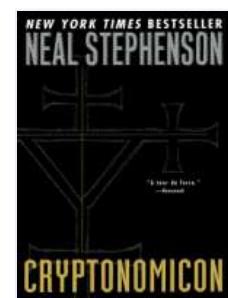
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Unlikely Heroine Who Outwitted America's Greatest Cryptographer

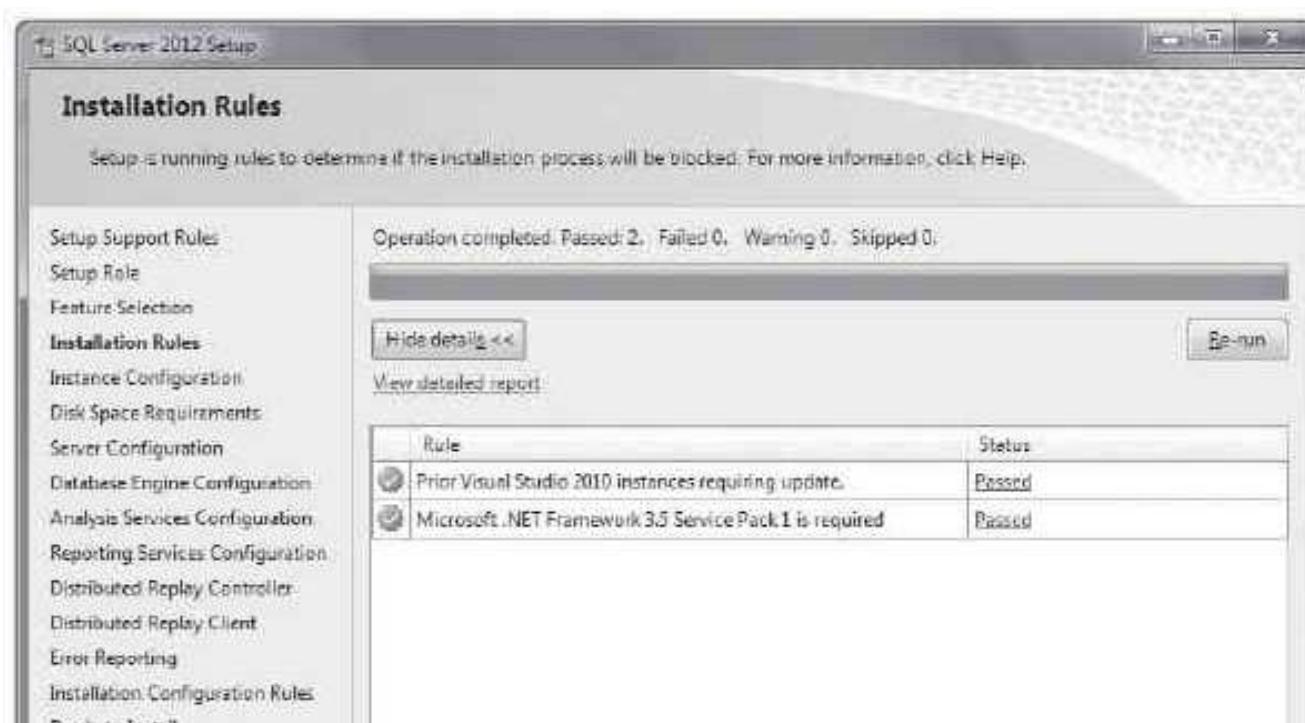
24

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Seguidamente debemos seleccionar las características de la instalación, es recomendado **Select All**:



Luego se procede a instalar los requerimientos básicos de SQL Server 2012, en este caso no hay errores ya que en esta instalación no hay Visual NET.



8 views

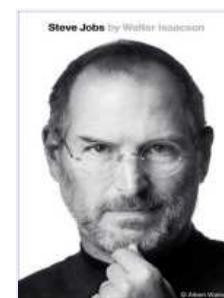
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

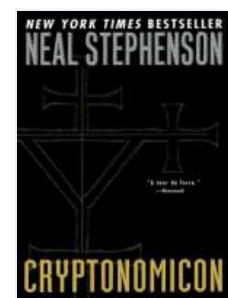
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



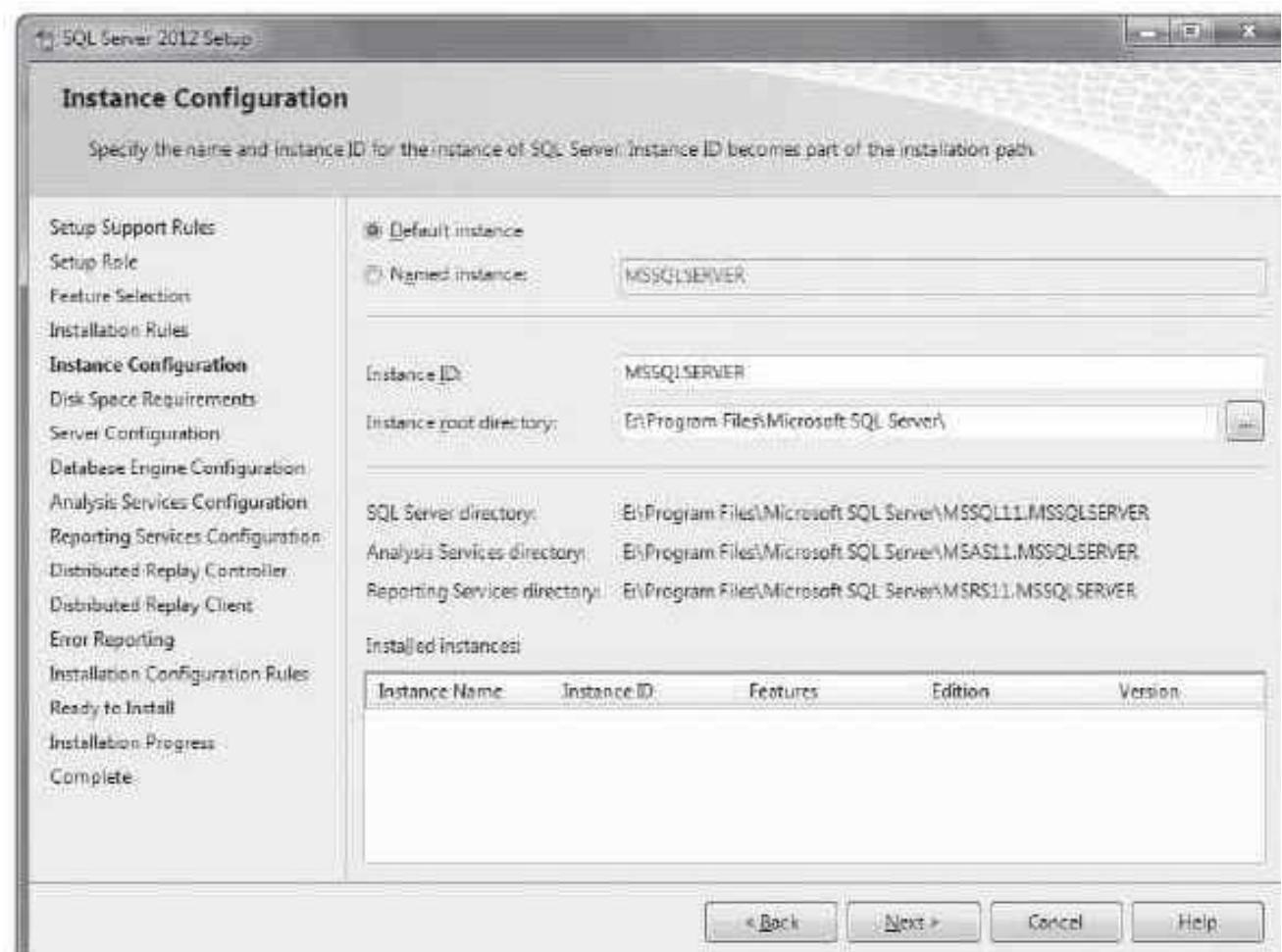
Cryptonomicon



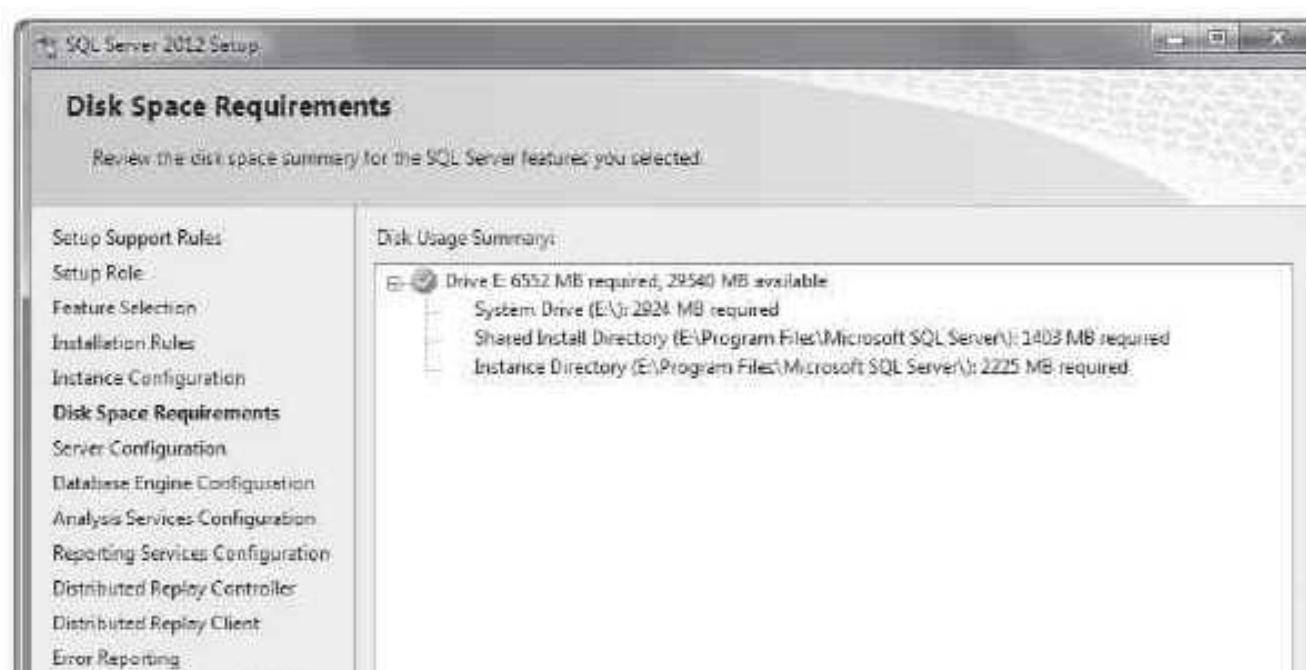
The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in History

CAP. 1: MICROSOFT SQL SERVER 2012

En la ventana **Instance Configuration** se debe configurar el tipo de instancia a instalar, lugar donde se grabaran los archivos de SQL Server 2012, normalmente se direcciona en Programa.



El asistente verificará los requerimientos de espacio en disco antes de proceder con la instalación de los archivos en la ubicación especificada en el paso anterior.



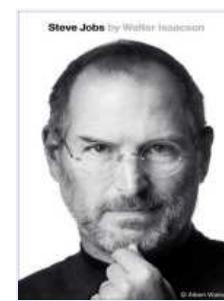
8 views

1 0

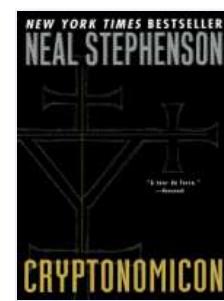
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon

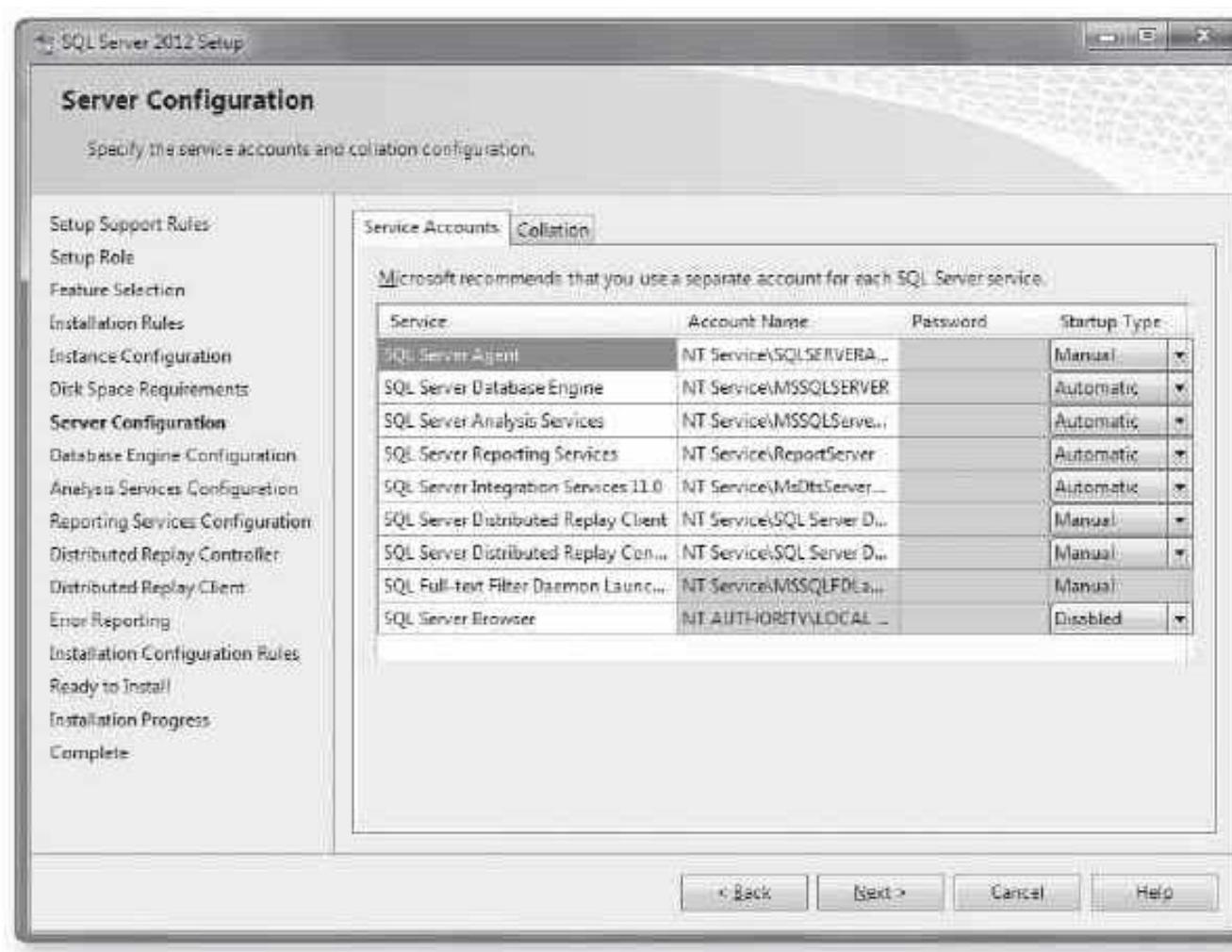


The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in Tech History

26

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En la siguiente ventana sólo presionar **Next** puesto que el asistente de SQL Server 2012 c forma de instalarlo.



En la siguiente ventana se debe especificar el modo de autenticación que tendrá el acceso base de datos SQL Server 2012, si está instalando en un computador personal o portátil se usar Windows Authentication Mode, si se encuentra en una organización y las bases vulnerables entonces use Mixed Mode para poder definir una clave de acceso a los objetos



8 views

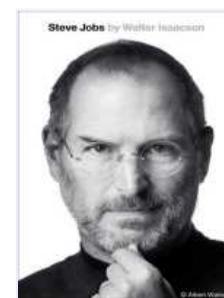
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

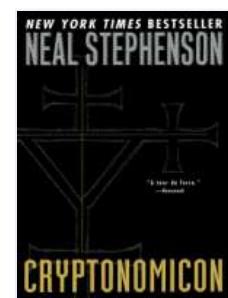
Uploaded by [anon_61286589](#)[Full description](#)

 Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in History

CAP. 1: MICROSOFT SQL SERVER 20

Antes de presionar **Next >** debe seleccionar **Add Current User** para determinar el usuario activo.



En la ventana **Analysis Services Configuration** también se debe agregar el usuario activo como **Current User**.



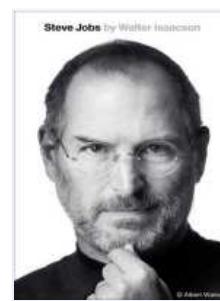
8 views

1 0

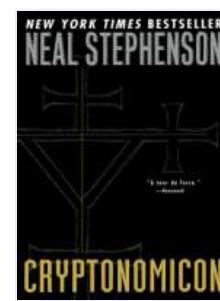
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

28

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Luego en la ventana **Reporting Services Configuration** debe seleccionar **Install and Configure** para iniciar con la instalación del producto.



En la ventana **Distributed Replay Controller** debe seleccionar **Add Current** para seleccionar el usuario de distribución.



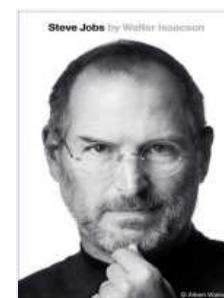
8 views

1 0

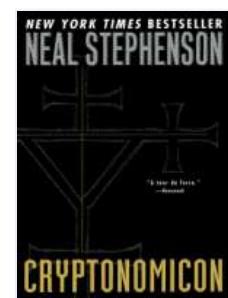
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
 [Save](#)
 [Embed](#)
 [Share](#)
 [Print](#)

RELATED TITLES



Steve Jobs



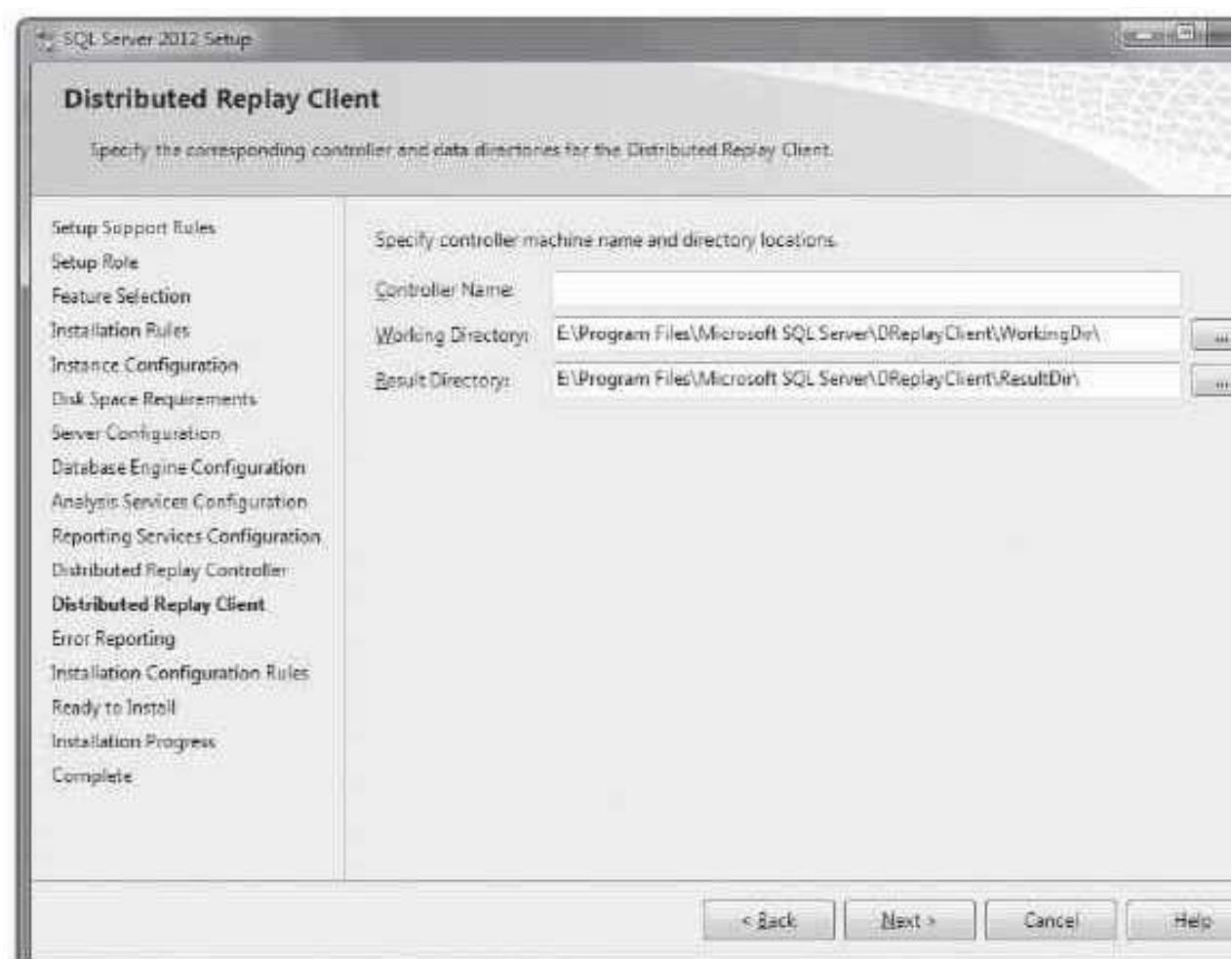
Cryptonomicon



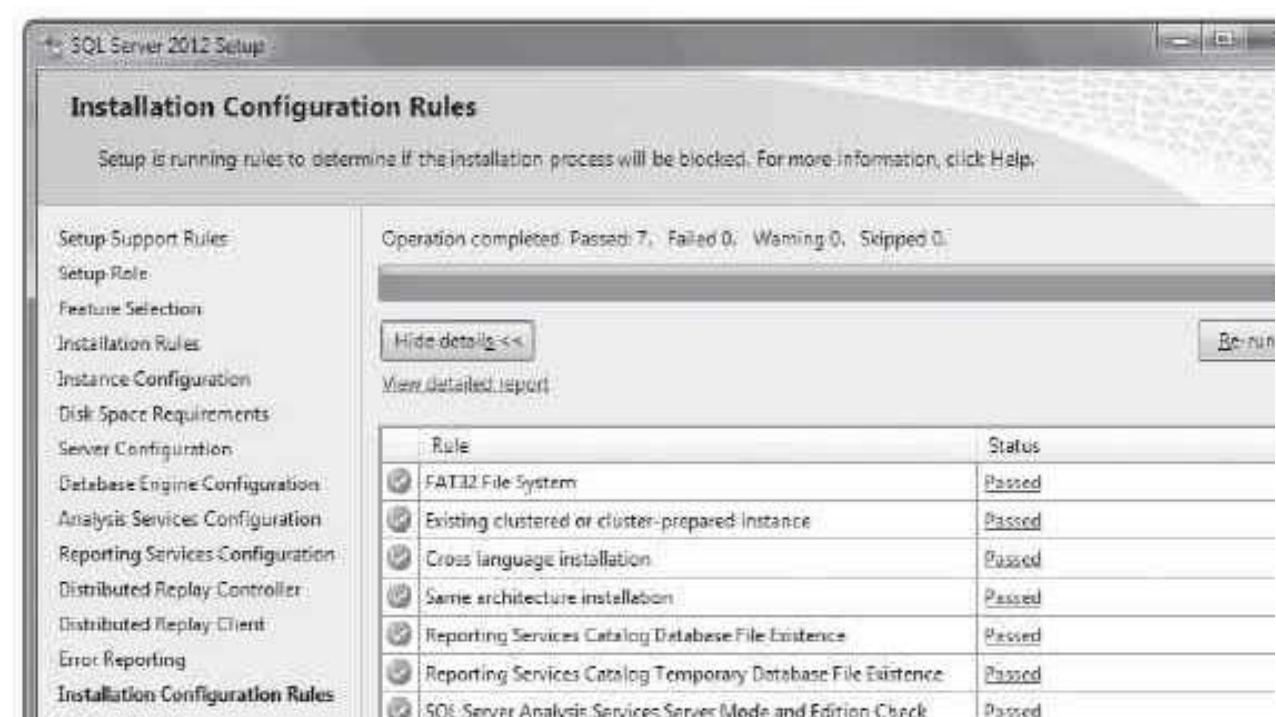
The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Unlikely Heroine Who Outwitted America's Safest Man

CAP. 1: MICROSOFT SQL SERVER 2012

Seguidamente se debe especificar el directorios de los archivos de control, estos ya se predeterminados; por lo tanto, sólo presione **Next>**.



En la ventana Installation Configuration Rules, el asistente verificará si todo es correcto, seleccionar **Next>** para comenzar con la instalación.



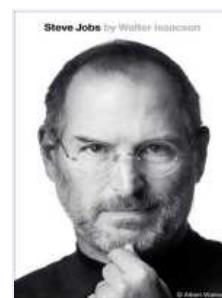
8 views

1 0

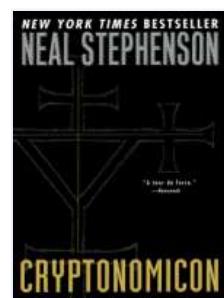
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon

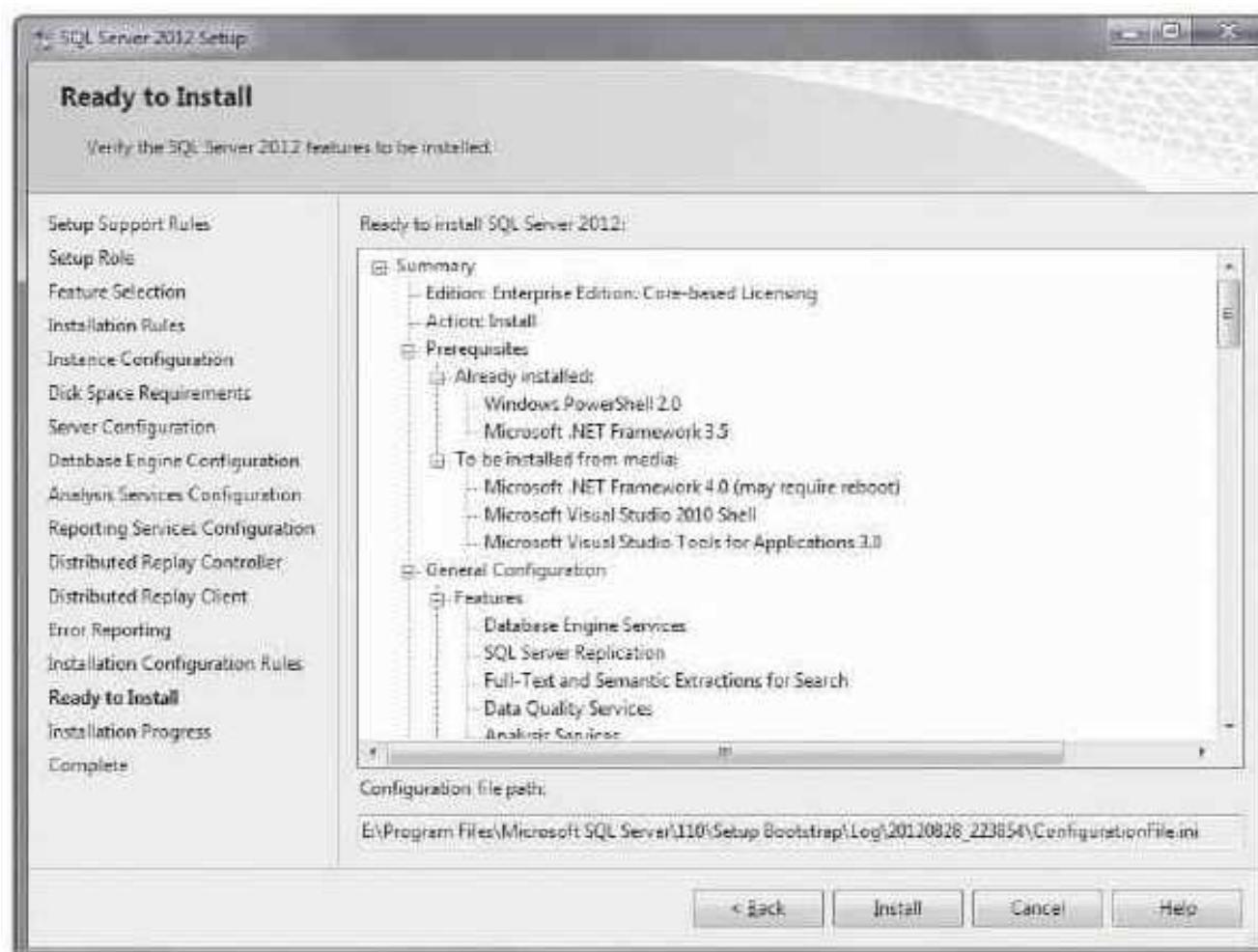


The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

30

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En la ventana **Ready to Install** estamos listos para iniciar la instalación, presionando el botón **Install**.



La instalación está en progreso.



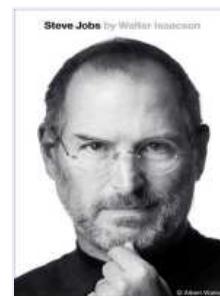
8 views

1 0

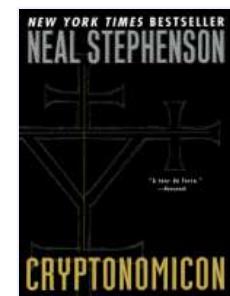
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

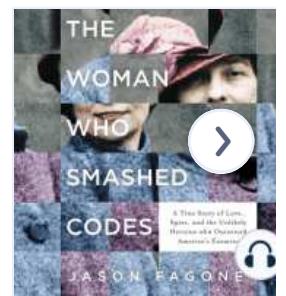
RELATED TITLES



Steve Jobs



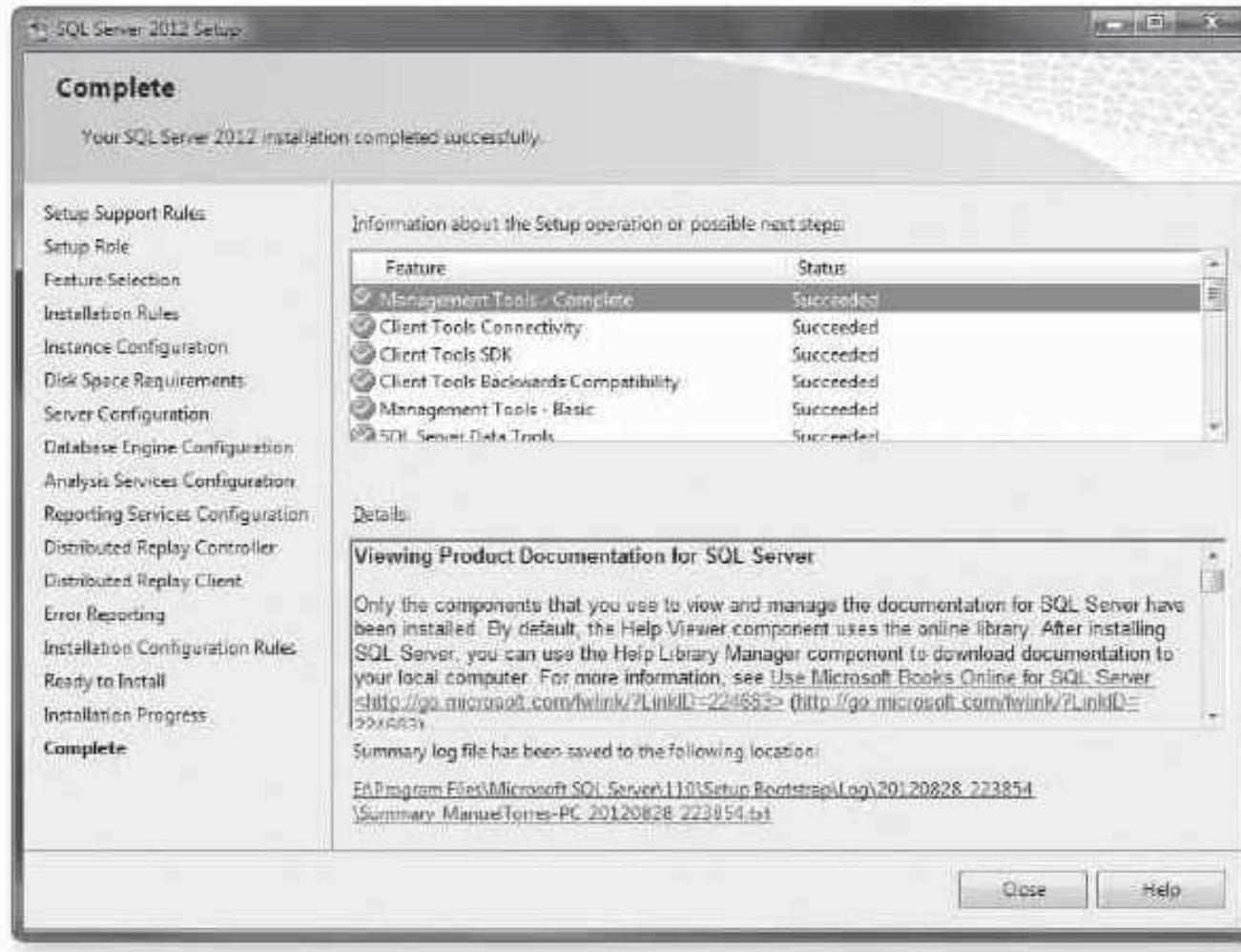
Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 1: MICROSOFT SQL SERVER 2012

Finalmente, el asistente de instalación muestra la ventana de instalación completa, aquí presionar **Close** para cerrar el asistente.



1.7. ACESSO AL SQL SERVER 2012

Desde el botón iniciar del Windows Seven debe seleccionar > Todos los programas > Microsoft SQL Server 2012 > SQL Server Management Studio.



Al iniciar la aplicación se muestra la pantalla inicial de SQL Server 2012.

8 views

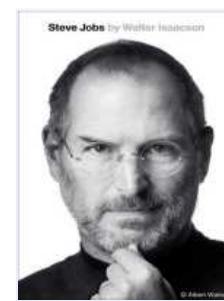
1 like

0 comments

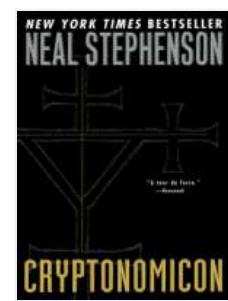
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



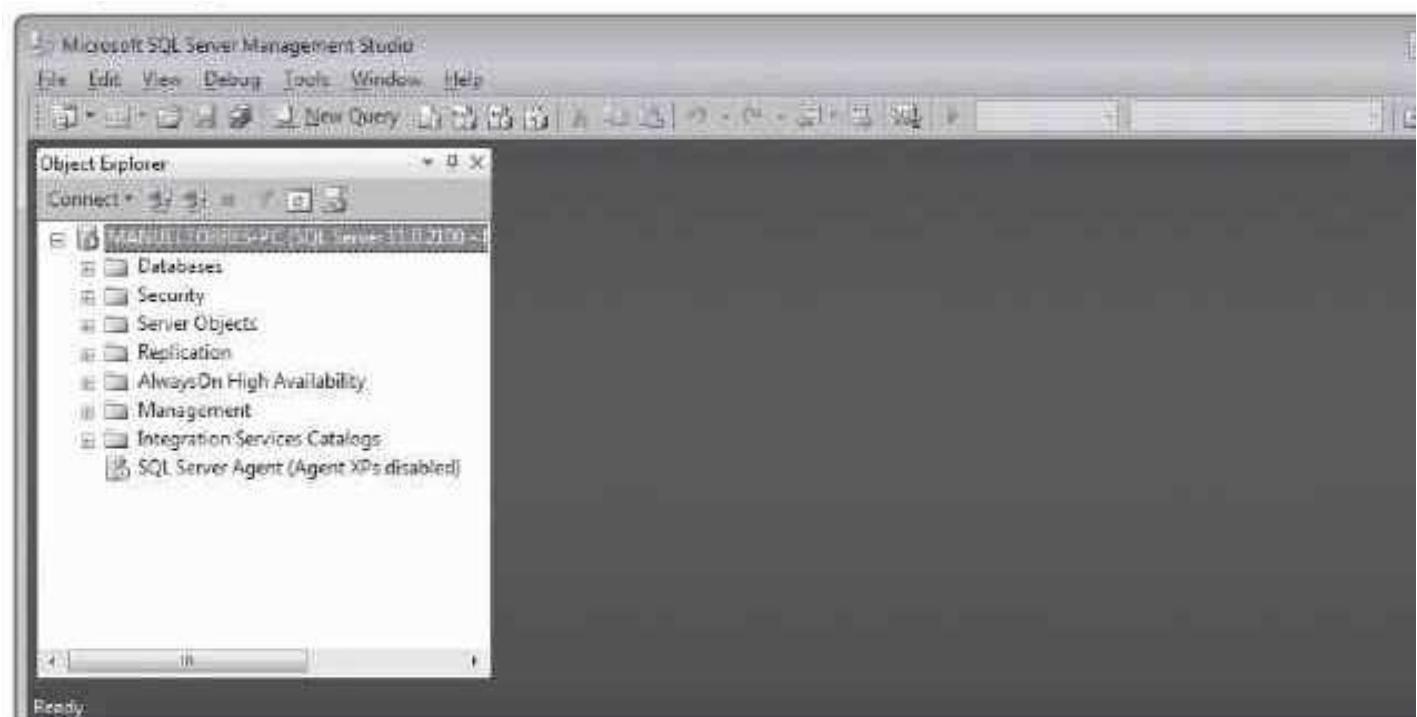
The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in History

32 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Seguidamente debemos seleccionar el nombre del servidor y el tipo de autenticación con la ventana **Database Engine Configuration**.



A continuación se muestra el entorno de SQL Server 2012.



Después de seleccionar **New Query** se mostrará de la siguiente manera:



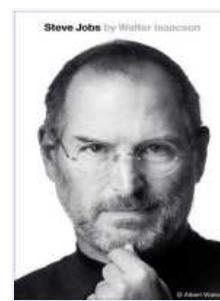
8 views

1 0

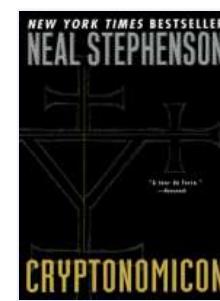
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 1: MICROSOFT SQL SERVER 2012

1. Barra de Herramientas: SQL Editor



Desde aquí se muestran las bases de datos disponibles, también se puede combinar las teclas CTRL+U.



Permite ejecutar un conjunto de instrucciones, también se puede pulsar F5.



Permite verificar si el conjunto de instrucciones es correcto, si es así, ejecuta.



El resultado de la ejecución de un conjunto de instrucciones se visualiza en tres entornos:

Results
tempdb
model
msdb
ReportServer
ReportServerTempDB
(6 row(s))

- **Texto(CTRL+T):** se muestran los resultados parecidos a la salida por consola.

name
1 master
2 tempdb
3 model
4 msdb
5 ReportServer
6 ReportServerTempDB

- **Grilla (CTRL+D):** presenta los resultados en forma de cuadrículas, esta forma de mostrarse es la más común.

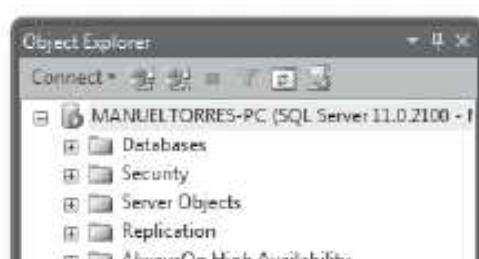
- **Reporte:** permite grabar en forma de reporte los resultados con extensión de este archivo es RPT.



Permiten colocar y eliminar asignación de comentarios sobre las sentencias seleccionadas por el usuario.

2. Panel Explorador de Objetos (F8)

Desde aquí se podrá administrar los objetos del servidor como bases de datos, seguridad, procedimientos almacenados, etc. Hay que tener en cuenta que usted puede conectarse a varios servidores y administrarlos al mismo tiempo.



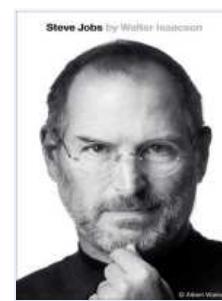
8 views

1 0

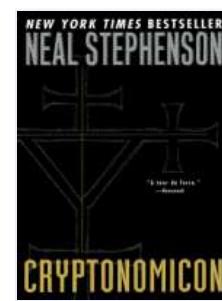
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

34

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

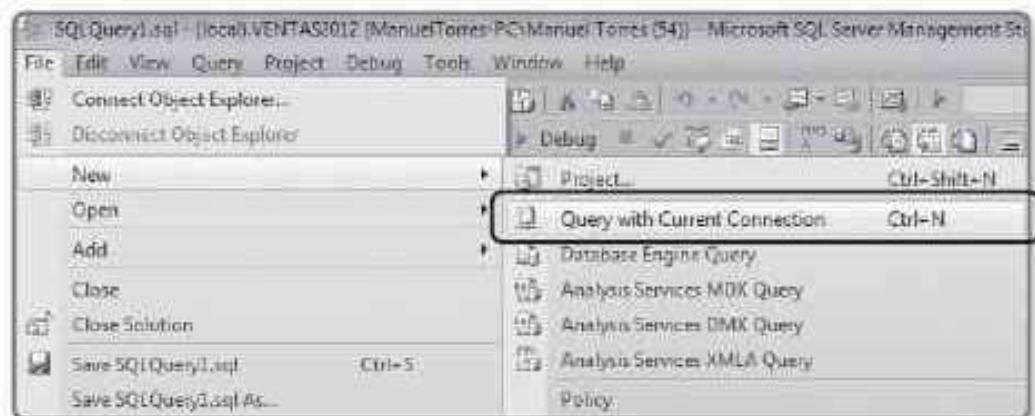
Veamos algunas opciones presentadas dentro del Explorador de Objetos:

- Si queremos conectarnos a un nuevo servidor debe presionar sobre el botón Connect Engine...
- Si queremos visualizar las bases de datos del sistema debemos seleccionar Databases Databases...
- Si tuvieramos una base de datos llamada Ventas2012 como podríamos sus procedimientos almacenados: seleccionar Databases > Ventas2012 > Programmability > Stored Procs

3. Entorno de desarrollo

SQL Server se caracteriza por implementar script dentro del editor de código ya que de esta manera podremos tener acceso a todos los objetos de una base. Todo esto gracias a los comandos que nos permite ejecutar desde versiones anteriores al SQL Server 2012 se viene utilizando la administración de ficheros eso quiere decir que se podrá implementar script desde diferentes hojas con consultas.

Para agregar una nueva hoja de edición debe seleccionar el botón **New Query** desde las herramientas o desde el menú File > New > Database Engine Query.



4. Panel de propiedades (F4)

Mientras desarrollemos script sobre las bases de datos no será necesario el manejo de propiedades, así es que para nuestro caso cerraremos este panel.

1.8. CONFIGURACIÓN DE FUENTE PARA EL ENTORNO DE TRABAJO

Transact-SQL se caracteriza por generar procesos en una base de datos por medio de los cuales se ejecutan dentro de un editor de consultas, en muchas ocasiones podemos configurar los operadores y símbolos en los script; por ejemplo, la letra o con el número cero (O - 0) ambos pueden ocasionar errores lógicos cuando ejecutemos algún script; por lo tanto, se recomienda

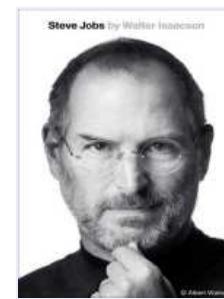
8 views

1 0

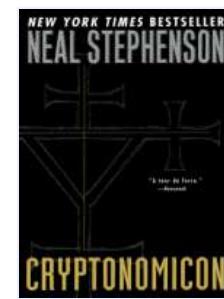
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs

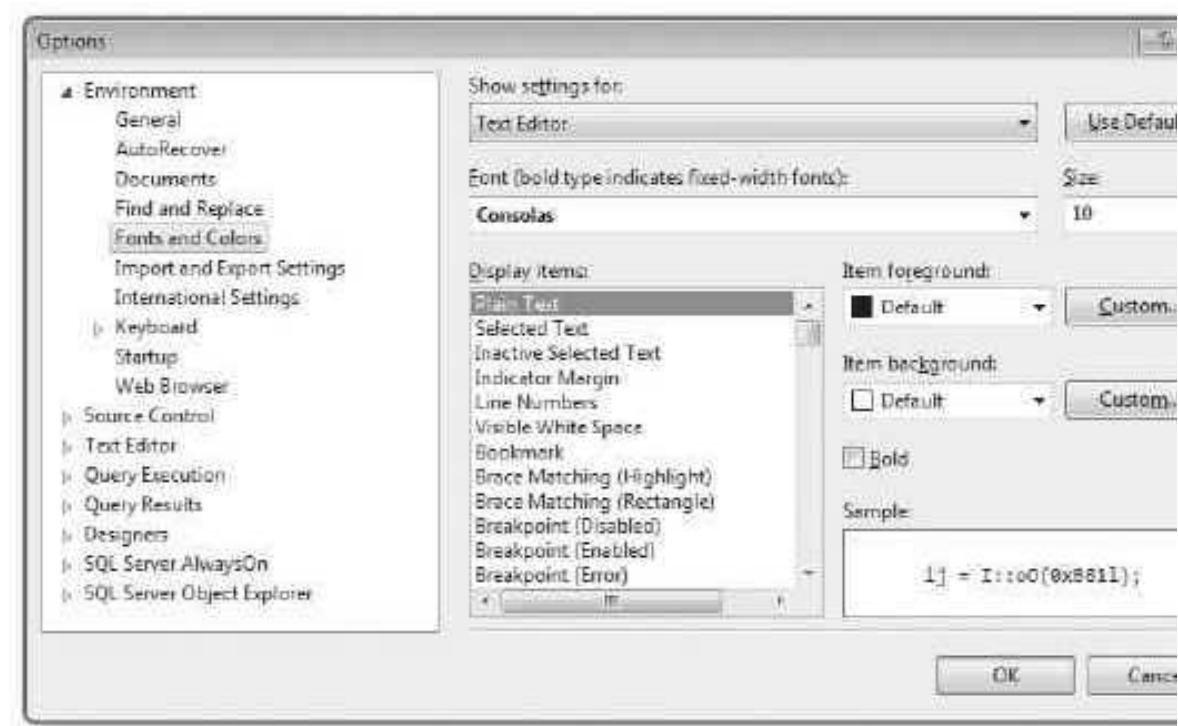


Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

● Presione OK



1.9. LENGUAJE DE DEFINICIÓN DE DATOS (LDD)

El lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregación que permiten manipular datos. Los elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

Existen dos (2) tipos de comandos SQL:

- Los comandos del Lenguaje de Definición de Datos (DDL) que permiten crear y definir nuevas estructuras de datos, campos e índices.
- Los comandos del Lenguaje de Manipulación de Datos (DML) que permiten modificar y generar datos existentes para insertar, modificar o eliminar, así como, ordenar, filtrar y extraer datos de la base de datos.

Para el desarrollo de los casos propuestos se propone el siguiente enunciado:

La ciudad de Lima está construyendo una línea metropolitana de transporte que une los distritos de la ciudad. Para lo cual cuenta con unas máquinas en cada estación que permiten cargar y recargar de dinero en unas tarjetas que permiten el acceso a estas líneas.

En cualquiera de los casos cuando el cliente introduce el dinero, el sistema debe, en primer lugar, validar el tipo de billete, validar que es correcta la definición del mismo y procesar lo seleccionado para este proceso acaba cuando la maquina emite una boleta y le entrega su cambio por dicho proceso.

1.10. SENTENCIA CREATE

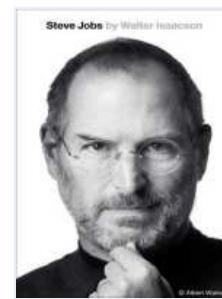
8 views

1 0

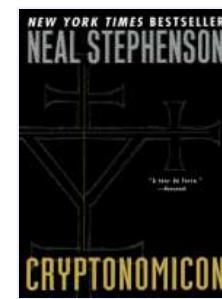
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

36 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 1.1

Implementar un script que permita crear la base de datos **METROPOLITANO** con valores e

CREANDO UNA BASE DE DATOS CON VALORES ESTÁNDAR

COMANDO LDD CREATE DATABASE

CREATE DATABASE METROPOLITANO

En el script se implementa la creación de la base de datos Metropolitano en el servidor local. Para ver los valores estándar se tiene que ejecutar el siguiente script:

SP_HELPDB METROPOLITANO

name		owner		status		compatibility	
METROPOLITANO	7.81 MB	ManuelTomes-PC\ManuelTomes	31	Aug 7 2012	Status=ONLINE Updateability=READ_WRITE UserAcc	100	
name	fileid	filename	filegroup	size	maxsize	growth	usage
METROPOLITANO	1	E:\Program Files\Microsoft SQL Server\MSSQL10.MSS	PRIMARY	2304 KB	Unlimited	1024 KB	data only
METROPOLITANO_log	2	E:\Program Files\Microsoft SQL Server\MSSQL10.MSS	NULL	576 KB	2147483648 KB	10%	log only

CASO DESARROLLADO N° 1.2

Implementar un script que permita crear la tabla **BOLETA** dentro de la base de datos **METR** con las siguientes características:

Boleta N° 000000011515452

Fecha de Emisión: 06/08/2012
Monto: 10.00

LIMA-PERÚ

CREANDO UNA TABLA DE LA BASE DE DATOS METROPOLITANO

COMANDO LDD CREATE TABLE

```
CREATE TABLE BOLETA (
    NUMEROBOLETA      CHAR(15) NOT NULL,
    FECHAEMISION     DATE NOT NULL,
    MONTO             MONEY NOT NULL
)
GO
```

En el script se implementa la creación de la tabla BOLETA que permite definir la columnas Número de Fecha de Emisión y el Monto asignado por el proceso. Como podrá ver el comando Create se comienza acuerdo al objeto especificado en el script. Considere que para ejecutar el script de creación de deberá activar la base de datos para esto deberá ejecutar el siguiente script:

```
USE METROPOLITANO
GO
```

8 views

1 like

0 comments

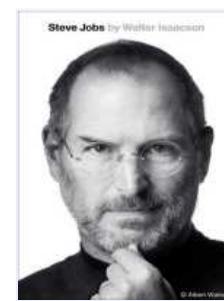
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

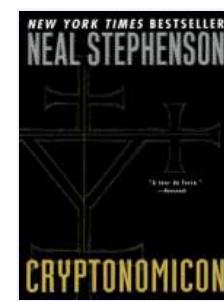
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 1: MICROSOFT SQL SERVER 20

CASO DESARROLLADO N° 1.3

Implementar un script que permita crear el **TRIGGER TX_MENSAJE** que muestre un mensaje cuando se realice una inserción o actualización a la tabla **BOLETA**.

CREANDO UNA TRIGGER A LA TABLA BOLETA

COMANDO LDD CREATE TRIGGER

```
CREATE TRIGGER TX_MENSAJE
ON BOLETA
FOR INSERT, UPDATE
AS
    PRINT 'LA BOLETA SE ACTUALIZO CORRECTAMENTE'
GO
```

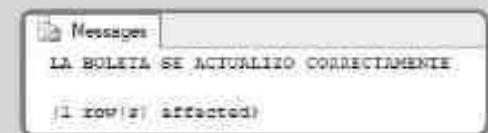
En el script se implementa la creación del desencadenador TX_MENSAJE que permite mostrar un mensaje cuando el usuario registra una Boleta nueva o cuando se actualiza algún campo de la tabla Boleta.

Para probar el Trigger debe agregar un Nuevo registro a la tabla Boleta con el siguiente script:

```
INSERT INTO BOLETA
VALUES('00000013', '07/08/2012', 20.50)
```

o una actualización a la tabla Boleta con el siguiente script:

```
UPDATE BOLETA
SET FECHAEMISION='07/08/2012',
MONTO=20
WHERE NUMEROBOLETA='00000013'
```



En ambos casos la imagen mostrada es la misma ya que al añadir o actualizar un registro el Trigger se activa.

CASO DESARROLLADO N° 1.4

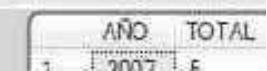
Implementar un script que permita crear el procedimiento almacenado **SP_TOTALBOLETA**.

CREANDO UN PROCEDIMIENTO ALMACENADO A LA TABLA BOLETA

COMANDO LDD CREATE PROCEDURE

```
CREATE PROCEDURE SP_TOTALBOLETAS
AS
BEGIN
    SELECT YEAR(FECHAEMISION) AS [AÑO],
           COUNT(*) AS [TOTAL]
    FROM BOLETA
    GROUP BY YEAR(FECHAEMISION)
END
GO
```

En el script se implementa la creación del procedimiento almacenado SP_TOTALBOLETAS que permite contabilizar el total de boletos registrados por año.



8 views

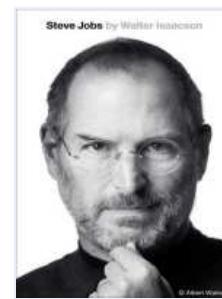
1 like

0 comments

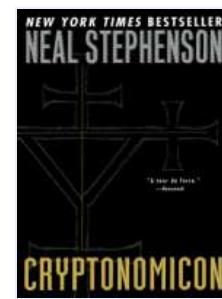
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

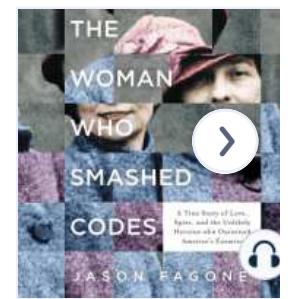
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

38 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 1.5

Implementar un script que permita crear la función **FN_TOTALBOLETAS** de un dígito ingresado por el usuario a la tabla **BOLETAS**.

CREANDO UNA FUNCIÓN ESCALAR A LA TABLA BOLETA

```
CREATE FUNCTION FN_TOTALBOLETAS(@AÑO INT)
RETURNS INT
AS
BEGIN
    DECLARE @TOTAL INT
    SELECT @TOTAL=COUNT(*)
    FROM BOLETA
    WHERE YEAR(FECHAEMISION)=@AÑO
    GROUP BY YEAR(FECHAEMISION)
    RETURN @TOTAL
END
```

COMANDO LDD CREATE FUNCTION

En el script se implementa la creación de la función FN_TOTALBOLETAS que de acuerdo a un año ingresado por el usuario deberá mostrar el total boletas registradas en dicho año. Para poder ejecutar la función tomar 2 formas mostradas a continuación:

Usando una consulta:

```
SELECT DBO.FN_TOTALBOLETAS(2007) AS [TOTAL DE BOLETAS]
```

TOTAL DE BOLETAS	
1	5

O usando el comando Print:

```
PRINT 'EL TOTAL DE BOLETAS ES: '+STR(DBO.FN_TOTALBOLETAS(2007))
```

EL TOTAL DE BOLETAS ES:	5
-------------------------	---

La diferencia entre ambos es sólo la forma en que muestran los resultados lo demás es idéntico como se visualizar en las imágenes de los comandos.

La función STR permite convertir a cadena un valor de otro tipo, en este caso la función devuelve un número y al tratar de imprimirlo con el comando Print emitirá un error si no lo convierte, a continuación muestra el error ocasionado sino aplican la función STR.

```
Msg 745, Level 16, State 1, Line 1
Conversion failed when converting the varchar value 'EL TOTAL DE BOLETAS ES: ' to data type int.
```

CASO DESARROLLADO N° 1.6

Implementar un script que permita crear la vista **VBOLETAS** donde muestre los registros en la tabla **BOLETAS**.

CREANDO UNA VISTA DE LA TABLA BOLETA

```
CREATE VIEW VBOLETA
AS
SELECT B.NUMEROBOLETA,B.FECHAEMISION,B.MONTO
FROM BOLETA B
```

COMANDO LDD CREATE VIEW

En el script se implementa la creación de la vista VBOLETA que permite listar los registros de la tabla Boleta. Para poder ejecutar la vista debe colocar el siguiente script:

8 views

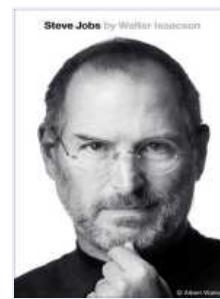
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

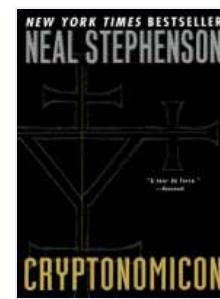
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

1.11. SENTENCIA ALTER

COMANDO DE MODIFICACIÓN DE OBJETOS DE UNA BASE DE DATOS

ALTER

```
ALTER OBJETO NOMBREOBJETO (
    --Datos modificados
)
```

La sentencia ALTER permite la modificación de un objeto asociado a una base de datos, puede modificar arc grupo de archivos, cambiar atributos de un objeto.

CASO DESARROLLADO N° 1.7

Implementar un script que permita agregar un archivo secundario a la base de datos **METR** llamado **METROPOLITANO_SEC2** de tamaño inicial 5MB y un tamaño máximo de 10MB co de crecimiento de 2%.

AGREGANDO UN ARCHIVO SECUNDARIO A LA BASE DE DATOS

COMANDO LDD ALTER DATABASE

```
ALTER DATABASE METROPOLITANO
ADD FILE (
    NAME='METROPOLITANO_SEC2',
    FILENAME='E:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\METROPOLITANO_SEC2.N
    SIZE=5MB,
    MAXSIZE=10MB,
    FILEGROWTH=2%
)
```

En el script se agrega un archivo secundario a la base de datos METROPOLITANO con las características solicitadas en el caso. Hay que comprobar los archivos que contaba dicha base de datos, con el siguiente comando:

```
SP_HELPDB METROPOLITANO
```

Note: en la imagen siguiente la base de datos estándar tiene solo 2 archivos:

name	db_size	owner	objid	created	status	compatibility_level	
METROPOLITANO	2.81 MB	ManuelTorres-PC\Manuel Torres	31	Aug 7 2012	Status=ONLINE, Updateability=READ_WRITE, UserAccess=ALLOWED	100	
name	fileid	filename	filegroup	size	maxsize	growth	usage
METROPOLITANO	1	E:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERV...	PRIMARY	2304 KB	Unlimited	1024 KB	data only
METROPOLITANO_log	2	E:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERV...	NULL	576 KB	2147483648 KB	10%	log only

Y al agregar un archivo secundario se vería de la siguiente forma:

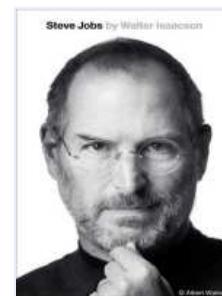
8 views

1 0

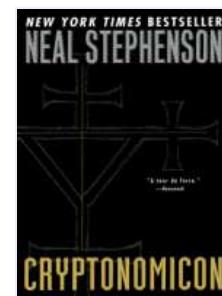
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

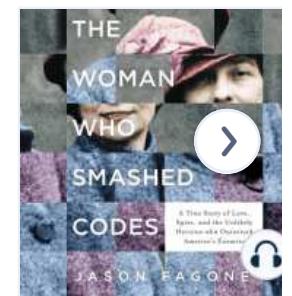
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

40

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 1.8

Implementar un script que permita modificar la precisión de la columna Número de Boleta inicialmente con el valor 15, modificarlo por 20 de la tabla Boleta:

MODIFICAR EL VALOR DE LA COLUMNA EN UNA TABLA

```
--1. Verificar las columnas de la tabla Boleta
SP_COLUMNS BOLETA
GO

--2. Modificar la columna Numero de Boleta
ALTER TABLE BOLETA
    ALTER COLUMN NUMEROBOLETA CHAR(20) NOT NULL
GO
```

COMANDO LDD ALTER TABLE

En el punto uno se verifica las precisiones de las columnas inicialmente definidas, la imagen siguiente muestra la precisión de la columna numeroBoleta es 15.

TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME	PRECISION	LENGTH
1	METROPOLITANO	dbo	BOLETA	1	char	15	15
2	METROPOLITANO	dbo	BOLETA	-9	date	10	20
3	METROPOLITANO	dbo	BOLETA	3	money	19	21

Luego de modificar la columna numeroBoleta con la instrucción ALTER TABLE usted podrá notar el cambio de precisión ejecutando nuevamente la instrucción SP_COLUMNS así lo muestra la siguiente imagen:

TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME	PRECISION	LENGTH
1	METROPOLITANO	dbo	BOLETA	1	char	20	20
2	METROPOLITANO	dbo	BOLETA	-9	date	10	20
3	METROPOLITANO	dbo	BOLETA	3	money	19	21

CASO DESARROLLADO N° 1.9

Implementar un script que permita modificar el TRIGGER TX_MENSAJE que adicione anterior a la fecha actual sólo cuando se inserta o actualiza la tabla BOLETA.

MODIFICAR UN TRIGGER

```
ALTER TRIGGER TX_MENSAJE
ON BOLETA
FOR INSERT, UPDATE
AS
    PRINT 'LA BOLETA SE ACTUALIZO CORRECTAMENTE'
    PRINT 'FECHA'+CAST(GETDATE() AS VARCHAR(10))
GO
```

COMANDO LDD ALTER TRIGGER

En el script se implementa la modificación del desencadenador TX_MENSAJE que permite mostrar un mensaje cuando el usuario registra una Boleta nueva o cuando se actualiza algún campo de la tabla Boleta además mostrar la fecha de dicha actualización.

La función CAST permite la conversión de cualquier tipo y cuenta con el siguiente formato:

CAST(VALOR AS TIPODATOS)

La diferencia entre STR y CAST es que el primero convierte directamente a cadena y por tanto imprime espacios en el lado izquierdo de la conversión. CAST convierte a cualquier tipo de datos y no incorpora espacios en blanco.

8 views

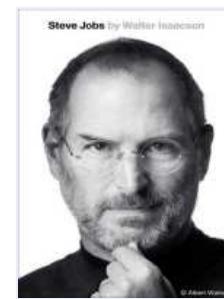
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

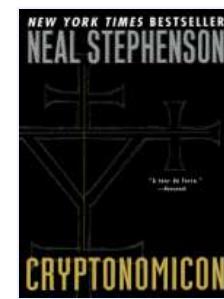
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

CASO DESARROLLADO N° 1.10

Implementar un script que permita modificar el procedimiento almacenado **SP_TOTALBOLETAS** tabla **BOLETA**, donde muestre la columna MES agrupando el número de boletas por cada mes y mostrar el total de boletas registradas.

MODIFICANDO UN PROCEDIMIENTO ALMACENADO

```
ALTER PROCEDURE SP_TOTALBOLETAS
AS
BEGIN
    SELECT YEAR(FECHAEMISION) AS [AÑO],
           MONTH(FECHAEMISION) AS [MES],
           COUNT(*) AS [TOTAL]
    FROM BOLETA
    GROUP BY ROLLUP(YEAR(FECHAEMISION),MONTH(FECHAEMISION))

END
GO
```

COMANDO LDD ALTER PROCEDURE

En el script se implementa la modificación del procedimiento almacenado donde se muestran el total de boletas según el año y un mes respectivo y al final de cada cambio se calcula el total de boletas por año que en la fila 6 se muestra 2007 NULL 5, esto significa que el total de boletas registradas en el año 2007 es 5 lo mismo se aplica al año 2008 en la fila 13 y en el año 2012 en la fila 15, tenga en cuenta que la fila 16 muestra el consolidado de boletas registradas que será el resultado del total de boletas de los años 2007 (5 boletas), 2008 (7 boletas) y 2012 (2 boletas).

	AÑO	MES	TOTAL
1	2007	1	1
2	2007	3	1
3	2007	5	1
4	2007	10	1
5	2007	11	1
6	2007	NULL	5
7	2008	1	1
8	2008	4	1
9	2008	8	1
10	2008	9	1
11	2008	10	1
12	2008	12	2
13	2008	NULL	7
14	2012	8	2
15	2012	NULL	2
16	NULL	NULL	14

Finalmente, observe la fila 14, la interpretación sería que en el mes de agosto del año 2012 se registró

8 views

1 like

0 comments

eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)

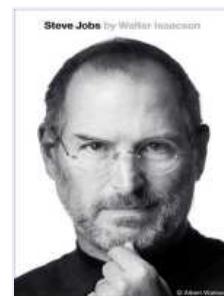
Save

Embed

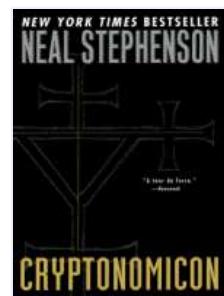
Share

Print

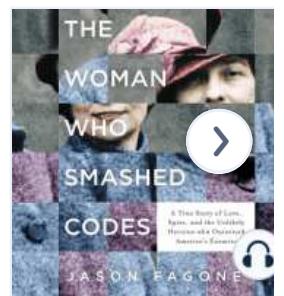
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

42

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 1.11

Implementar un script que permita modificar la función **FN_TOTALBOLETAS** en la cual añadir el parámetro mes y muestre el total de boletas según el año y mes ingresado **BOLETAS**.

MODIFICANDO UNA FUNCIÓN ESCALAR

COMANDO LDD ALTER FUNCTION

```
ALTER FUNCTION FN_TOTALBOLETAS(@AÑO INT,@MES INT)
RETURNS INT
AS
BEGIN
    DECLARE @TOTAL INT
    SELECT @TOTAL=COUNT(*)
    FROM BOLETA
    WHERE YEAR(FECHAEMISION)=@AÑO AND MONTH(FECHAEMISION)=@MES
    GROUP BY YEAR(FECHAEMISION)
    RETURN @TOTAL
END
```

En el script se implementa la modificación de la función FN_TOTALBOLETAS que de acuerdo a un año ingresado por el usuario debe mostrar el total de boletas registradas en dichos parámetros. Para ejecutar la función puede tomar 2 formas mostradas a continuación:

Usando una consulta:

```
SELECT dbo.FN_TOTALBOLETAS(2012,3) AS [TOTAL DE BOLETAS]
```

TOTAL DE BOLETAS	
1	4

O usando el comando Print:

```
DECLARE @AÑO INT=2012,@MES INT=3
PRINT 'EL TOTAL DE BOLETAS DEL MES '+CAST(@MES AS CHAR(2))+'
DEL AÑO '+CAST(@AÑO AS CHAR(4))+'
ES: '+ CAST(dbo.FN_TOTALBOLETAS(@AÑO,@MES) AS CHAR(4))
```

```
EL TOTAL DE BOLETAS DEL MES 3 DEL AÑO 2012 ES: 4
```

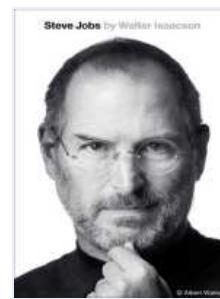
8 views

1 0

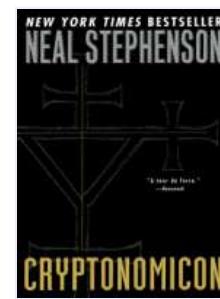
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

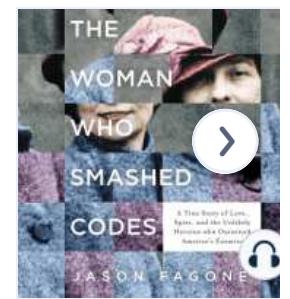
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CASO DESARROLLADO N° 1.12

Implementar un script que permita modificar la vista **VBOLETAS** donde muestre sólo la Número de Boleta y Fecha de Emisión de la tabla **BOLETAS**, además de encriptar la implementación.

MODIFICANDO UNA VISTA

COMANDO LDD ALTER VIEW

```
ALTER VIEW VBOLETA
WITH ENCRYPTION
AS
    SELECT B.NUMEROBOLETA,B.FECHAEMISION
        FROM BOLETA B
```

En el script se implementa la modificación de la vista VBOLETA que solo muestra las columnas Número de Boleta y Fecha de Emisión además de encriptarlo:

```
SELECT * FROM VBOLETA
```

Para visualizar la implementación de la vista VBOLETA se debe colocar el siguiente script:

```
SP_HELPTEXT VBOLETA
```

La activación de la encriptación dentro de la vista no permite visualizar el script que implementa dicha vista, esta es una forma de proteger el script mostrando el siguiente mensaje:

The text for object 'VBOLETA' is encrypted.

```
SP_HELPTEXT VBOLETA
```

Si en la implementación de la vista no se coloca la cláusula WITH ENCRYPTION entonces al visualizar el script se mostrará:

Text
1 CREATE VIEW VBOLETA
2 AS:
3 SELECT B.NUMEROBOLETA,B.FECHAEMISION
4 FROM BOLETA B

1.12. SENTENCIA DROP

COMANDO DE ELIMINACIÓN DE OBJETOS DE UNA BASE DE DATOS

DROP OBJETO NOMBREOBJETO

La sentencia DROP permite la eliminación de un objeto asociado a una base de datos.

DROP

8 views

1 like

0 comments

eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)

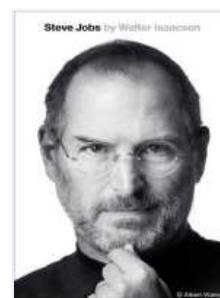
Save

Embed

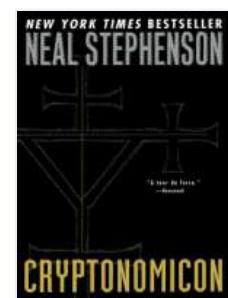
Share

Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

44

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 1.13

Implementar un script que permita eliminar la base de datos **METROPOLITANO**.

ELIMINANDO UNA BASE DE DATOS

COMANDO LDD DROP DATABASE

```
USE MASTER
GO
```

```
DROP DATABASE METROPOLITANO
GO
```

En el script se implementa la eliminación de la base de datos METROPOLITANO, para este caso la base de datos no debe estar en uso ya que ocurrirá un error y mostrará el siguiente mensaje:

```
Msg 3702, Level 16, State 3, Line 1
Cannot drop database "METROPOLITANO" because it is currently in use.
```

Para que la base de datos METROPOLITANO no sea la base activa se tiene que activar la base de datos maestra con el siguiente script USE MASTER.

CASO DESARROLLADO N° 1.14

Implementar un script que permita eliminar la tabla **BOLETA** de la base de datos **METROPOLITANO**.

ELIMINANDO UNA TABLA

COMANDO LDD DROP TABLE

```
DROP TABLE BOLETA
GO
```

En el script se implementa la eliminación de la tabla BOLETA asociada a la base de datos METROPOLITANO. Tenga en cuenta que al eliminar una tabla también se eliminan los datos contenidos en la tabla, incluyendo triggers, constraints y permisos especificados en la tabla.

CASO DESARROLLADO N° 1.15

Implementar un script que permita eliminar el trigger **TX_MENSAJE** asociado a la tabla BOLETA de la base de datos **METROPOLITANO**.

ELIMINANDO UN TRIGGER

COMANDO LDD DROP TRIGGER

```
DROP TRIGGER TX_MENSAJE
GO
```

En el script se implementa la eliminación del trigger TX_MENSAJE, la eliminación de un trigger obedece cuando ya no necesite estar asociado a una tabla, porque SQL permite habilitar e inhabilitar un trigger con la sentencia Disable.

8 views

1 0

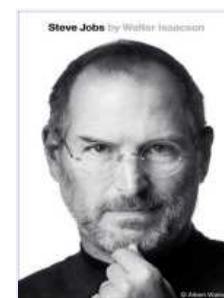
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

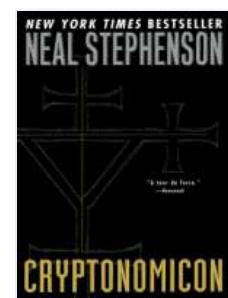
[Full description](#)

 Save  Embed  Share  Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A Tru



Gestión de base de datos

CAPACIDAD:

El lector será capaz de comprender los conceptos básicos de base de datos e implementar script para la creación de la misma.

CONTENIDO:

- *Qué es una base de datos*
- *Objetivos de los sistemas de base de datos*
- *Las bases de datos en SQL Server*
- *Estructura de una base de datos*
- *Archivos y grupos físicos de la base de datos*
- *Motor de base de datos*
- *Crear una base de datos*
- *Enunciado: Reserva de vuelos*
- *Separar y adjuntar una base de datos*
- *Procedimiento almacenado sp_detach_db*
- *Manejo de Esquemas*

8 views

1 0

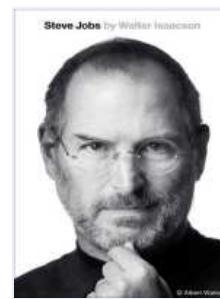
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

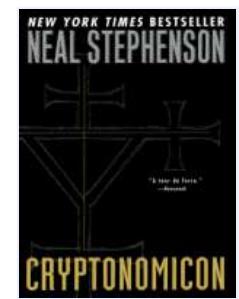
[Full description](#)

   
Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who
Smashed
Codes: A Tru

8 views

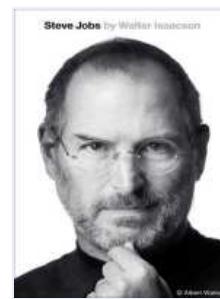
1 like

0 comments

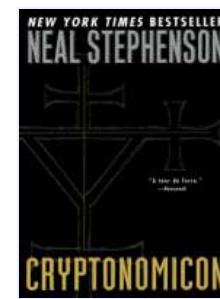
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
 [Save](#)
 [Embed](#)
 [Share](#)
 [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 2: GESTIÓN DE BASE DE DATOS

2.1. QUÉ ES UNA BASE DE DATOS

Es un conjunto de información relacionada sobre un tema particular, organizada de tal que suministra una base o fundamento para procedimientos, como la recuperación de la información, la elaboración de conclusiones en base a la data y la toma de decisiones de la organización. Se dice que cualquier conjunto de información que sirva a lo nombrado anteriormente se le calificará como base de datos, aun cuando la información no esté almacenada en el ordenador informático.

Desde aquí podemos desprender un concepto más básico sobre las bases de datos ya que los seres humanos desde los inicios del hombre siempre trató de almacenar información ya sea plasmándola en piedras, maderas o imprimiéndolas en papel. Si consideramos estos ejemplos llegamos a la conclusión de que el hombre siempre quiso manifestar su idea registrándola para siempre. Lo mismo nos sucede en la actualidad puesto que si desea registrar algún evento o fenómeno descubierto tendrá la necesidad de registrarla quizás para no olvidarla, determinar estadísticas o simplemente almacenarlas para su conveniencia.

La necesidad de una base de datos surge en los años 60, como respuesta a la anarquía que existía entre las organizaciones al tener cada vez mayor cantidad de archivos inclusive con información que no se relacionaba entre ellos, la propuesta de una base de datos como concepto tecnológico es:

- Registrar los datos que son importantes para una organización y deberán decidir el nivel de importancia de los mismos.
- Organizarlos de manera correcta.
- Colocarlos en un repositorio único, es decir, asignarle un nombre como Base de Datos.
- Impedir que las aplicaciones externas accedan a los datos directamente, sólo podrán hacerlo por medio de una conexión hacia dicha base, para esto SQL como gestor de base de datos establecerá las reglas de juego para dicho acceso.

Las aplicaciones que permite la administración de los datos almacenados en una o varias bases de datos se denomina **Sistema de Gestión de Bases de Datos (SGBD)**.

2.2. OBJETIVOS DE LOS SISTEMAS DE BASE DE DATOS

Los objetivos fundamentales de los SGBD son los siguientes:

1. Independencia de los datos y los programas de aplicación

Algunas aplicaciones de desarrollo de software implementaron su propio manejo de datos, el cual podría ser una solución adecuada siempre y cuando este evolucione de acuerdo tanto en el lenguaje de programación como el control de la data pero a veces eso no siempre es así.

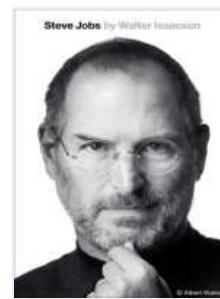
8 views

1 0

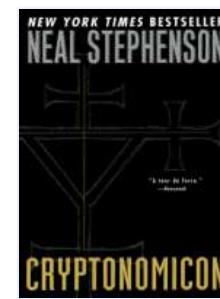
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

48

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

2. Minimización de la redundancia

La minimización responde a la necesidad de tener un control de los datos no redundantes para reducir los tiempos de acceso hacia los datos, ya que la eliminación completa de la redundancia sólo existe el control de dichos datos en un grado óptimo de control.

3. Integración y sincronización de las bases de datos

Con la integración se garantiza que los datos administrados en el mundo real sean representados de manera verídica en el mundo lógico de los datos. La sincronización está vinculada directamente a los lenguajes de programación en la cual una aplicación puede administrar la data desde diferentes puntos generando el uso simultáneo de la data por diferentes usuarios.

La integración de datos responde a la corrección del significado y la consistencia de los datos en el mundo real del cual proceden y que las aplicaciones sólo tienen por misión el resultado final, mas no valida los datos mostrados.

4. Seguridad y recuperación

La seguridad es un tema bastante delicado en la actualidad puesto que todo desarrollador de una aplicación y administrador de una base de datos busca siempre tener a salvo la información de una organización. Un SGBD garantiza el acceso autorizado a todos los usuarios configurados en un determinado servidor y así poder reducir el grado de vulnerabilidad de la data.

La recuperación encierra puntos como aplicar reingeniería a la base de datos y disponer de métodos para dicha actividad, mostrando un reporte de las fallas que se podría ocurrir.

5. Facilidad de manipulación de la información

Los usuarios de una base de datos pueden acceder a ella con solicitudes para resolver problemas diferentes. El SGBD debe contar con la capacidad de una búsqueda rápida por criterios, debe permitir que los usuarios planteen sus demandas de una forma simple sin perderse en las complejidades del tratamiento de los archivos y del direccionamiento de los datos. Los lenguajes actuales brindan lenguajes de alto nivel, con diferentes grados de facilidad para el programador, que garantizan este objetivo, los llamados sublenguajes de datos.

6. Control centralizado

Uno de los objetivos más importantes de los SGBD es garantizar el control centralizado de la información. Permite comprobar, de manera sistemática y única, los datos que se almacenan en la base de datos, así como el acceso a ella.

8 views

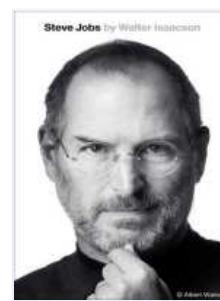
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

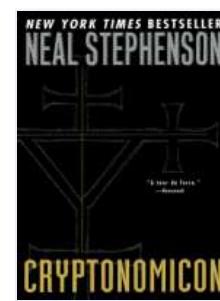
Uploaded by [anon_61286589](#)[Full description](#)

 Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



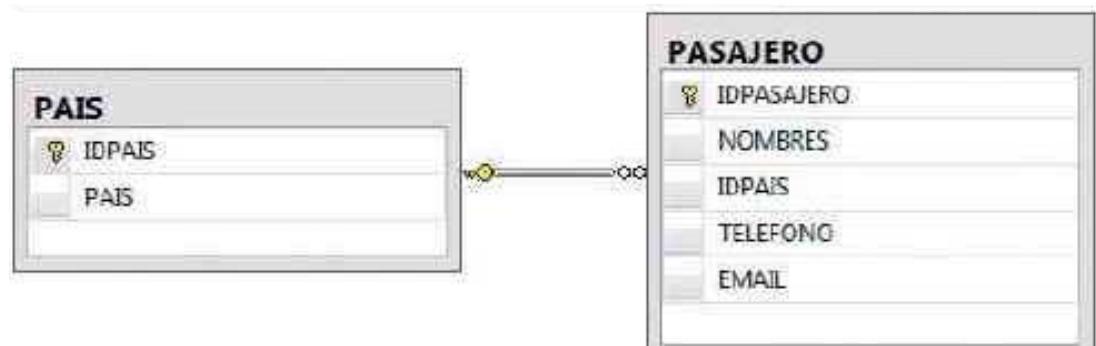
The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of Mathematics

CAP. 2: GESTIÓN DE BASE DE DATOS

MASTER	Master es el núcleo de toda instancia en SQL Server, eso quiere decir que esta base contiene información vital sobre una instancia de SQL Server. Se compone de tablas de sistema que registran el seguimiento de la instalación del servidor y de la administración de todas las bases de datos que se generen posteriormente. La información que contiene es bases de datos, opciones de configuración, triggers a nivel de servidor y servidores con enlace.
TEMPDB	Es una base de datos temporal, técnicamente es informal para SQL Server y su uso es generalmente cuando se realizan operaciones de clasificación y agregación. En ocasiones, las aplicaciones usan las tablas, procesos y cursor de esta tabla ya que provee de un espacio de trabajo temporal y tiene la capacidad de regenerarse cada vez que arranca SQL Server.
MODEL	Model es una plantilla para todas las bases de datos creadas en el servidor, esto lo podrá implementar cuando ejecute el comando CREATE DATABASE, dando lugar a la creación de una nueva base de datos basada en un modelo predefinido por SQL Server; este le añade páginas vacías que serán propias de la nueva base de datos.
MSDB	Es empleada por el servicio SQL Server Agent y por los servicios de integración de SQL Server en la cual guarda información con respecto a tareas de automatización, historial de copias de seguridad, información de rastreo de registros, tareas, alertas, cuentas proxy, planes de mantenimiento y registros de mensajería de la base de datos.

2.4. ESTRUCTURA DE UNA BASE DE DATOS

Una Base de Datos está compuesta por un conjunto de tablas o archivos.



En este caso vemos un modelo entidad relación entre dos tablas: la tabla PAIS contiene registros de los países de donde provienen los pasajeros y la tabla PASAJERO contiene el conjunto de los pasajeros grabados por algún proceso dentro de la agencia de viaje.

Veamos los registros de la tabla PAIS:

IDPAIS	PAIS
1	PERU
2	ARGENTINA
3	CHILE

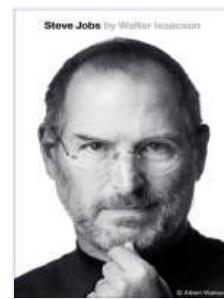
8 views

1 0

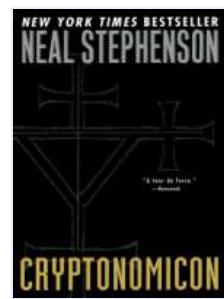
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

50

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Y ahora veremos los registros de la tabla PASAJERO:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4 P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5 P0005	LULU LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM
6 P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7 P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8 P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
9 P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
10 P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM

La base de datos Agencia presenta dos entidades:

- Registros sobre los países llamada entidad PAIS, almacenado dentro de un registro en el disco duro de la computadora llamada PAIS.
- Registros sobre los pasajeros llamada entidad PASAJERO, almacenado dentro de un registro en el disco duro de la computadora llamada PASAJERO.

Lo que podemos entender es que una tabla llega a ser una entidad y; por lo tanto, toda cualquier cosa física o abstracta que se puede representar, en este caso ambas tablas pero podríamos implementar la tabla PAGO que representa a los abstractos ya que un pago físicamente pero si se puede representar como una entidad ya que un pago tiene una monto.

Entonces podemos llegar a la siguiente conclusión:

BASE DE DATOS : AGENCIA
TABLAS O ENTIDAD : PASAJERO PAIS



8 views

1 like

0 comments

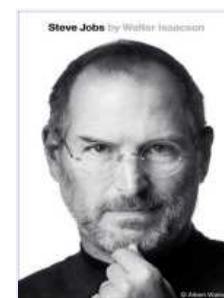
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

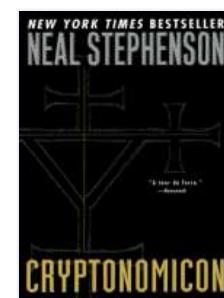
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 2: GESTIÓN DE BASE DE DATOS

CAMPO FORÁNEO: es el campo que une a otra entidad formando una extensión de la tabla. Una entidad puede tener muchos campos claves, si consideramos que toda entidad tiene una clave, el foráneo se enlazará justamente con este campo.

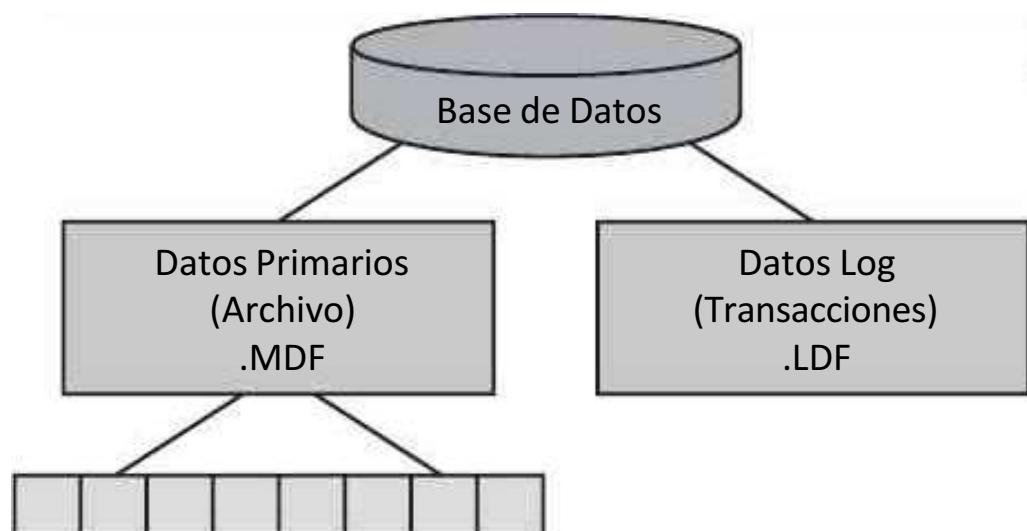
REGISTRO: representa el conjunto de valores por cada campo de una misma fila. Esta representación es el resultado de los consultas que SQL Server espera devolver a los usuarios.

DATO: es un valor único que no representa nada mientras no se une a otros datos. Por ejemplo, el número 18 podría representar la edad, el número de boletas registradas o el número de faltas de un estudiante.

2.5. ARCHIVOS Y GRUPOS FÍSICOS DE LA BASE DE DATOS

Principalmente SQL Server divide su trabajo en un archivo para datos y otro para las transacciones.

ARCHIVOS DE DATOS PRIMARIOS (MDF)	Toda base de datos tiene un archivo de datos primario que realiza el seguimiento de todos los demás archivos, además de almacenar datos. Por convenio este archivo tiene la extensión MDF.
ARCHIVO DE REGISTRO (LOG)	Todas las bases de datos por lo menos tendrán un archivo de registro que contiene la información necesaria para recuperar todas las transacciones que suceden sobre la misma. Por convenio la extensión de este archivo es LDF.



2.6. MOTOR DE BASE DE DATOS

Es el servicio principal que SQL Server proporciona al usuario, este tendrá por misión recuperar, procesar y proteger los datos. Este servicio proporciona acceso controlado y privado de transacciones que debe cumplir SQL Server como servidor a las aplicaciones consumidoras.

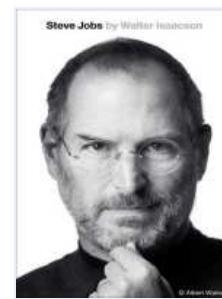
8 views

1 0

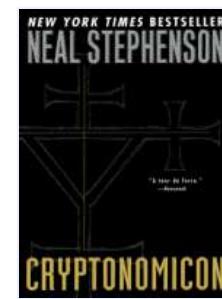
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

52

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

- Implementar sistemas para obtener acceso y cambiar los datos almacenados en la base de datos que incluye implementar los sitios Web o las aplicaciones que funcionan con los datos, crear procedimientos que utilicen las herramientas y utilidades de SQL Server para trabajar con los datos.
- Aplicar los sistemas implementados en la organización o en los clientes.
- Proporcionar soporte técnico administrativo diario para optimizar el rendimiento de los datos.

2.7. CREAR UNA BASE DE DATOS

Hay que considerar que la creación de una base de datos es meramente implementación y no lo representaremos de forma visual, es decir, usaremos en todo momento Transact-SQL para la implementación.

Una base de datos en SQL Server es una réplica de la base de datos Model, ya que esta proporciona páginas vacías es que podemos crear archivos lógicos dentro de la base de datos. Es la implementación profesional de la base de datos.

Sintaxis:

```
CREATE DATABASE NombreBaseDatos
[  
    ON PRIMARY (
        NAME      = 'NombreArchivoLogico',
        FILENAME = 'NombreArchivoFisico',
        SIZE     = TamañoEnDisco,
        MAXSIZE  = MaximoTamaño,
        FILEGROWTH = FactorDeCrecimiento
    )]  
  
    [ LOG ON (
        NAME      = 'NombreArchivoLogico',
        FILENAME = 'NombreArchivoFisico',
        SIZE     = TamañoEnDisco,
        MAXSIZE  = MaximoTamaño,
        FILEGROWTH = FactorDeCrecimiento
    )]
```

Donde:

- **NOMBREBASEDATOS:** es el nombre de la nueva base de datos, deben ser únicos entre sí y pueden tener hasta 128 caracteres, a menos que no se especifique ningún nombre para el registro. Si no se especifica ningún nombre SQL genera un nombre lógico al anexar

8 views

1 like

0 comments

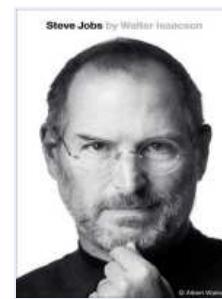
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

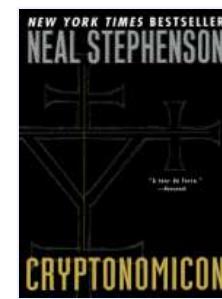
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 2: GESTIÓN DE BASE DE DATOS

- **NAME:** especifica el nombre lógico del archivo. Este archivo es el utilizado para referenciar en las sentencias del Transact-SQL que se ejecutan después.
- **FILENAME:** especifica el nombre de archivo en el sistema operativo. Se debe especificar el acceso y nombre del archivo que el sistema operativo reconocerá cuando se utiliza la base de datos. La ruta de acceso debe especificar un directorio en el servidor.
- **SIZE:** especifica el tamaño para el archivo. Cuando este parámetro no es especificado, el archivo de Registro SQL le asigna automáticamente 1MB. El mínimo predeterminado es 1MB y el máximo es 2TB.
- **MAXSIZE:** especifica el tamaño máximo de crecimiento del archivo, se pueden utilizar los valores de 1MB a 2TB, el valor predeterminado es 1MB, sólo se puede especificar números enteros.
- **FILEGROWTH:** especifica el incremento de crecimiento del archivo, este valor no puede ser menor que el maxsize. Emplee un número entero. Un valor 0 indica que no hay crecimiento. El crecimiento se puede especificar en MB, KB o %. El valor predeterminado es de 10%.

2.8. ENUNCIADO: RESERVA DE VUELOS

Para todos los casos desarrollados de este material usaremos el siguiente enunciado. En cada capítulo se tocará temas de sentencias, funciones o cualquier tipo de instrucción de SQL que serán resueltos basados a este caso. Tome interés en las entidades, capacidades del mismo y las relaciones entre ellos.

El sistema de reserva de vuelos es un sistema que permite al usuario hacer consultas y reservas de vuelos, además de poder comprar los boletos de viajes de forma remota, sin la necesidad de acudir a una agencia de viajes. Se desea implementar un sistema de reservas que sea accesible vía web.

El sistema actualmente cuenta con un terminal de servicio de reserva en donde se presenta una interfaz de bienvenida al sistema, describiendo los servicios ofrecidos junto con la opción para registrarse por primera vez que accede a ella o caso contrario usar el sistema de reserva de vuelos. Esta información se da por medio de un usuario (email) y una clave previamente registrada en el sistema.

Una vez registrado el usuario este podrá seleccionar los siguientes procesos:

- Consulta de vuelos
- Reserva de vuelos
- Compra de boletos aéreos

La consulta de vuelos se puede realizar de las siguientes formas:

- Horarios de vuelos
- Tarifas de vuelos

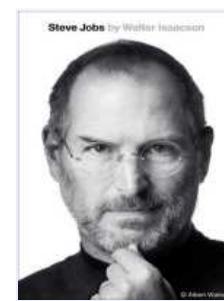
8 views

1 0

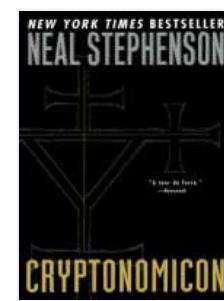
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

54

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

La reserva de vuelo permite al usuario hacer una reserva para un vuelo particular, especificando fecha y horario, bajo una tarifa establecida. Es posible reservar un itinerario compuesto de varios vuelos, para uno o más pasajeros, además de poder reservar asientos.

La compra permite al usuario, dada una reserva de vuelo y un número de tarjeta de crédito, adquirir los boletos aéreos.

Los boletos aéreos serán posteriormente enviados al usuario o estarán listos para ser adquiridos en el terminal del aeropuerto antes de la salida de su vuelo.

Es necesario estar previamente registrado con un número de tarjeta de crédito válida para las compras de boletos aéreos, o bien proveerla en el momento de la compra.

Además de los servicios de vuelo, el usuario podrá en cualquier momento leer, modificar o eliminar su propio registro, todo esto después de haber sido validado como usuario del sistema.

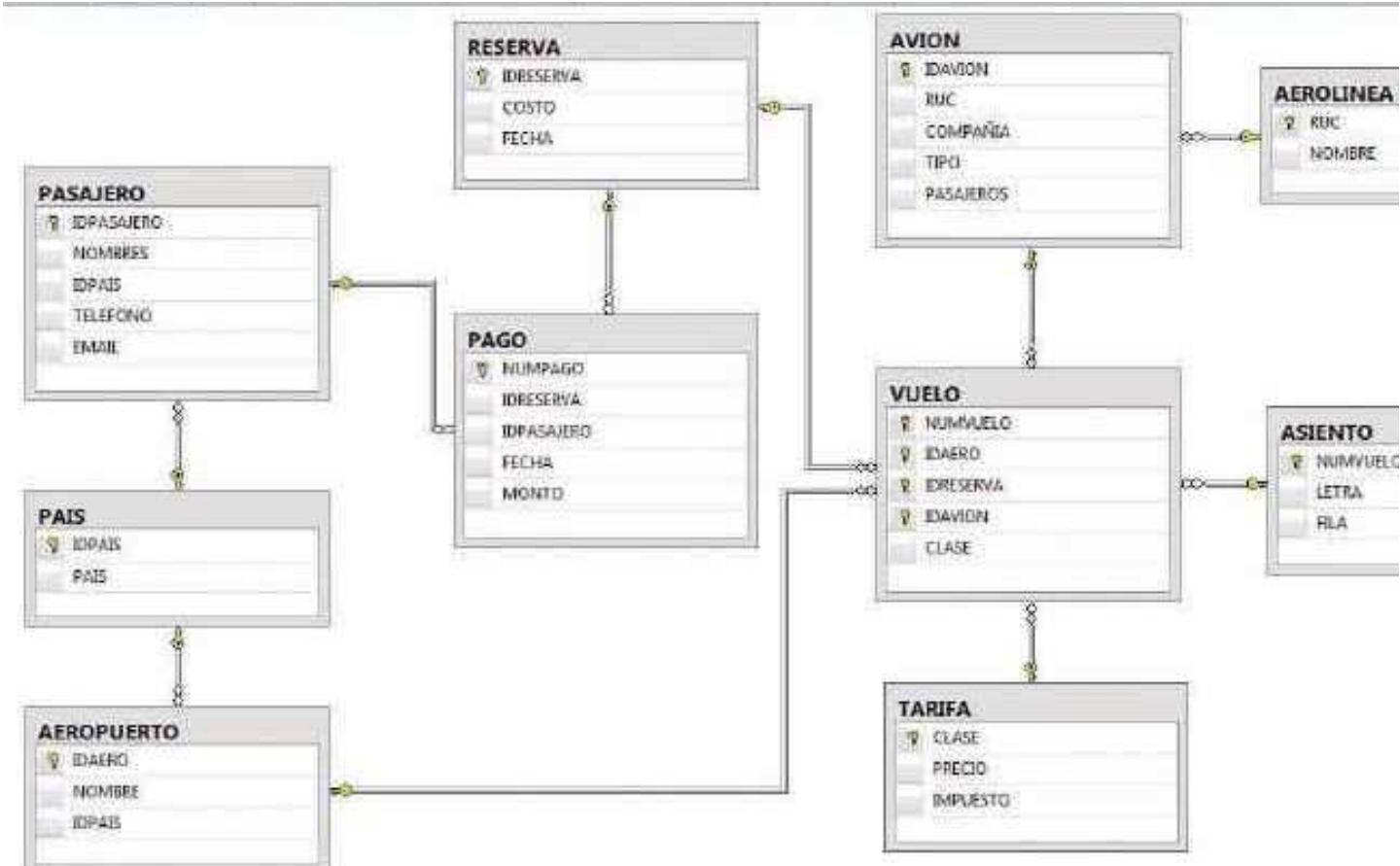


Fig. 2.1

En la Fig. 2.1 se muestra el Modelo Entidad Relación del caso propuesto, este será la base de todos los casos propuestos en este libro.

A continuación mostraremos las capacidades de cada columna:

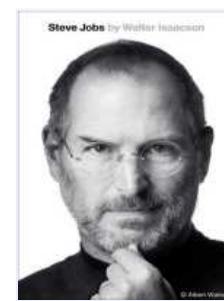
8 views

1 0

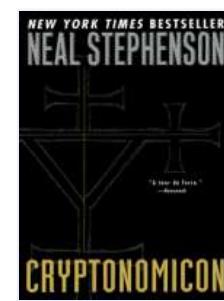
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



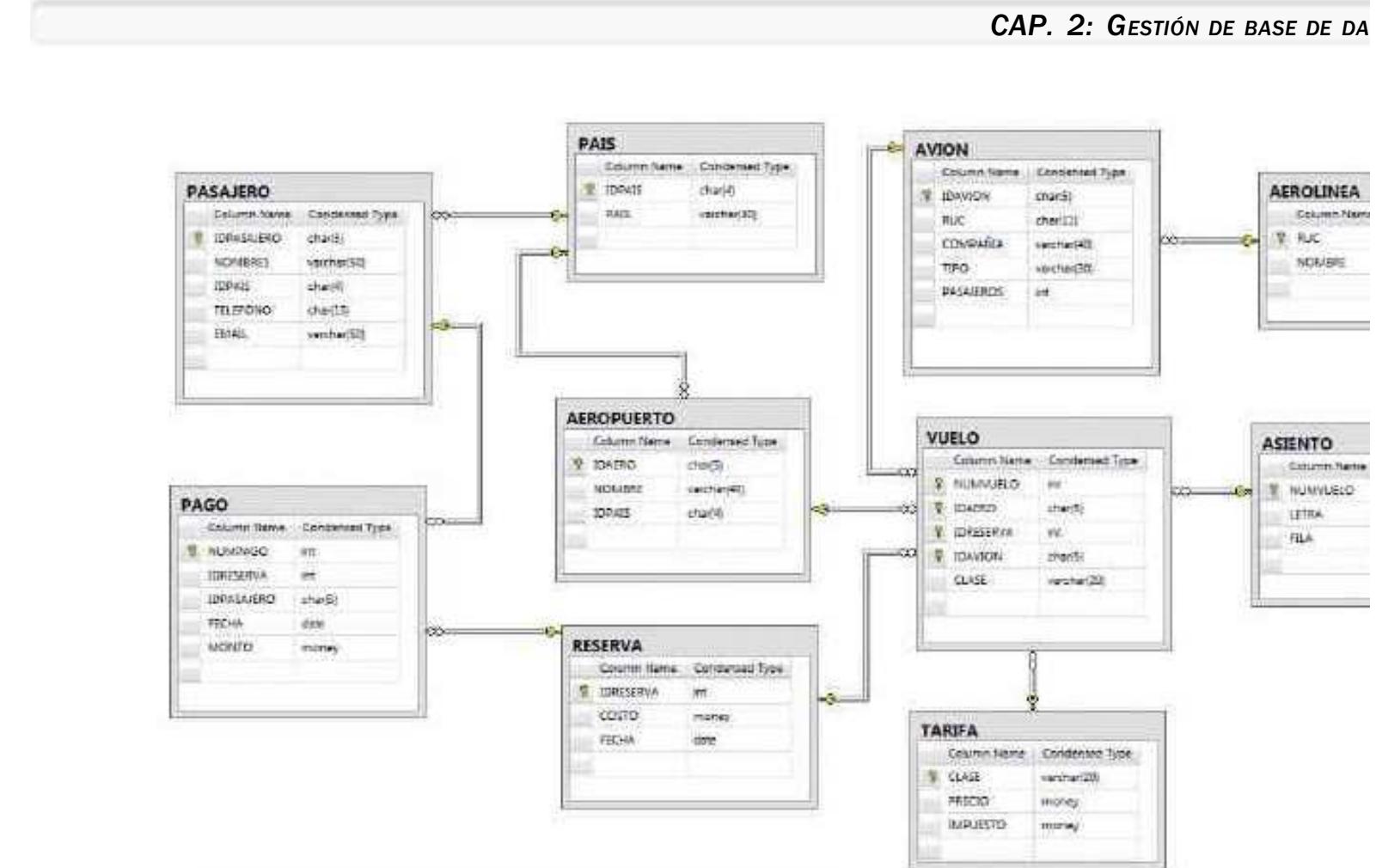
Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America



Para obtener un Diagrama con especificaciones de columnas y tipos de datos debe seguir los siguientes pasos:

- Debe ejecutar el script de la base de datos Agencia que se encuentra en el CD del libro.
- Desde el explorador de objetos de Sql Server 2012 debe seleccionar Databases > Agencia > Diagrams y presionar clic derecho para seleccionar New Database Diagram.



- Desde la ventana anterior debe seleccionar las tablas que necesite para generar su diagrama.

8 views

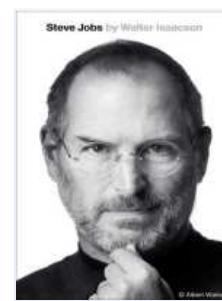
1 like

0 comments

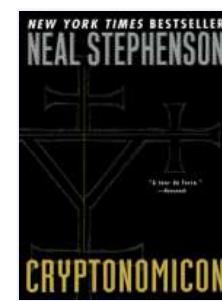
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

56

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 2.1

Implementar la base de datos **AGENCIACD1** con valores por defecto.

```
--1.
CREATE DATABASE AGENCIACD1
GO

--2.
IF DB_ID('AGENCIACD1') IS NOT NULL
BEGIN
    USE MASTER
    DROP DATABASE AGENCIACD1
END

CREATE DATABASE AGENCIACD1
GO
```

En el punto uno se muestra la primera forma de crear una base de datos en forma estándar aquí es que podría ya existir una base de datos con el mismo nombre en el servidor por lo tanto es necesario revisar las bases creadas o eliminarlas antes de crear una nueva.

En el punto dos se implementa el script que permitirá verificar si la base de datos existe, así se activa la base de datos MASTER y se aplica DROP DATABASE para eliminar la base de datos existente. Seguidamente se procederá a crear la base de datos.

La creación de base de datos con valores por defecto implica crearla con valores que SQL Server establece de manera predeterminada eso quiere decir que la asignación de nombre lógico, físico, tamaño y su forma de crecimiento es controlado en forma estándar.

Antes de mostrar la capacidad asignada a la base de datos primero necesita verificar si la base de datos **AGENCIACD1** existe, para esto digite el siguiente script:

```
SELECT NAME,DBID,CRDATE,FILENAME
FROM SYS.SYSDATABASES
GO
```

El script mostrado permite listar las bases de datos creadas en el servidor activo, también es posible listar de la siguiente forma **SELECT * FROM SYS.SYSDATABASES** pero muestra algunas columnas irrelevantes; por tal motivo en el script se muestra la columna Nombre de la base de datos (**NAME**), el identificador de base (**dbID**), la fecha de creación (**crDate**) y la ubicación dentro del sistema operativo (**fileName**).

La imagen siguiente muestra el resultado de ejecutar el script:

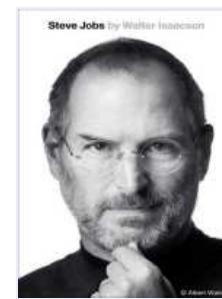
8 views

1 0

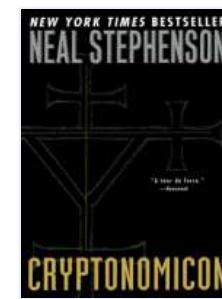
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 2: GESTIÓN DE BASE DE DATOS

El siguiente script mostrará las capacidades asignadas a la base de datos creada en forma:

```
SP_HELPDB AGENCIA
GO
```

La imagen siguiente muestra el resultado de ejecutar el script:

	name	db_size	owner	dbid	created	status	collation	
1	AgenciaCD1	2.81 MB	ManuelTorres-PC\Manuel Torres	27	Aug 6 2012	Status=ONLINE	Updateability=READ_WRITE	
	name	fileid	filename	filegroup	size	maxsize	growth	usage
1	AgenciaCD1	1	E:\Program Files\Microsoft SQL Server\MSSQL10_MSSQL\DATA\AgenciaCD1.mdf	PRIMARY	2304 KB	Unlimited	1024 KB	data only
2	AgenciaCD1_log	2	E:\Program Files\Microsoft SQL Server\MSSQL10_MSSQL\DATA\AgenciaCD1_log.LDF	NULL	576 KB	2147483648 KB	10%	log only

Como notará el procedimiento almacenado SP_HELPDB separa el archivo lógico del físico en dos resultados al mismo tiempo. Entonces, ahora podremos determinar los siguientes:

- Nombre asignado al archivo primario (AgenciaCD1)
- Nombre asignado al archivo de transacciones (AgenciaCD1_log)
- Tamaño en disco (2.81 MB) que representa el total acumulado entre el archivo primario y el archivo de transacciones (576 KB).
- Ubicación de los archivos en el sistema operativo E:\Program Files\Microsoft SQL Server\MSSQLSERVER\MSSQL\DATA\AgenciaCD1.mdf y E:\Program Files\Microsoft SQL Server\MSSQLSERVER\MSSQL\DATA\AgenciaCD1_log.LDF.
- El máximo tamaño del archivo primario es UNLIMITED, es decir, el tamaño está sujeto a ofrecido por el sistema operativo y su forma de crecimiento es de 1024KB.
- Mientras que el archivo de transacciones cuenta con un máximo tamaño de 2GB con crecimiento del 10% del tamaño inicial.

CASO DESARROLLADO N° 2.2

Implementar la Base de Datos AGENCIACD2 ubicado en la carpeta E:\SISTEMA_AGENCIA\ es un archivo primario con tamaño inicial de 15MB, un tamaño máximo de 30MB y un crecimiento de 5MB, además el archivo de transacciones debe tener un tamaño inicial de 5MB y como máximo una tasa de crecimiento de 10%.

```
CREATE DATABASE AGENCIACD2
ON PRIMARY(
    NAME='AGENCIACD2_PRI',
    FILENAME='E:\SISTEMA_AGENCIA\AGENCIACD2.MDF',
    SIZE=15MB,
    MAXSIZE=30MB,
```

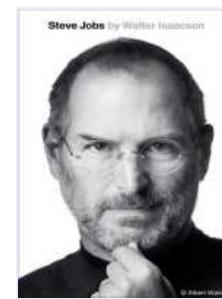
8 views

1 0

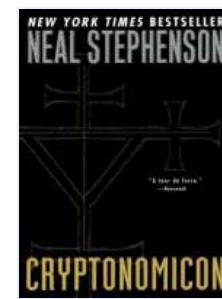
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

58

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el script se implementa el archivo primario y el de transacciones con los tamaños especiales, hay que tener en cuenta que el Name se definen nombres distintos puesto que el tienen que diferenciar, aquí se asignó AGENCIACD2_PRI por ser el archivo primario y AGENCIACD2_LOG al archivo de transacciones; pero hay que recalcar que no necesariamente deberán formar cada usuario propone su propio nombre de archivo lógico.

La imagen a continuación muestra los archivos desde el explorador de Windows como nota operativo asigna al archivo primario el tipo **SQL Server Database Primary Data File** y de transacciones el tipo **SQL Server Database Transaction Log File** eso quiere decir que el sistema operativo ya reconoce estos tipos de archivo.

Nombre	Fecha de modificación	Tipo	Tamaño
AGENCIACD2.LDF	05/08/2012 12:59 p.m.	SQL Server Database Transaction Log File	5,120 KB
AGENCIACD2.MDF	05/08/2012 12:59 p.m.	SQL Server Database Primary Data File	15,360 KB

Ahora, verificaremos los valores asignados a los archivos implementados: empezaremos por el archivo AGENCIACD2_PRI.MDF:

```
USE AGENCIACD2
GO
SP_HELPFILE AGENCIACD2_PRI
```

name	filename	filegroup	size	maxsize	growth	usage
AGENCIACD2_PRI	E:\SISTEMA_AGENCIA\AGENCIACD2.MDF	PRIMARY	15360 KB	30720 KB	5120 KB	data only

Como la verificación se realiza por archivo, primero debemos activar la base de datos con USE.

Finalmente, verificaremos los valores implementados al archivo AGENCIACD2_LOG con el siguiente script:

```
USE AGENCIACD2
GO
SP_HELPFILE AGENCIACD2_LOG
```

name	filename	filegroup	size	maxsize	growth	usage
AGENCIACD2_LOG	E:\SISTEMA_AGENCIA\AGENCIACD2.LDF	NULL	5120 KB	10240 KB	10%	log only

CASO DESARROLLADO N° 2.3

Implementar la base de datos AGENCIACD3 en la carpeta E:\SISTEMA_AGENCIA\ con los siguientes parámetros:

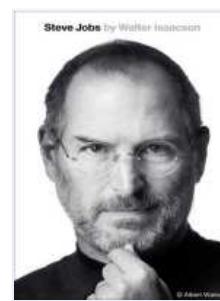
8 views

1 0

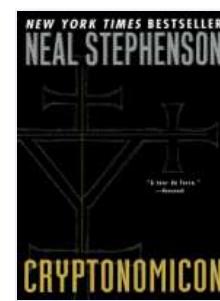
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

```

CREATE DATABASE AGENCIACD3
ON PRIMARY(
    NAME='AGENCIACD3_PRI',
    FILENAME='E:\SISTEMA_AGENCIA\AGENCIACD3.MDF',
    SIZE=10MB,
    MAXSIZE=40MB,
    FILEGROWTH=5MB
),
(
    NAME='AGENCIACD3_SEC',
    FILENAME='E:\SISTEMA_AGENCIA\AGENCIACD3.NDF',
    SIZE=5MB,
    MAXSIZE=30MB,
    FILEGROWTH=5%
)
LOG ON(
    NAME='AGENCIACD3_LOG',
    FILENAME='E:\SISTEMA_AGENCIA\AGENCIACD3.LDF',
    SIZE=4MB,
    MAXSIZE=20MB,
    FILEGROWTH=10%
)
GO

```

En el script se implementa el archivo primario, secundario y el de transacciones con los valores especificados en el caso, nótese que los nombres lógicos varían según el tipo de archivo a implementar pero hasta aquí no es necesario diferenciar los archivos físicos ya que el sistema opera tres tipos de extensiones como MDF para el primario, NDF para el secundario y el LDF para las transacciones.

La siguiente imagen muestra los 3 archivos implementados, el que nos faltaba especificar es el secundario, el cual es reconocido como SQL Server Database Secondary Data File.

AGENCIACD3.LDF	06/08/2012 02:07 p.m.	SQL Server Database Transaction Log File	4.096 KB
AGENCIACD3.MDF	06/08/2012 02:07 p.m.	SQL Server Database Primary Data File	10.240 KB
AGENCIACD3.NDF	06/08/2012 02:07 p.m.	SQL Server Database Secondary Data File	5.120 KB

Ahora verificaremos los valores asignados al archivo AGENCIACD3_SEC.NDF:

```

USE AGENCIACD3
GO
SP_HELPFILE AGENCIACD3_SEC

```

name	filename	filegroup	size	maxsize	growth	usage
AGENCIACD3_SEC	E:\SISTEMA_AGENCIA\AGENCIACD3.NDF	PRIMARY	5120	30720	5120	0

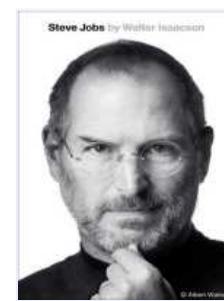
8 views

1 0

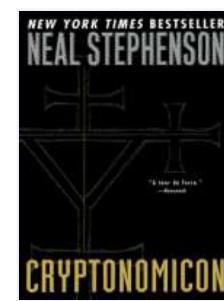
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

60

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 2.4

Implementar la base de datos AGENCIACD4 en la carpeta E:\SISTEMA_AGENCIA\ con configuración:

- Archivo de datos: un tamaño inicial de 10MB, máximo de 40MB y un factor de crecimiento de 5%.
- Archivo secundario 1: un tamaño inicial de 5MB, máximo de 30MB y un factor de crecimiento de 5%.
- Archivo secundario 2: un tamaño inicial de 2MB, máximo de 15MB y un factor de crecimiento de 2%.
- Archivo de transacciones: un tamaño inicial de 4MB, máximo de 20MB y un factor de crecimiento de 10%.

```

CREATE DATABASE AGENCIACD4
ON PRIMARY(
    NAME='AGENCIACD4_PRI',
    FILENAME='E:\SISTEMA_AGENCIA\AGENCIACD4.MDF',
    SIZE=10MB,
    MAXSIZE=40MB,
    FILEGROWTH=5MB
),
(
    NAME='AGENCIACD4_SEC1',
    FILENAME='E:\SISTEMA_AGENCIA\AGENCIACD41.NDF',
    SIZE=5MB,
    MAXSIZE=30MB,
    FILEGROWTH=5%
),
(
    NAME='AGENCIACD4_SEC2',
    FILENAME='E:\SISTEMA_AGENCIA\AGENCIACD42.NDF',
    SIZE=2MB,
    MAXSIZE=15MB,
    FILEGROWTH=2%
)
LOG ON(
    NAME='AGENCIACD4_LOG',
    FILENAME='E:\SISTEMA_AGENCIA\AGENCIACD4.LDF',
    SIZE=4MB,
    MAXSIZE=20MB,
    FILEGROWTH=10%
)

```

En el script se implementa un archivo primario, dos secundarios y el de transacciones. Los tamaños especificados en el caso, hay que tener en cuenta que por tener dos secundarios, es necesario modificar el nombre lógico (NAME) y en el disco de nombre del archivo (FILENAME) ya que el script lo genera para un solo secundario.

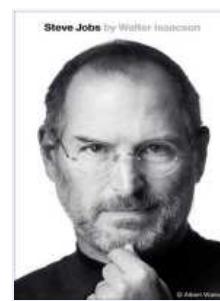
8 views

1 0

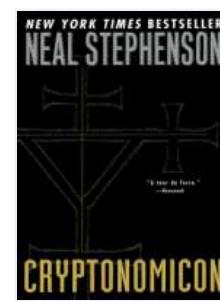
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 2: GESTIÓN DE BASE DE DATOS

Ahora verificaremos los valores asignados a los archivos que componen la base de datos AGENCIACD4:

```
USE AGENCIACD4
GO
SP_HELPDB AGENCIACD4
```

	name	db_size	owner	dbid	created	status		
1	AGENCIACD4	21.00 MB	ManuelTorres-PC\Manuel Torres	30	Aug 5 2012	Status=ONLINE Updateability=READ_WRITE		
	name	fileid	filename	filegroup	size	maxsize	growth	used
1	AGENCIACD4_PRI	1	E\SISTEMA_AGENCIA\AGENCIACD4.MDF	PRIMARY	10240 KB	40960 KB	5%	data
2	AGENCIACD4_LOG	2	E\SISTEMA_AGENCIA\AGENCIACD4.LDF	NULL	4096 KB	20480 KB	10%	log
3	AGENCIACD4_SEC1	3	E\SISTEMA_AGENCIA\AGENCIACD41.NDF	PRIMARY	5120 KB	30720 KB	5%	data
4	AGENCIACD4_SEC2	4	E\SISTEMA_AGENCIA\AGENCIACD42.NDF	PRIMARY	2048 KB	15360 KB	2%	data

2.9. SEPARAR Y ADJUNTAR UNA BASE DE DATOS

Cuando se crea una base de datos en el servidor este lo protege de gestiones desde el sistema operativo. Esto quiere decir que no lo podrá eliminar desde el explorador de Windows o cambiar su nombre. Para esto el motor de base de datos propone políticas de gestión sobre la base de datos propias de su funcionamiento, funciones, sentencias e instrucciones para dicho control.

Pongamos el caso que necesite mover los archivos de base de datos desde su lugar de origen a otra unidad con mayor espacio. Debemos seguir los siguientes pasos:

- Cerrar todos los archivos asociados a la base de datos.
- Desactivar los bloqueos del sistema operativo.
- Eliminar la entrada que tiene sobre la base de datos desde la instancia instalada.
- Mover los archivos físicos mediante el explorador de Windows.
- Crear una nueva entrada de base de datos dentro de la instancia.
- Activar la base de datos.

Cuando se separa una base de datos, el servidor sigue los siguientes pasos:

- El servidor envía una solicitud de bloqueo CommitExclusive a la base de datos.
- El servidor espera hasta que todas las transacciones en curso se confirmen o se reviertan.
- El servidor genera todos los metadatos que necesita para separar la base de datos.
- La base de datos es marcada como eliminada.
- El servidor confirma la transacción.

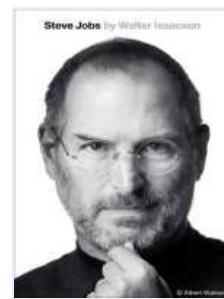
8 views

1 0

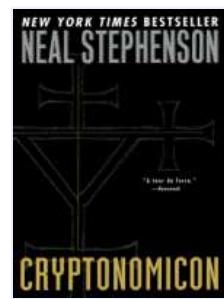
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

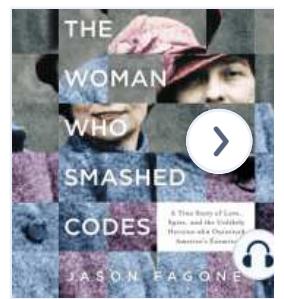
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

62

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012**2.10. PROCEDIMIENTO ALMACENADO SP_DETACH_DB**

Permite separar una base de datos del servidor actual.

Sintaxis:

SEPARAR LA BASE DE DATOS ACTIVA**SP_DETACH_DB**

```
SP_DETACH_DB 'NOMBRE_BASE_DATOS'
GO

SP_DETACH_DB @DBNAME='NOMBRE_BASE_DATOS'
GO
```

CASO DESARROLLADO N° 2.5

Script que permite separar la base de datos AGENCIACD5 creada con las siguientes características:

- Archivo de datos: un tamaño inicial de 10MB, máximo de 90MB y un factor de crecimiento de 10%.
- Archivo secundario: un tamaño inicial de 5MB, máximo de 50MB y un factor de crecimiento de 5%.
- Archivo de transacciones: un tamaño inicial de 5MB, máximo de 50MB y un factor de crecimiento de 10%.

```
CREATE DATABASE AGENCIACD5
ON PRIMARY(
    NAME='AGENCIACD5_PRI',
    FILENAME='E:\SISTEMA_AGENCIA\AGENCIACD5.MDF',
    SIZE=10MB,
    MAXSIZE=90MB,
    FILEGROWTH=5MB
),
(
    NAME='AGENCIACD5_SEC',
    FILENAME='E:\SISTEMA_AGENCIA\AGENCIACD5.NDF',
    SIZE=5MB,
    MAXSIZE=50MB,
    FILEGROWTH=5%
)
LOG ON(
    NAME='AGENCIACD5_LOG',
    FILENAME='E:\SISTEMA_AGENCIA\AGENCIACD5.LDF',
    SIZE=5MB,
    MAXSIZE=50MB,
    FILEGROWTH=10%
)
GO
```

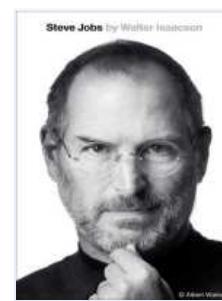
8 views

1 0

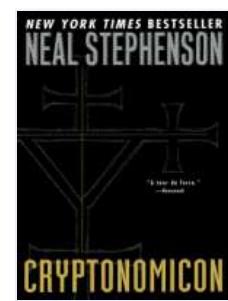
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

Al intentar eliminarlo como cualquier archivo, el mensaje mostrado por el sistema operativo es el siguiente:



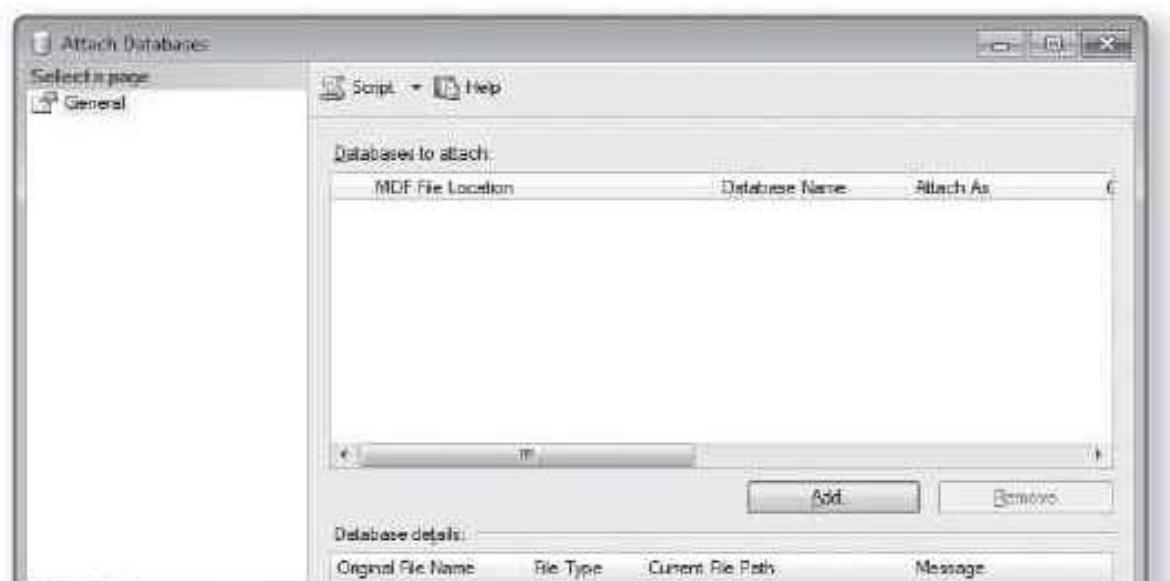
Así realicemos las actividades que propone la ventana de Archivo en uso no se logrará eliminarlo mientras este se encuentre unido a los demás archivos, en el script siguiente se procede a liberar los archivos.

```
EXEC SP_DETACH_DB 'AGENCIACD5'
GO
```

Una vez ejecutado el script usted podrá: cambiar el nombre de los archivos MDF, LDF y N y podrá mover de una carpeta a otra o eliminarlas por separado. Hay que notar que separar los datos no se podrá visualizar en el Explorador de Objetos del Management de SQL Server.

Ahora adjuntaremos en una sola instancia todos los archivos de la base de datos separados paso anterior, esta vez lo realizaremos por medio de un asistente de adjuntar base de datos en los siguientes pasos:

- Desde el explorador de objetos, presiones clic derecho sobre el nodo de Base de Datos ' (Attach) Adjuntar.



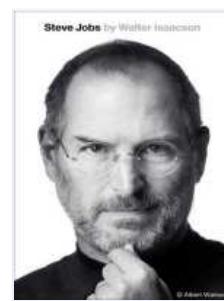
8 views

1 0

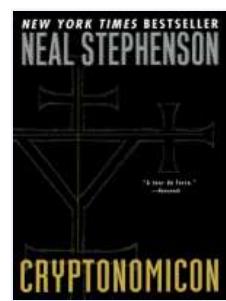
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

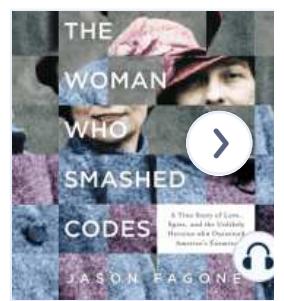
RELATED TITLES



Steve Jobs



Cryptonomicon

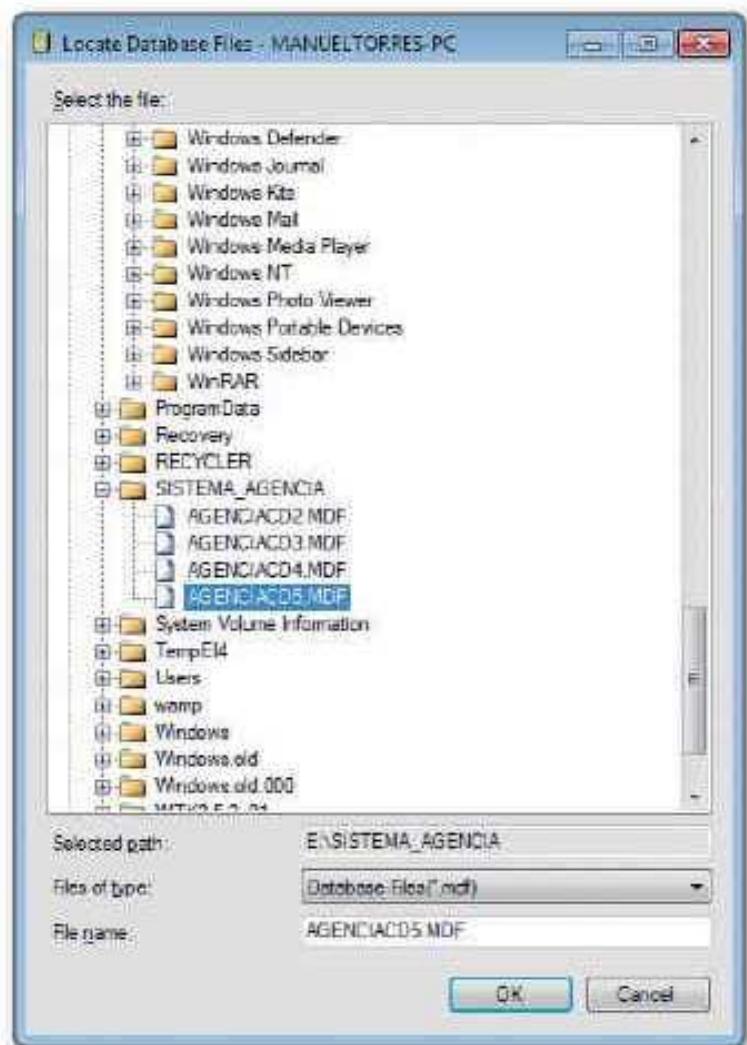


The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

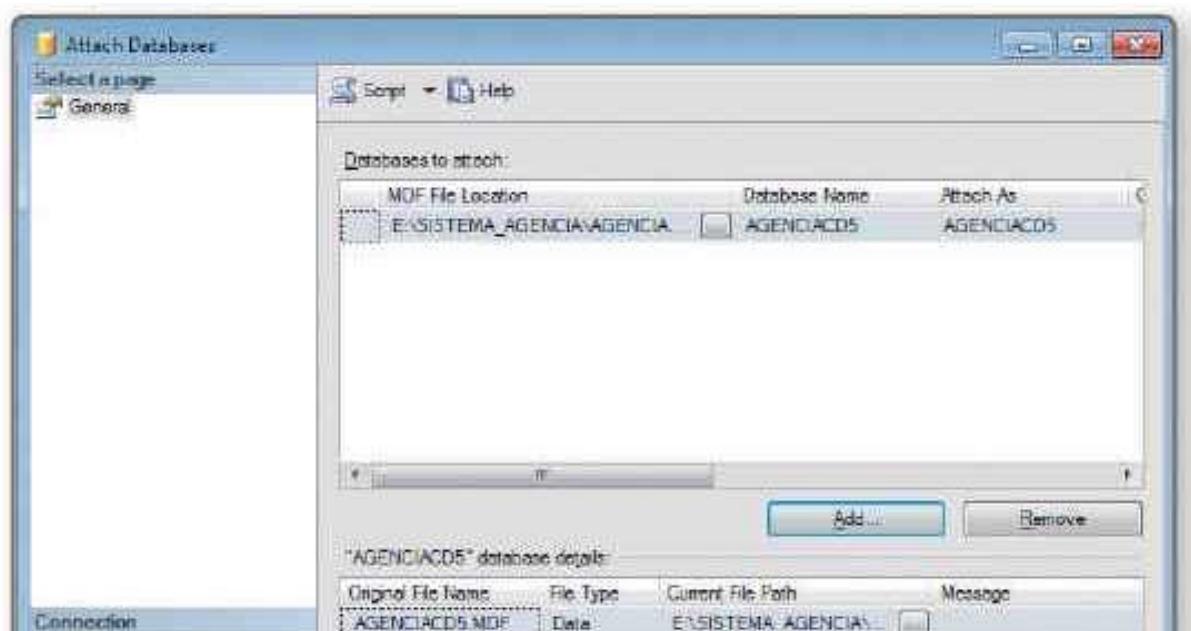
64

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Desde esta ventana deberá seleccionar los archivos que componen una base de datos, I presione el botón Add... (Aregar) y seleccione el archivo primario de la base de datos como se muestra en la siguiente imagen:



- Presione OK y observe que los archivos que componían la base de datos AGENCIACD5 por completo, para iniciar el proceso de adjuntar presione OK.



8 views

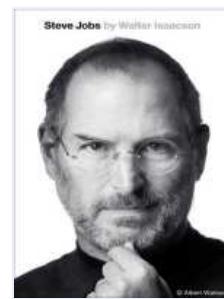
1 like

0 comments

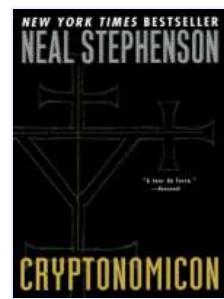
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



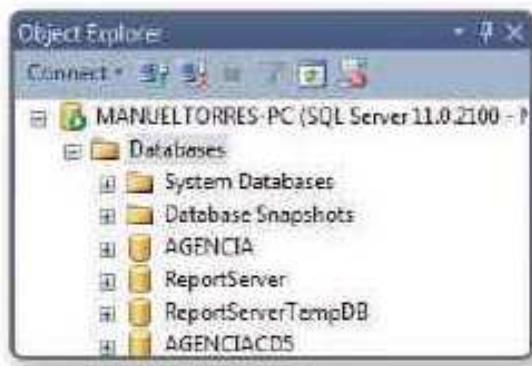
Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

CAP. 2: GESTIÓN DE BASE DE DATOS

- Finalmente refresque el nodo de base de datos desde el explorador de objetos presionando el botón derecho sobre el nodo Databases > seleccionando Actualizar (REFRESH) y observará que el nodo AGENCIACD5 aparece nuevamente.



2.11. MANEJO DE ESQUEMAS

El esquema de una base de datos (en inglés, Database Schema) describe la estructura de los datos que todos los objetos que se puedan crear o administrar en SQL se encuentran dentro de un estándar.

El manejo de esquemas responde a temas como agrupación de objetos y seguridad de los datos. Cuando se crea un objeto de base de datos o se especifica una entidad de seguridad de dominio (usuario o grupo) como la propietaria del objeto, o la entidad de seguridad de dominio se asocia con la base de datos como esquema. Esa entidad de seguridad de dominio será la propietaria del esquema.

Sintaxis de creación:

CREANDO UN ESQUEMA CREATE SCHEMA

```
CREATE SCHEMA NOMBRE_ESQUEMA AUTHORIZATION NOMBRE_PROPIETARIO
```

Sintaxis de eliminación de esquemas:

Para eliminar un esquema debe tener en cuenta que no tenga objetos asociados a él. Poniendo la sintaxis reflejará los pasos a seguir:

ELIMINANDO UN ESQUEMA DROP SCHEMA

```
DROP TABLE NOMBRE_TABLA
GO
```

```
DROP SCHEMA NOMBRE_ESQUEMA AUTHORIZATION NOMBRE_PROPIETARIO
```

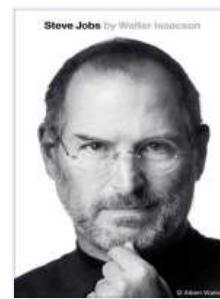
8 views

1 0

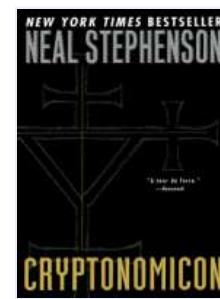
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

66

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 2.6

Implemente los esquemas RRHH, CONTABILIDAD y PAGO dentro de la base de datos AGENCIACD5.

```
--1.
USE AGENCIACD5
GO

--2.
CREATE SCHEMA RRHH AUTHORIZATION DBO
GO
CREATE SCHEMA CONTABILIDAD AUTHORIZATION DBO
GO
CREATE SCHEMA PAGO AUTHORIZATION DBO
GO

--3.
SELECT * FROM SYS.SCHEMAS
WHERE PRINCIPAL_ID=1
GO
```

En el punto uno se activa la base de datos AGENCIACD5 para poder crear los esquemas dentro

En el punto dos se crean tres esquemas dentro de la base de datos activa, tener en cuenta que el propietario es DBO ya que no hemos definido propietarios ni roles.

En el punto tres se verifica la existencia de los esquemas dentro de la base de datos activa. Tenemos que hay varios esquemas dentro de una base de datos, pero la identificación asociada al proyecto es uno por ese motivo la consulta se condiciona a ese valor por la columna PRINCIPAL_ID.

2.12. Los TIPOS DE DATOS EN SQL SERVER 2012

SQL brinda una serie de datos para almacenar la información, la correcta selección del tipo de dato implica un determinado valor a almacenar; por ejemplo: Carácter, Enteros, Binario, Fechas, etc.

TIPO DE DATOS CARÁCTER

Los datos carácter tienen una combinación de letras, símbolos y números. Tenemos los siguientes tipos de datos:

Char	Los datos deben tener una longitud fija (Hasta 8KB). Nombraremos algunos ejemplos para poder darte una idea de la longitud de los campos: <ul style="list-style-type: none"> - El Documento Nacional de Identificación DNI que cuenta con 8 caracteres fijos, entonces SQL lo declarar como DNI CHAR(8). - Una determinada empresa categoriza a sus trabajadores entre A, B, C y D, entonces SQL lo declarar como CATEGORIA CHAR(1). - En la base de datos de una Agenda se desea registrar el número telefónico de varias personas, este podría declararlo como TELEFONO CHAR(7) hay que tener en cuenta que los números telefónicos tienen más de 7 dígitos, en función de la región o país; el número 7 es simplemente referencial.
------	---

8 views

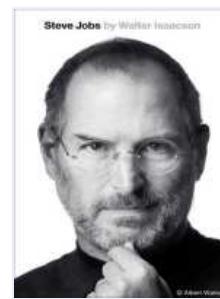
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

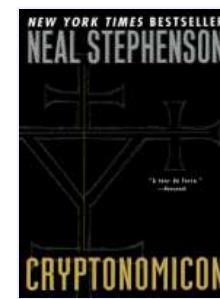
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A Tru

CAP. 2: GESTIÓN DE BASE DE DA

TIPO DE DATOS BINARIOS

Los datos tipo Binario almacenan cadenas de bits. La data consiste en número Hexadecimales.

Binary	La data tiene una longitud fija (8KB).
varBinary	Los datos pueden variar en el número de dígitos hexadecimales (Hasta 8KB).
Image	La data puede tener una longitud variable y exceder los 8KB.

TIPO DE DATOS FECHA

Las datos fecha y hora consisten en combinaciones válidas de estos datos.

DateTime	Fechas en el Rango del 01 Enero del 1753 al 31 Diciembre del 9999. 08-05-2007 12:35:29.123
SmallDateTime	Fechas en el Rango del 01 enero 1900 al 06 Junio 2079. 08-05-2007 12:35:00
DateTime2	08-05-2007 12:35:29.1234567
Time	12:35:29.1234567
Date	08-05-2007
DateTimeOffset	08-05-2007 12.35.29.123467 +12:15

TIPO DE DATOS DECIMAL

Datos decimal consisten en información que almacena información significativa después del punto decimal.

Decimal	Los datos pueden tener hasta 38 dígitos, podrán estar en el lado derecho del punto decimal.
Numeric	Es equivalente a Decimal

TIPO DE DATOS PUNTO FLOTANTE

Números aproximados (Punto Flotante)

Float	Datos en el Rango de -1.79E +308 hasta 1.79E +308
Real	Datos en el Rango de -3.40E +38 hasta 3.40E +38

TIPO DE DATOS ENTERO

Consiste en almacenar datos numéricos positivos o negativos.

BigInt	De -2^{63} (-9.223.372.036.854.775.808) a $2^{63}-1$ (9.223.372.036.854.775.807)
Int	De -2^{31} (-2.147.483.648) a $2^{31}-1$ (2.147.483.647)
Smallint	De -2^{15} (-32.768) a $2^{15}-1$ (32.767)
Tinyint	De 0 a 255

8 views

1 like

0 comments

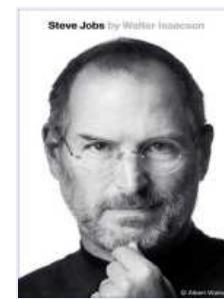
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

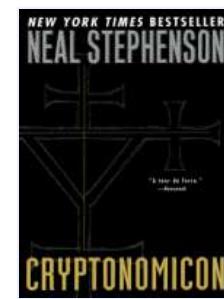
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in History

68

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

2.13. TIPOS DE DATOS DEFINIDOS POR EL USUARIO

En vista que SQL Server nos proporciona una serie de Tipos de Datos a veces necesitaremos uno especial. Como el caso del DNI que cuenta con 8 caracteres de tipo Char o el número telefónico que tiene 9 caracteres.

Sintaxis de creación:

CREANDO UN TIPO DEFINIDO SP_ADDTYPE

```
SP_ADDTYPE NOMBRE_TIPO,'TIPO_DATOS','RESTRICCION'
GO
```

Sintaxis de eliminación de esquemas:

Para eliminar un esquema debe tener en cuenta que no tenga objetos asociados a él. Poniendo la sintaxis reflejará los pasos a seguir:

ELIMINANDO UN TIPO DEFINIDO SP_DROPTYPE

```
SP_DROPTYPE 'NOMBRE_TIPO'
GO
```

CASO DESARROLLADO N° 2.7

Implemente un script que permita añadir el tipo de datos DNI dentro de la base de datos AGENCIACD5 además de restringirle que no debe contener espacios vacíos.

```
--1.
USE AGENCIACD5
GO

--2.
SP_ADDTYPE DNI,'CHAR(8)', 'NOT NULL'
GO

--3.
SELECT S.NAME,S.XTYPE,S.LENGTH
      FROM SYS.SYSTYPES S
      GO
```

En el punto uno se activa la base de datos AGENCIACD5 ya que será aquí donde se agreguen los tipos de datos definidos por el usuario.

En el punto dos se agrega el tipo de datos DNI de 8 caracteres con la restricción NOT NULL.

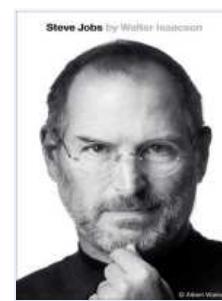
8 views

1 0

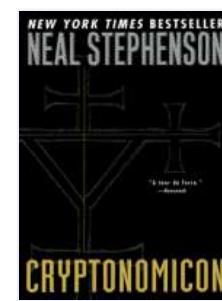
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the World's Greatest Codebreakers

CAP. 2: GESTIÓN DE BASE DE DATOS

En el script anterior se muestra la implementación de la creación de una tabla llamada PASAJERO en la cual se hace uso del nuevo tipo llamado DNI que tenga en cuenta que el tipo DNI ya debe estar en la base de datos.

Veamos el caso que no se restrinja el **NOT NULL** en la creación del tipo de datos DNI, la estructura de la tabla sería de la siguiente forma:

```
CREATE TABLE PASAJERO(
    IDPASAJERO      CHAR(5)          NOT NULL,
    NOMBRES         VARCHAR(50)       NOT NULL,
    DNI              DNI              NOT NULL
)
GO
```

En el punto tres se muestra el script que permitirá verificar los tipos de datos habilitados dentro de la base de datos activa.

La imagen siguiente muestra que el tipo de datos DNI se ubica en el último lugar de los tipos en AGENTACD5.

	name	xtype	length
27	varchar	167	8000
28	binary	173	8000
29	char	175	8000
30	timestamp	189	8
31	nvarchar	231	8000
32	nchar	239	8000
33	xml	241	-1
34	systemname	231	256
35	DNI	175	8

2.14. PROPIEDADES DE LOS CAMPOS

A partir de este punto comenzaremos con los pasos previos a la creación de tablas dentro de la base de datos activa, sólo nos falta definir dos propiedades comunes entre los campos de una tabla:

- Propiedad NULL
- Propiedad IDENTITY

La propiedad NULL es la más común entre las asignaciones que se puede realizar al momento de crear una tabla. Tenemos dos formas de expresar el término NULL, puesto que al señalar NULL estamos indicando que el contenido de dicha columna no es obligatorio; por lo tanto, no devuelve valores nulos. Si se necesita especificar lo contrario, es decir, que se obligue a un valor dentro de la columna, entonces se tendrá que especificar con NOT NULL. Veamos el

8 views

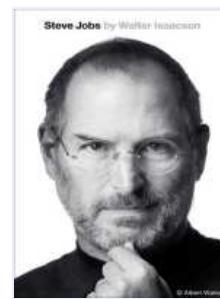
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

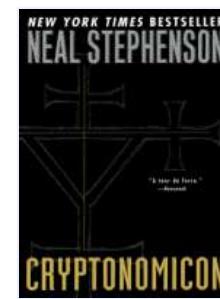
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

70

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Si se necesita crear una tabla PASAJERO con la estructura anterior hay que tener en cuenta que las columnas IDPASAJERO y NOMBRES son obligatorias en ser registradas y que la columna EMAIL no es obligatoria. Considerese que cuando no se especifica también se refiere a un valor de tipo NULL.

La propiedad IDENTITY sólo es aplicable a columnas de tipo numérico ya que define un autómatas que generan valores que pueden representar una numeración de valores en forma automática por lo tanto dentro de la tabla. Por ejemplo, podría tratarse de una tabla de facturas y su columna NUMBER tenga asignada la propiedad IDENTITY para identificar el número de factura registrada en cada caso:

```
CREATE TABLE PAGO(
    NUMPAGO INT IDENTITY(1,1),
    FECHA DATE NOT NULL,
    MONTO MONEY NOT NULL,
)
GO
```

En el script anterior se crea la tabla PAGO que tiene la columna NUMPAGO de tipo INT con autoincremento de inicio 1 con un salto de 1.

2.15. LAS TABLAS

Son objetos compuestos por una estructura (columnas) que almacenan información en forma interrelacionada entre ellos formando filas acerca de un objeto en general.

Las tablas son representaciones de la entidad del mundo real; por lo tanto, las columnas existentes en ella, serán las características de una entidad y los valores ingresados serán los datos que representan un hecho real.

PASAJERO	
ID	IDPASAJERO
NOMBRE	NOMBRES
PAÍS	IDPAIS
TELÉFONO	TELEFONO
EMAIL	EMAIL

La tabla pasajero representa a un pasajero del mundo real el cual cuenta con un código para identificarse como pasajero de la empresa, sus nombres, su código de país, un teléfono y un correo electrónico. Desde aquí tendremos que definir las capacidades de cada columna ya que eso derivará en el ingreso de los valores a dichas columnas.

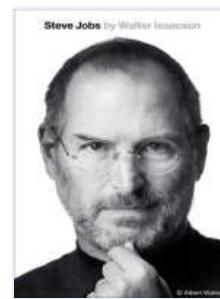
8 views

1 0

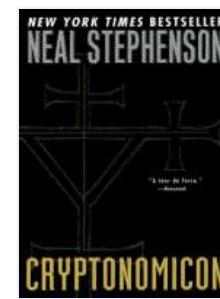
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

CAP. 2: GESTIÓN DE BASE DE DATOS

2.16. IMPLEMENTACIÓN DE TABLAS CON PROPIETARIO DBO

Sintaxis de creación:

CREANDO UNA TABLA CREATE TABLE

```
CREATE TABLE PROPIETARIO.NOMBRE_TABLA (
    CAMPO1 TIPO_DATOS RESTRICCION,
    CAMPO2 TIPO_DATOS RESTRICCION,
    CAMPO3 TIPO_DATOS RESTRICCION
)
GO
```

Donde:

- **Propietario:** Dependerá si tiene un esquema predefinido en la base de datos, si no definir entonces no es necesario especificar un propietario ya que se tiene un propietario nativo
- **Campo:** Es el nombre de la columna que se le asignará, recuerde que debe tener valor contenido y no debe tener espacio en blanco.
- **Tipo_Datos:** Estos tipos de datos ya se encuentran pre-establecidos por SQL Server podemos usar los tipos definidos por el usuario.
- **Restricción:** Se aplica para dar consistencia al valor que se registrarán en las columnas Null, Not Null, Default, Primary Key, etc.

CASO DESARROLLADO N° 2.8

Implemente un script que permita crear la tabla PASAJERO con las especificaciones mostradas en el modelo siguiente:

PASAJERO	
Column Name	Condensed Type
IDPASAJERO	char(5)
NOMBRES	varchar(50)
IDPAIS	char(4)
TELEFONO	char(15)
EMAIL	varchar(50)

```
USE AGENCIACD5
GO
```

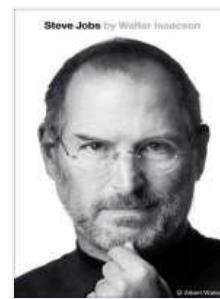
8 views

1 0

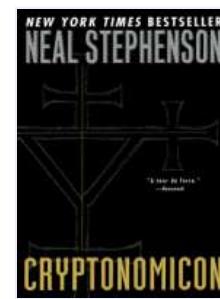
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

72

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Tome en cuenta que esta implementación sólo crea la tabla no agrega restricciones ni mi campos claves. También se pudo haber definido un propietario de la tabla de la siguiente:

```
CREATE TABLE DBO.PASAJERO(
    IDPASAJERO      CHAR(5)          NOT NULL,
    NOMBRES         VARCHAR(50)       NOT NULL,
    IDPAIS          CHAR(4)          NOT NULL,
    TELEFONO        CHAR(15)         NOT NULL,
    EMAIL           VARCHAR(50)       NOT NULL
)
GO
```

2.17. IMPLEMENTACIÓN DE TABLAS CON ESQUEMAS

Primero veamos la sintaxis de la implementación de esquemas en las tablas:

Sintaxis de creación:

**CREANDO UN ESQUEMA
CREATE SCHEMA**

```
CREATE SCHEMA NOMBRE_ESQUEMA AUTHORIZATION DBO
GO
```

Veremos un ejemplo de implementación de esquemas sobre la base de datos AGENCIACD5, cual se definirán 3 esquemas que estarán distribuidas de la siguiente manera:

- RESERVACIONES
- VUELO
- ASIENTO
- TARIFA
- RESERVA
- PAGO
- LOGISTICA
- AVION
- AEROLINEA
- AEROPUERTO
- PAIS
- CLIENTES
- PASAJERO

```
USE AGENCIACD5
GO

CREATE SCHEMA RESERVACIONES AUTHORIZATION DBO
GO
CREATE SCHEMA LOGISTICA AUTHORIZATION DBO
GO
CREATE SCHEMA CLIENTES AUTHORIZATION DBO
GO

CREATE TABLE RESERVACIONES.VUELO(
    NUMVUELO      INT             NOT NULL,
    IDAERO        CHAR(5)         NOT NULL,
```

8 views

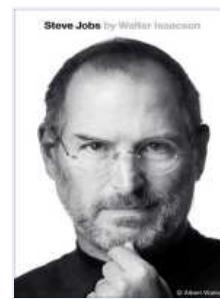
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

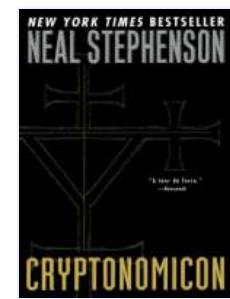
Uploaded by [anon_61286589](#)[Full description](#)

 Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

CAP. 2: GESTIÓN DE BASE DE DATOS

```

CREATE TABLE RESERVACIONES.TARIFA(
    CLASE          VARCHAR(20)      NOT NULL,
    PRECIO          MONEY           NOT NULL,
    IMPUESTO        MONEY           NOT NULL
)
GO

CREATE TABLE RESERVACIONES.RESERVA(
    IDRESERVA      INT             IDENTITY NOT NULL,
    COSTO           MONEY           DEFAULT 0,
    FECHA           DATE            DEFAULT GETDATE()
)
GO

CREATE TABLE RESERVACIONES.PAGO(
    NUMPAGO         INT             NOT NULL,
    IDRESERVA       INT             NOT NULL,
    IDPASAJERO      CHAR(5)         NOT NULL,
    FECHA           DATE            NOT NULL,
    MONTO           MONEY           NOT NULL
)
GO

CREATE TABLE LOGISTICA.AVION(
    IDAVION         CHAR(5)         NOT NULL,
    RUC              CHAR(11)        NOT NULL,
    COMPAÑIA        VARCHAR(40)     NOT NULL,
    TIPO             VARCHAR(30)     NOT NULL,
    PASAJEROS        INT             NOT NULL
)
GO

CREATE TABLE LOGISTICA.AEROLINEA(
    RUC              CHAR(11)        NOT NULL,
    NOMBRE           VARCHAR(40)     NOT NULL
)
GO

CREATE TABLE LOGISTICA.AEROPUERTO(
    IDAERO           CHAR(5)         NOT NULL,
    NOMBRE           VARCHAR(40)     NOT NULL,
    IDPAIS           CHAR(4)         NOT NULL
)
GO

CREATE TABLE LOGISTICA.PAIS(
    IDPAIS           CHAR(4)         NOT NULL,
    PAIS              VARCHAR(30)     NOT NULL
)
GO

CREATE TABLE CLIENTES.PASAJERO(
    IDPASAJERO      CHAR(5)         NOT NULL,
    NOMBRES          VARCHAR(50)     NOT NULL,
    IDPAIS           CHAR(4)         NOT NULL,
    TELEFONO         CHAR(15)        NOT NULL,
)
  
```

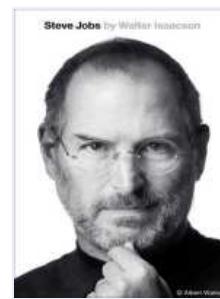
8 views

1 0

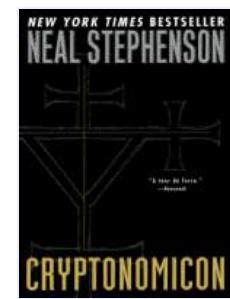
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

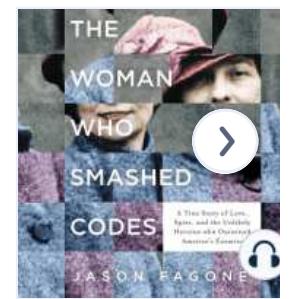
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

74

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

La imagen siguiente muestra el Diagrama de Base de Datos hasta el momento:



Note que cada título de tabla tiene entre paréntesis el esquema al que pertenece, en el caso de que no se especifique el esquema en la creación de la tabla su esquema será DBO y no presentarán ninguno.

2.18. DEFINICIÓN DE LAS LLAVES PRIMARIAS Y FORÁNEAS

Es un tipo de restricción estructural que permite la vinculación de los datos a otras tablas. Es obligatoria la consistencia de las mismas.

Sintaxis de asignación de llaves primarias:

**LLAVE PRIMARIA
PRIMARY KEY**

```
ALTER TABLE NOMRRF_TARIA
```

8 views

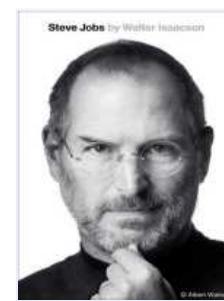
1 like

0 comments

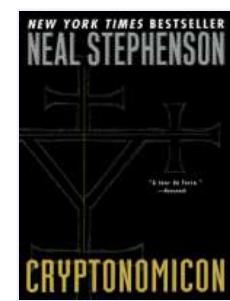
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

CAP. 2: GESTIÓN DE BASE DE DATOS

Sintaxis de asignación de llaves foráneas:

LLAVE FORANEA FOREIGN KEY

```
ALTER TABLE NOMBRE_TABLA
ADD FOREIGN KEY (COLUMNNA) REFERENCES TABLA_ASOCIADA(COLUMNNA_ASOCIA)
```

CASO DESARROLLADO N° 2.9

Implemente un script que permita añadir la restricción estructural campo clave a IDPASAJERO de la tabla PASAJERO.

PASAJERO	
Column Name	Condensed Type
IDPASAJERO	char(5)
NOMBRES	varchar(50)
IDPAIS	char(4)
TELEFONO	char(15)
EMAIL	varchar(50)

--1.

```
CREATE TABLE PASAJERO(
    IDPASAJERO      CHAR(5)      NOT NULL,
    NOMBRES         VARCHAR(50)   NOT NULL,
    IDPAIS          CHAR(4)      NOT NULL,
    TELEFONO        CHAR(15)     NOT NULL,
    EMAIL           VARCHAR(50)   NOT NULL
)
GO
```

--2.

```
ALTER TABLE PASAJERO
    ADD PRIMARY KEY NONCLUSTERED (IDPASAJERO)
GO
```

En el punto uno se crea la tabla PASAJERO en la cual define sus columnas y solamente las hace obligatorias con la restricción NOT NULL. Recuerde que un campo clave tiene que tener la restricción NOT NULL obligatoriamente.

En el punto dos se modifica la tabla PASAJERO agregando la llave primaria a la tabla en la cual hace referencia a la columna IDPASAJERO.

También podríamos representar el script en un solo bloque de instrucciones de la siguiente forma:

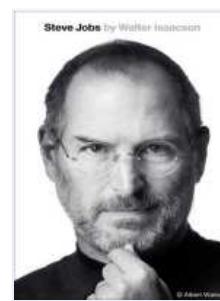
8 views

1 0

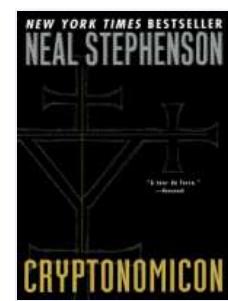
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

76

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012**CASO DESARROLLADO N° 2.10**

Implemente un script que permita añadir la restricción estructural entre las columnas IDPAIS de la tabla PASAJERO asociado a la tabla PAIS.

PASAJERO	
Column Name	Condensed Type
IDPASAJERO	char(5)
NOMBRES	varchar(50)
IDPAIS	char(4)
TELEFONO	char(15)
EMAIL	varchar(50)

PAIS	
Column Name	Condensed Type
IDPAIS	char(4)
PAIS	varchar(30)

--1.

```
CREATE TABLE PASAJERO(
    IDPASAJERO      CHAR(5)      NOT NULL,
    NOMBRES         VARCHAR(50)   NOT NULL,
    IDPAIS          CHAR(4)      NOT NULL,
    TELEFONO        CHAR(15)     NOT NULL,
    EMAIL           VARCHAR(50)   NOT NULL
)
GO
```

CREATE TABLE PAIS(

```
    IDPAIS          CHAR(4)      NOT NULL,
    PAIS            VARCHAR(30)   NOT NULL
)
GO
```

--2.

```
ALTER TABLE PASAJERO
    ADD PRIMARY KEY NONCLUSTERED (IDPASAJERO)
GO
ALTER TABLE PAIS
    ADD PRIMARY KEY NONCLUSTERED (IDPAIS)
GO
```

--3.

```
ALTER TABLE PASAJERO
    ADD FOREIGN KEY (IDPAIS) REFERENCES PAIS
GO
```

En el punto uno se implementa las tablas PASAJERO y PAIS con sus respectivas columnas.

En el punto dos se agrega los campos claves a ambas tablas en el caso de PASAJERO el car-

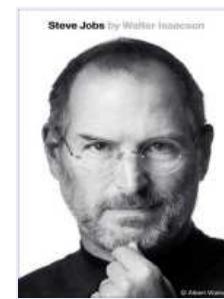
8 views

1 0

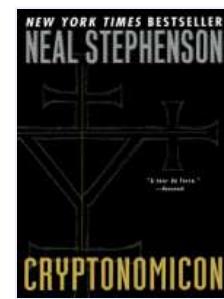
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

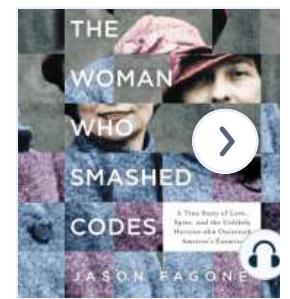
RELATED TITLES



Steve Jobs



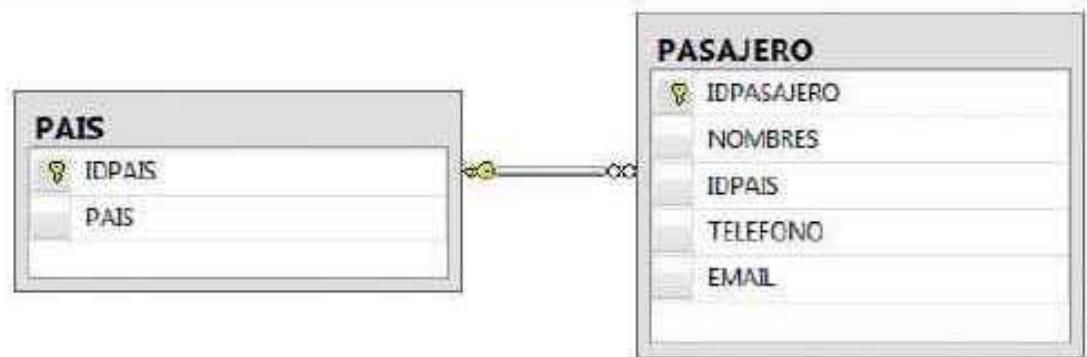
Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 2: GESTIÓN DE BASE DE DATOS

El resultado de la asociación se muestra en la siguiente imagen:



2.19. RESTRICCIONES DE LOS CAMPOS: UNIQUE, CHECK Y DEFAULT

UNIQUE: permite determinar que los valores registrados en una misma columna no sea es decir, se mantengan únicos, como ya vimos anteriormente el campo clave también tiene restricción, pero existe una diferencia entre ellos en la cual el campo clave sólo se puede restringir con campos específicos mientras que a todas las columnas restantes se puede restringir con L que considerar que es sólo un decir puesto que todas las columnas de una tabla no son car

Debemos tener en cuenta que una columna definida como UNIQUE acepta a lo más un mientras que una columna restringida con PRIMARY KEY no permite dicha acción.

CASO DESARROLLADO N° 2.11

Implemente un script que permita añadir la restricción UNIQUE a la columna PAÍS, esto permitira registrar nombre de paises sin repetirse.

PAÍS	
Column Name	Condensed Type
IDPAIS	char(4)
PAÍS	varchar(30)

```

--1.
ALTER TABLE PAÍS
ADD CONSTRAINT UN_PAÍS UNIQUE(PAÍS)
GO

--2.
INSERT INTO PAÍS
VALUES('0014','PERU')
GO
  
```

En el punto uno se implementa la restricción a la columna PAÍS de la tabla PAÍS. note que

8 views

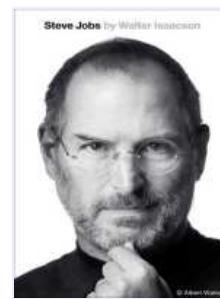
1 like

0 comments

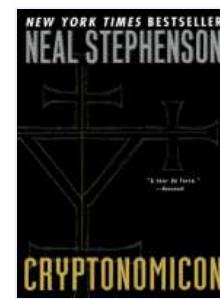
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

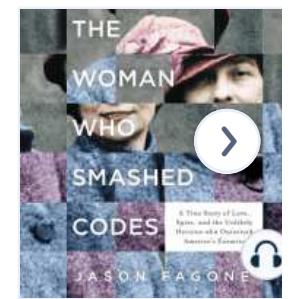
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

78

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Otra forma de representar la misma acción es colocando el siguiente script:

```
CREATE TABLE PAIS(
    IDPAIS CHAR(4) NOT NULL,
    PAIS VARCHAR(30) NOT NULL UNIQUE
)
GO
```

CHECK: permite restringir el rango de valores que pueden estar permitidos ingresar en columnas de una tabla.

CASO DESARROLLADO N° 2.12

Implemente un script que permita añadir la restricción a la columna FECHA de la tabla PAGO sólo permita registrar fechas menores o iguales al día actual.

PAGO	
Column Name	Condensed Type
NUMPAGO	int
IDRESERVA	int
IDPASAJERO	char(5)
FECHA	date
MONTO	money

```
--1.
ALTER TABLE PAGO
    ADD CONSTRAINT CHK_FECHA CHECK (FECHA<=GETDATE())
GO

--2.
INSERT INTO PAGO
    VALUES(1,1,'P0002','20/08/2013',500)
GO
```

En el punto uno se aplica la restricción a la columna FECHA con el nombre de **CHK_FECHA** en cuenta que restringir a nivel CHECK lleva a una expresión encerrada entre paréntesis obligatoriamente.

En el punto dos se intenta registrar el pago número 1 con fecha 20/08/2013 siendo hoy 19/08/2013 el mensaje de error se muestra en la siguiente imagen:

```
Msg 547, Level 15, State 0, Line 1
The INSERT statement conflicted with the CHECK constraint "CHK_FECHA".
The conflict occurred in database "ACCREDITACION", table "PAGO", column "FECHA".
```

8 views

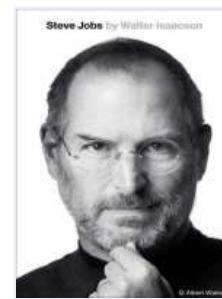
1 like

0 comments

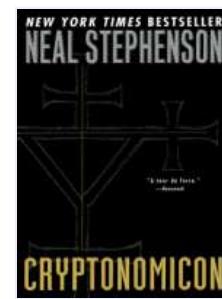
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CASO DESARROLLADO N° 2.13

Implemente un script que permita añadir la restricción a la columna FECHA de la tabla PAGO si el usuario no registra la fecha se inserte la fecha actual por defecto.

PAGO	
Column Name	Condensed Type
NUMPAGO	int
IDRESERVA	int
IDPASAJERO	char(5)
FECHA	date
MONTO	money

```
--1.
ALTER TABLE PAGO
    ADD CONSTRAINT DFL_FECHA DEFAULT GETDATE() FOR FECHA
GO

--2.
INSERT INTO PAGO
    VALUES(1,2,'P0002','20/08/2013',500)
GO

INSERT INTO PAGO
    VALUES(1,2,'P0002',DEFAULT,500)
GO
```

En el punto uno se agrega la restricción DFL_FECHA a la tabla PAGO, no se olvide de se sintaxis puesto que es totalmente distinta a las implementaciones anteriores.

En el punto dos se muestran dos tipos de inserción sobre dicha columna en la primera se fecha para la cual el valor por defecto no se aplica, mientras que en la segunda no se es fecha más bien se invoca al valor por defecto con la cláusula DEFAULT, es decir se insert actual en el registro.

En el caso se quisiera eliminar una restricción se debe alterar la tabla aplicando un DROP C

Veamos un script donde elimine la restricción por defecto de fecha.

```
ALTER TABLE PAGO
    DROP CONSTRAINT DFL_FECHA
GO
```

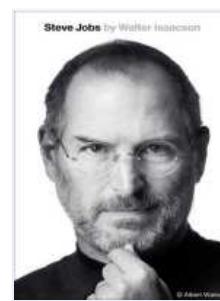
8 views

1 0

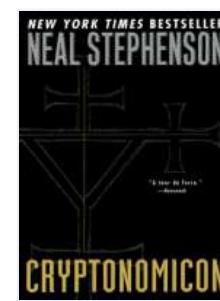
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

80

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012**2.20. Esquema de la base de datos AGENCIA para el uso de los casos desarrollo**

Como ya lo habíamos mencionado líneas anteriores todos los capítulos de este material trae base a un solo caso, para lo cual mostraremos a continuación todo el script a usar más final de base de datos:

```
--LA BASE DE DATOS
IF DB_ID('AGENCIA') IS NOT NULL
BEGIN
    USE MASTER
    DROP DATABASE AGENCIA
END

CREATE DATABASE AGENCIA
GO

--ACTIVACIÓN DE LA BASE AGENCIA
USE AGENCIA
GO

--IMPLEMENTANDO LAS TABLAS
CREATE TABLE AEROLINEA(
    RUC CHAR(11) NOT NULL,
    NOMBRE VARCHAR(40) NOT NULL
)
GO

CREATE TABLE AVION(
    IDAVION CHAR(5) NOT NULL,
    RUC CHAR(11) NOT NULL,
    COMPAÑIA VARCHAR(40) NOT NULL,
    TTPO VARCHAR(30) NOT NULL,
    PASAJEROS INT NOT NULL
)
GO

CREATE TABLE TARIFA(
    CLASE VARCHAR(20) NOT NULL,
    PRECIO MONEY NOT NULL,
    IMPUESTO MONEY NOT NULL
)
GO

CREATE TABLE RESERVA(
    IDRESERVA INT IDENTITY NOT NULL,
    COSTO MONEY DEFAULT 0,
    FECHA DATE DEFAULT GETDATE()
)
GO

CREATE TABLE PASAJERO/

```

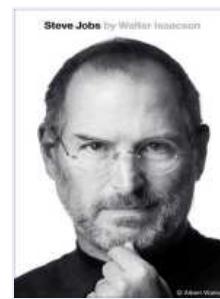
8 views

1 0

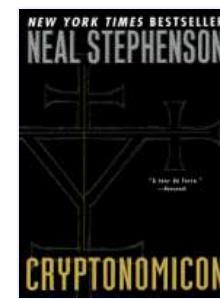
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

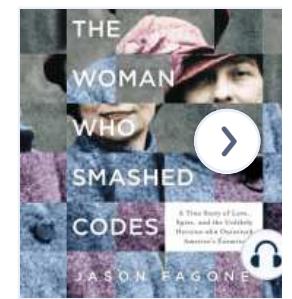
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

CAP. 2: GESTIÓN DE BASE DE DATOS

```

CREATE TABLE PAIS(
    IDPAIS CHAR(4) NOT NULL,
    PAIS VARCHAR(30) NOT NULL
)
GO

CREATE TABLE ASIENTO(
    NUMVUELO INT NOT NULL,
    LETRA CHAR(2) NOT NULL,
    FILA INT NOT NULL
)
GO

CREATE TABLE PAGO(
    NUMPAGO INT NOT NULL,
    IDRESERVA INT NOT NULL,
    IDPASAJERO CHAR(5) NOT NULL,
    FECHA DATE NOT NULL,
    MONTO MONEY NOT NULL
)
GO

CREATE TABLE AEROPUERTO(
    IDAERO CHAR(5) NOT NULL,
    NOMBRE VARCHAR(40) NOT NULL,
    TDPAWS CHAR(4) NOT NULL
)
GO

CREATE TABLE VUELO(
    NUMVUELO INT NOT NULL,
    IDAERO CHAR(5) NOT NULL,
    IDRESERVA INT NOT NULL,
    IDAVION CHAR(5) NOT NULL,
    CLASE VARCHAR(20) NOT NULL
)
GO

--IMPLEMENTADO LAS LLAVES PRIMARIAS
ALTER TABLE AEROLINEA
    ADD PRIMARY KEY NONCLUSTERED (RUC)
ALTER TABLE AVION
    ADD PRIMARY KEY NONCLUSTERED (IDAVION)
ALTER TABLE AEROPUERTO
    ADD PRIMARY KEY NONCLUSTERED (IDAERO)
ALTER TABLE TARIFA
    ADD PRIMARY KEY NONCLUSTERED (CLASE)
ALTER TABLE VUELO
    ADD PRIMARY KEY NONCLUSTERED (NUMVUELO, IDAERO, IDRESERVA, IDAVION)
ALTER TABLE RESERVA
    ADD PRIMARY KEY NONCLUSTERED (IDRESERVA)
ALTER TABLE PAIS
    ADD PRIMARY KEY NONCLUSTERED (TDPAWS)

```

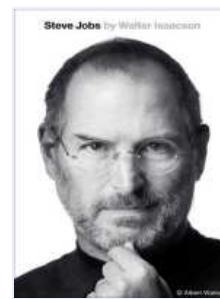
8 views

1 0

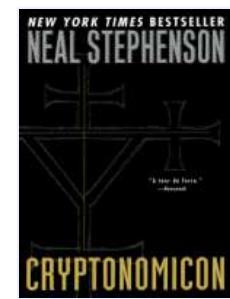
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

82

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

```
--IMPLEMENTANDO LAS LLAVES SECUNDARIAS
ALTER TABLE PAGO
    ADD FOREIGN KEY (IDRESERVA) REFERENCES RESERVA
ALTER TABLE PAGO
    ADD FOREIGN KEY (IDPASAJERO) REFERENCES PASAJERO
ALTER TABLE AVION
    ADD FOREIGN KEY (RUC) REFERENCES AEROLINEA
ALTER TABLE VUELO
    ADD FOREIGN KEY (IDAERO) REFERENCES AEROPUERTO
ALTER TABLE VUELO
    ADD FOREIGN KEY (IDRESERVA) REFERENCES RESERVA
ALTER TABLE VUELO
    ADD FOREIGN KEY (IDAVION) REFERENCES AVION
ALTER TABLE VUELO
    ADD FOREIGN KEY (CLASE) REFERENCES TARIFA
ALTER TABLE VUELO
    ADD FOREIGN KEY (NUMVUELO) REFERENCES ASIENTO
ALTER TABLE AEROPUERTO
    ADD FOREIGN KEY (IDPAIS) REFERENCES PAIS
ALTER TABLE PASAJERO
    ADD FOREIGN KEY (IDPAIS) REFERENCES PAIS
GO

--LLENADO DE REGISTROS
INSERT INTO AEROLINEA VALUES('10123456789','LAN PERU')
TNSERT TNTO AFROI TNFA VALIUES('10123456710','AFROPRIII')
INSERT INTO AEROLINEA VALUES('10123456711','TACA')
INSERT INTO AEROLINEA VALUES('10123456712','BIRD PERU')
INSERT INTO AEROLINEA VALUES('10123456713','LAN CUSCO')

INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0001','PERU')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0002','ARGENTINA')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0003','CHILE')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0004','ECUADOR')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0005','BRASIL')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0006','VENEZUELA')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0007','PARAGUAY')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0008','URUGUAY')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0009','BOLIVIA')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0010','MEXICO')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0011','HONDURAS')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0012','EEUU')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0013','PUERTO RICO')

SET DATEFORMAT DMY
GO

INSERT INTO RESERVA (COSTO,FECHA)
VALUES (0,'01/10/11'),
       (0,'06/10/11'),
       (0,'14/11/11'),
       (0,'16/11/11'),
       (0,'12/12/11')
```

8 views

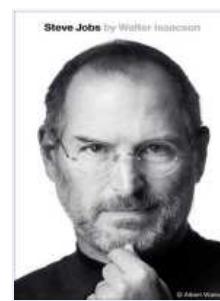
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

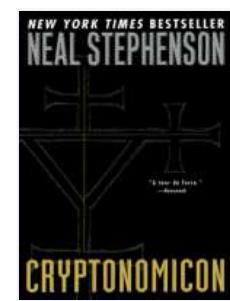
Uploaded by [anon_61286589](#)[Full description](#)

 Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Last Great Unsolved Puzzles of Cryptography

CAP. 2: GESTIÓN DE BASE DE DATOS

```

INSERT INTO PASAJERO
VALUES ('P0001', 'ANGELA TORRES LAZARO', '0001',
'99999999', 'ATORRES@HOTMAIL.COM'),
('P0002', 'FERNANDA TORRES LAZARO', '0001',
'99999999', 'FTORRES@HOTMAIL.COM'),
('P0003', 'MARIA ZAMORA MEJIA', '0005',
'957564526', 'MZAMORA@GMAIL.COM'),
('P0004', 'GUADALUPE ACOSTA FERRER', '0002',
'957564526', 'GACOSTA@HOTMAIL.COM'),
('P0005', 'LUZ LAZARO MENOR', '0001',
'99999999', 'LLAZARO@GMAIL.COM'),
('P0006', 'KARLA GALLEGOS SILVA', '0007',
'957564526', 'KGALLEGOS@HOTMAIL.COM'),
('P0007', 'NERY CALLE DE LA CRUZ', '0010',
'957564526', 'NCALLE@GMAIL.COM'),
('P0008', 'HEIDI RENGIFO REATEGUI', '0004',
'957564526', 'HRENGIFO@HOTMAIL.COM'),
('P0009', 'MARISOL DIAZ ZAMBRANO', '0004',
'957564526', 'MDIAZ@GMAIL.COM'),
('P0010', 'LINDA TUME VARAS', '0008',
'957564526', 'LTUME@HOTMAIL.COM')
  
```

INSERT INTO PAGO

```

VALUES (1, 'P0005', '01/10/11', 500),
(2, 'P0003', '06/10/11', 900),
(3, 'P0008', '14/11/11', 500),
(4, 'P0002', '16/11/11', 1200),
(5, 'P0001', '12/12/11', 1500),
(6, 'P0006', '17/12/11', 590),
(7, 'P0003', '14/01/12', 400),
(8, 'P0003', '15/01/12', 1300),
(9, 'P0008', '01/02/12', 1000),
(10, 'P0002', '02/04/12', 1800),
(11, 'P0001', '09/04/12', 1200),
(12, 'P0006', '01/08/12', 400),
(13, 'P0003', GETDATE() + 1, 800)
  
```

INSERT INTO TARIFA

```

VALUES ('SUPER VIP', 1200, 12),
('VIP', 1000, 12),
('NACIONAL', 800, 12),
('ECONOMICO', 500, 0)
  
```

INSERT INTO AEROPUERTO

```

VALUES ('AE01', 'BARILOCHE', '0002'),
('AE02', 'MAR DEL PLATA', '0002'),
('AE03', 'JORGE CHAVEZ', '0001'),
('AE04', 'SANTIAGO', '0003'),
('AE05', 'AICM', '0010'),
('AE06', 'JOSE JOAQUIN DE OLMEDO', '0004'),
('AE07', 'SIMON BOLIVAR', '0006'),
('AE08', 'SAO PAULO CONGONHAS', '0005'),
('AE09', 'STEVEN DENTON', '0007')
  
```

8 views

1 0

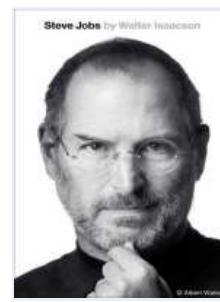
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

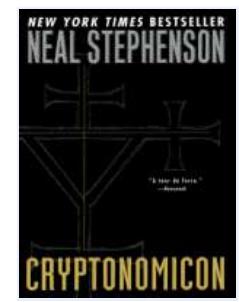
[Full description](#)

   
Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who
Smashed
Codes: A Tru

8 views

1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

[Full description](#)



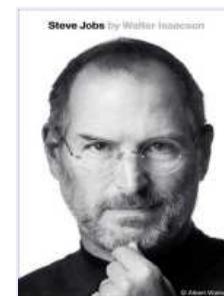
Save

Embed

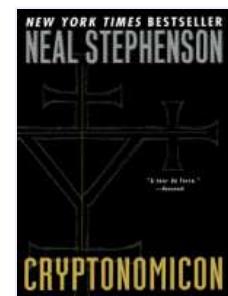
Share

Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of One of the Greatest Codebreakers in America



Lenguaje de manipulación de datos (DML)

CAPACIDAD:

El lector será capaz de aplicar instrucciones para los DML sobre una base de datos: primera parte se implementarán scripts sobre los DML Insert Into, Update y Delete.

Luego se combinan las tablas por medio de la cláusula JOIN en todas las formas. Además agrupar las columnas y mostrar consultas avanzadas.

Finalmente, se implementará la sentencia MERGE para la réplica de los registros de una o más tablas.

CONTENIDO:

- *Introducción a la manipulación de datos*
- *Insertar registros con INSERT INTO*
- *Modificación y actualización de datos de una tabla con UPDATE*
- *Eliminación de registros de una tabla con DELETE*

8 views

1 0

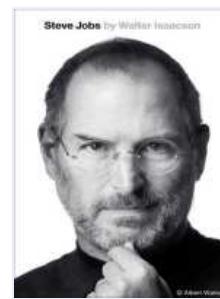
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

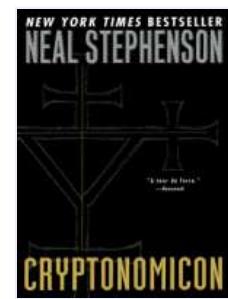
[Full description](#)

   
Save Embed Share Print

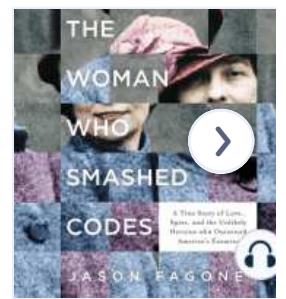
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who
Smashed
Codes: A Tru

8 views

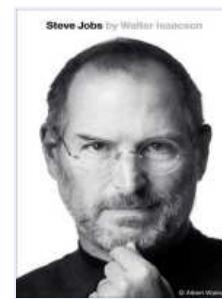
1 like

0 comments

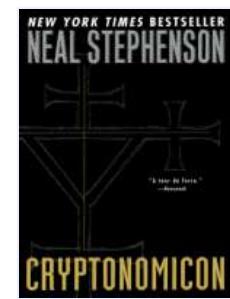
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon

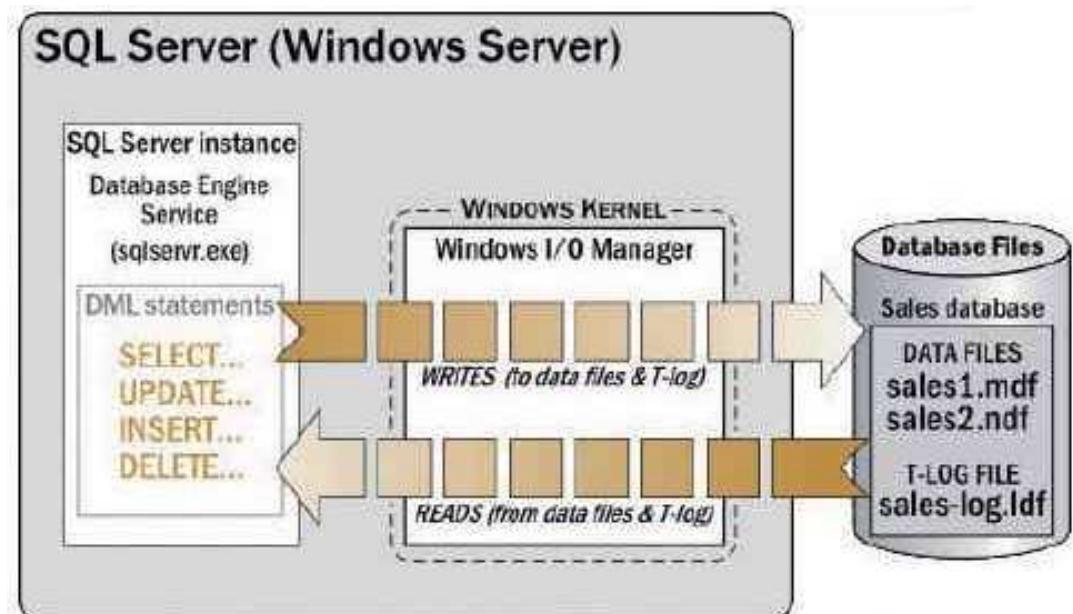


The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DML)

3.1. INTRODUCCIÓN A LA MANIPULACIÓN DE DATOS

El lenguaje de manipulación de datos (DML) es un término usado para recuperar y trabajar en SQL Server 2012. Incluye las instrucciones para agregar, modificar, consultar o quitar datos de la base de datos de SQL Server.



Las siguientes sentencias son de categoría DML:

- INSERT
- UPDATE
- SELECT
- DELETE
- MERGE
- BULK INSERT

3.2. INSERTAR REGISTROS CON INSERT INTO

La sentencia **INSERT** permite agregar una nueva fila a una tabla o vista de una determinada base de datos.

Consideraciones generales al insertar una fila nueva:

- Si desea reemplazar la fila registrada primero debe eliminarla con la sentencia **DELETE** o **TRUNCATE TABLE**, ya que ocasionaría un error de duplicidad de registro debido al código de la misma fila.
- Si desea actualizar la fila registrada debe usar la sentencia **UPDATE** condicionando el campo para una actualización personalizada.
- Para insertar un conjunto de registros a una nueva tabla debe combinar las sentencias **SELECT** y **INSERT**.
- Cuando la sentencia **INSERT** es ejecutada y detecta un error aritmético el motor de base de SQL muestra un mensaje de error desde la activación **SET ARITHABORT ON** y detiene la ejecución.

8 views

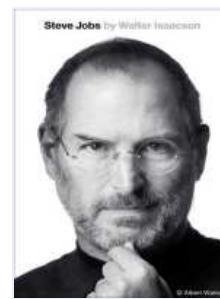
1 like

0 comments

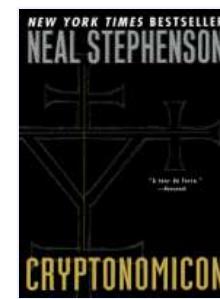
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

88 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Consideremos que la empresa **AEROCHEQUE** desea registrar sus aerolíneas dentro **AEROLINEA** para lo cual se comprobó que la estructura de ambas tablas son idénticas, va es factible enviar los registros de una tabla a otra.

Como se observa la subconsulta **SELECT IDCODIGO,LINEA FROM AEROCHEQUE** genera de registros que no puede ser considerado como una fila dentro de la tabla AEROLINEA genera un error; para una mejor implementación de este caso tenemos dos soluciones:

Usando la combinación de comandos **INSERT** y **SELECT**:

```
INSERT INTO AEROLINEA
    SELECT IDCODIGO, LINEA
    FROM AEROCHEQUE
GO
```

También podemos usar la sentencia **MERGE** como se muestra a continuación:

```
MERGE AEROLINEA AS TARGET
USING AEROCHEQUE AS SOURCE
ON (TARGET.RUC = SOURCE.IDCODIGO)
WHEN NOT MATCHED THEN
    INSERT VALUES(SOURCE.IDCODIGO, SOURCE.LINEA);
GO
```

Como observa se trata a la tabla **AEROLINEA** como el destino (**TARGET**), mientras que **AEROCHEQUE** comporta como la fuente (**SOURCE**) ya que desde aquí debemos extraer sus registros y la tabla destino, en la instrucción **ON** se debe colocar el enlace entre las 2 tablas y en la cláusula **WHEN NOT MATCHED THEN** se verifica que los registros no hayan sido registrados si todo es correcto, entonces inserta en destino desde las columnas fuentes **SOURCE.IDCODIGO, SOURCE.LINEA**.

- Con respecto a los campos declarados como **CHAR** o **VARCHAR** se tiene que considerar que si se registra un valor vacío el tipo **CHAR** rellenará de espacios vacíos hasta el límite de la longitud de la columna, mientras que el **VARCHAR** quitará los espacios finales hasta el último carácter que no sea un espacio.
- Si no especifica un valor en una columna declarada como **VARCHAR** o **TEXT** el motor de datos cargará dentro de estas columnas una cadena de longitud cero, tenga en cuenta las restricciones implementadas a dichos campos.
- Cuando se insertan valores dentro de una tabla y no se especifica las columnas a registrar, se deben colocar todos los valores obligatorios de la tabla, caso contrario se podría especificar la lista de columnas involucradas. Por ejemplo:

```
INSERT INTO AEROLINEA VALUES('10575241587', 'AEROMEXICO')
```

8 views

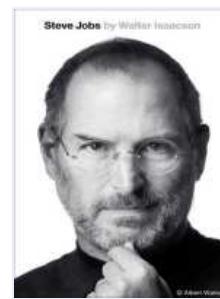
1 like

0 comments

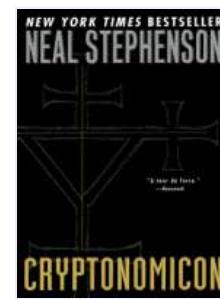
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

Sintaxis:

LMD INSERT INTO

```
INSERT [INTO]
[ESQUEMA] TABLA_O_VISTA [(Lista de columnas)]
VALUES ('Lista de Valores')
```

Donde:

- **INTO**: es una palabra clave opcional. Normalmente se usa entre el INSERT y la especifica tabla destino.
- **ESQUEMA**: es el nombre del esquema donde se encuentra registrada una tabla o la visión.
- **TABLA O VISTA**: es el nombre de la tabla o vista destino es decir la que recibirá los datos es posible especificar una variable de tipo TABLE como origen de tabla.
- **Lista de columnas**: esta permite especificar las columnas que tendrán un valor de registro define el orden de ingreso de dichos valores.
- **VALUES('Lista de Valores')**: Aquí se registran los valores según las columnas especificadas destino, se pueden registrar valores separados por comas y especificar funciones de capacidad declarada en la columna como DEFAULT, GETDATE(), etc.

Para todos los casos considere que la base de datos activa es **AGENCIA**; por tanto, no olvide el siguiente script antes de realizar los casos:

```
USE AGENCIA
GO
```

CASO DESARROLLADO N° 3.1:

Script que permita registrar los países en la tabla PAIS de la base de datos AGENCIA, usando simple de la instrucción INSERT.

PAIS	
Column Name	Condensed Type
IDPAIS	char(4)
PAIS	varchar(30)

```
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0001','PERU')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0002','ARGENTINA')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0003','CHILE')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0004','ECUADOR')
INSERT INTO PAIS (IDPAIS,PAIS) VALUES('0005','BRAZIL')
```

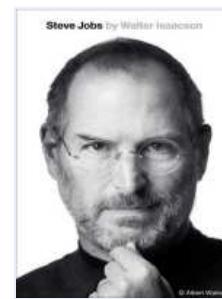
8 views

1 0

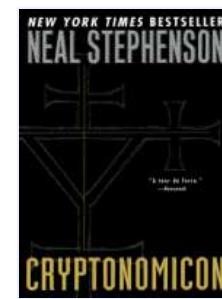
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in History

90

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el script anterior se agregan los países uno por uno y a la vez por cada línea de registros las columnas de la tabla destino, este último podría ser opcional quedando el script de forma:

```

INSERT INTO PAIS VALUES('0001','PERU')
INSERT INTO PAIS VALUES('0002','ARGENTINA')
INSERT INTO PAIS VALUES('0003','CHILE')
INSERT INTO PAIS VALUES('0004','ECUADOR')
INSERT INTO PAIS VALUES('0005','BRASIL')
INSERT INTO PAIS VALUES('0006','VENEZUELA')
INSERT INTO PAIS VALUES('0007','PARAGUAY')
INSERT INTO PAIS VALUES('0008','URUGUAY')
INSERT INTO PAIS VALUES('0009','BOLIVIA')
INSERT INTO PAIS VALUES('0010','MEXICO')
INSERT INTO PAIS VALUES('0011','HONDURAS')
INSERT INTO PAIS VALUES('0012','EEUU')
INSERT INTO PAIS VALUES('0013','PUERTO RICO')
GO

```

Hay que recalcar que este script insertará los registros especificados siempre y cuando la valores enviado a la tabla sea el mismo número de columnas especificados en la misma, asume que la tabla PAÍS tiene dos columnas.

CASO DESARROLLADO N° 3.2:

Script que permita registrar los países en la tabla **PAIS** de la base de datos **AGENCIA**, registro múltiple de filas.

PAIS	
Column Name	Condensed Type
IDPAIS	char(4)
PAIS	varchar(30)

```

INSERT INTO PAIS
VALUES('0001','PERU'),
('0002','ARGENTINA'),
('0003','CHILE'),
('0004','ECUADOR'),
('0005','BRASIL'),
('0006','VENEZUELA'),
('0007','PARAGUAY'),
('0008','URUGUAY'),
('0009','BOLIVIA'),
('0010','MEXICO'),
('0011','HONDURAS'),

```

8 views

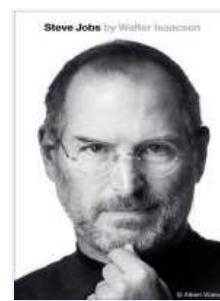
1 like

0 comments

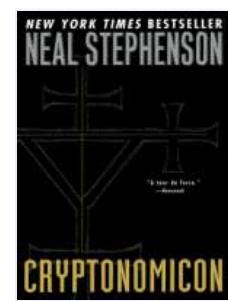
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DML)

En el script anterior no se especifica las columnas de la tabla **PAIS** ya que se le enviará de fila, si observamos la palabra **VALUES** sólo se registra una vez para muchas filas; esto sólo es cuando las filas siguientes se separan por una coma y así podrá colocar la cantidad de 1 conveniente. En el caso ocurriera un error en el centro de los registros el motor de base de las filas posteriores y registra una fila antes del error, para poder completar los registros implementar nuevamente la sentencia **INSERT INTO** ya que si se vuelve a ejecutar la sentencia un error de duplicidad de código primario.

```
Msg 2627, Level 14, State 1, Line 1
Violation of PRIMARY KEY constraint 'PK_PAIS_A5EA944B1920BESC'. Cannot insert duplicate key in object 'd
The statement has been terminated.
Msg 2627, Level 14, State 1, Line 2
Violation of PRIMARY KEY constraint 'PK_PAIS_A5EA944B1920BESC'. Cannot insert duplicate key in object 'd
The statement has been terminated.
```

Supongamos que el error ocurriera en el **IDPAIS='0008'** para poder registrar las demás filas colocar el siguiente script:

```
INSERT INTO PAIS
VALUES ('0008','URUGUAY'),
('0009','BOLIVIA'),
('0010','MEXICO'),
('0011','HONDURAS'),
('0012','EEUU'),
('0013','PUERTO RICO')
```

CASO DESARROLLADO N° 3.3:

Script que permita registrar los países en la tabla **PAÍS** de la base de datos **AGENCIA**, a orden personalizado de las columnas.

PAÍS	
Column Name	Condensed Type
IDPAIS	char(4)
PAÍS	varchar(30)

```
INSERT INTO PAIS (PAÍS, IDPAIS)
VALUES('PERU','0001'),
('ARGENTINA','0002'),
('CHILE','0003'),
('ECUADOR','0004'),
('BRASIL','0005'),
('VENEZUELA','0006'),
('PARAGUAY','0007'),
('URUGUAY','0008').
```

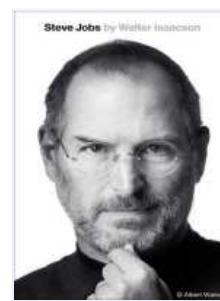
8 views

1 0

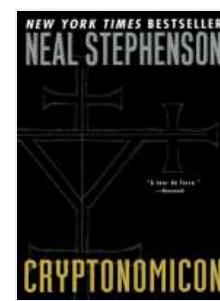
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

92

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.4:

Script que permite registrar las reservas en la tabla RESERVA de la base de datos AGENCIA menos valores que las columnas establecidas.

RESERVA					
Column Name	Condensed Type	Nullable	Identity	Default Value	
IDRESERVA	int	No	<input checked="" type="checkbox"/>		
COSTO	money	Yes	<input type="checkbox"/>	((0))	
FECHA	date	Yes	<input type="checkbox"/>	(getdate())	

```
INSERT INTO RESERVA (COSTO)
VALUES (150)
```

En el script anterior sólo se especificó el valor 150 a la columna **COSTO** de la tabla **RESERVA**. Las columnas **IDRESERVA** tienen la restricción de identidad que permite autogenerar un número consecutivo, en el caso del costo el valor por defecto es cero y en la fecha el valor por defecto es la fecha actual. Con esto determinamos que la instrucción **INSERT INTO** puede especificar más de una fila de datos y que las columnas implementadas siempre y cuando las columnas tengan una restricción de implementación como los valores por defecto o la identidad de la columna.

Para probar el registro de la reserva, coloque el siguiente script:

```
SELECT IDRESERVA, COSTO, FECHA
FROM RESERVA
```

	IDRESERVA	COSTO	FECHA
1	1	150.00	2012-08-14

Otra forma de registrar valores pre-determinados es a través del siguiente script:

```
INSERT INTO RESERVA
DEFAULT VALUES
```

	IDRESERVA	COSTO	FECHA
1	1	150.00	2012-08-14
2	2	0.00	2012-08-14

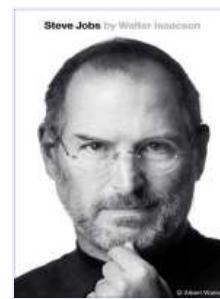
8 views

1 0

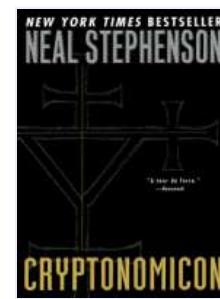
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CASO DESARROLLADO N° 3.5:

Script que permita implementar una vista y que por medio de esta registre las tarifas en la tabla de la base de datos **AGENCIA**.

TARIFA	
Column Name	Condensed Type
CLASE	varchar(20)
PRECIO	money
IMPUESTO	money

Primero, debemos crear la vista **VISTA_TARIFA** que contará con la misma estructura de la tabla. Debe ejecutar el siguiente script:

```
CREATE VIEW VISTA_TARIFA
AS SELECT CLASE,PRECIO,IMPUESTO
FROM TARIFA
GO
```

Luego insertaremos filas en la vista implementada con el siguiente script:

```
INSERT INTO VISTA_TARIFA
VALUES('TURISTA',1500,21)
GO
```

Para visualizar los registros de la tabla o la vista coloque los siguientes scripts:

```
SELECT * FROM TARIFA
SELECT * FROM VISTA_TARIFA
```

	CLASE	PRECIO	IMPUESTO
1	SUPER VIP	1200.00	12.00
2	VIP	1000.00	12.00
3	NACIONAL	800.00	12.00
4	ECONOMICO	500.00	0.00
5	TURISTA	1500.00	21.00

	CLASE	PRECIO	IMPUESTO
1	SUPER VIP	1200.00	12.00
2	VIP	1000.00	12.00
3	NACIONAL	800.00	12.00
4	ECONOMICO	500.00	0.00

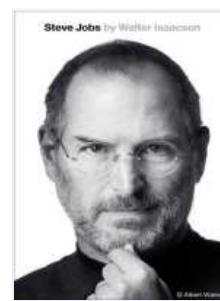
8 views

1 0

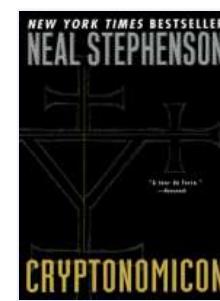
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

94

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.6:

Script que permita insertar los tres primeros pasajeros que tienen email HOTMAIL PASAJEROSHOTMAIL.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

Inicialmente los registros de la tabla **PASAJERO** deberán mostrar las siguientes filas:

	IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1	P0001	ANGELA TORRES LAZARO	0001	957564526	ATORRES@HOTMAIL.COM
2	P0002	FERNANDA TORRES LAZARO	0001	957564526	FTORRES@HOTMAIL.COM
3	P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4	P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5	P0005	LUZ LAZARO MENOR	0001	957564526	LLAZARO@GMAIL.COM
6	P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7	P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8	P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
9	P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
10	P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM

El siguiente script permite crear la tabla **PASAJEROSHOTMAIL**:

```
CREATE TABLE PASAJEROSHOTMAIL(
    IDPASAJERO CHAR(5) NOT NULL PRIMARY KEY,
    NOMBRES VARCHAR(50) NOT NULL,
    EMAIL VARCHAR(50) NOT NULL
)
GO
```

El siguiente script inserta los 3 primeros pasajeros que cuentan con email de Hotmail **PASAJEROSHOTMAIL**.

```
INSERT TOP(3) INTO PASAJEROSHOTMAIL
SELECT IDPASAJERO, NOMBRES, EMAIL
FROM PASAJERO
WHERE EMAIL LIKE '%@HOTMAIL%'
```

GO

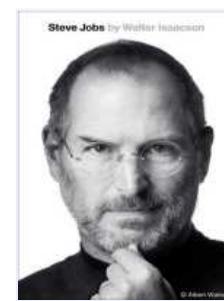
8 views

1 0

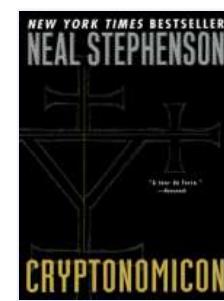
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CASO DESARROLLADO N° 3.7:

Script que permita insertar un registro a la tabla PASAJERO por intermedio de una variable de tipo TABLE.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

Para comenzar se debe declarar la variable @MISPASAJEROS de tipo TABLE e implementar la lista de columnas que participarán en la inserción, en este caso se tiene que especificar todos los campos de la tabla PASAJERO ya que a todas sus columnas se le aplicó la restricción NOT NULL.

```

DECLARE @MISPASAJEROS table( IDPASAJERO CHAR(5),
                             NOMBR  VARCHAR(50),
                             IDPAI   CHAR(4),
                             TELEF  CHAR(15),
                             CORRE  VARCHAR(50))

INSERT PASAJERO
    OUTPUT INSERTED.IDPASAJERO, INSERTED.NOMBRES,
           INSERTED.IDPAIS,INSERTED.TELEFONO,INSERTED.EMAIL
    INTO @MISPASAJEROS
    VALUES ('P0012','KAREN CASTILLO GIRALDO','0002',
           '957564526','KCASTILLO@HOTMAIL.COM');

GO
  
```

Para comprobar que el registro se ha insertado en la tabla PASAJERO debe ejecutar el siguiente script:

```
SELECT * FROM PASAJERO WHERE IDPASAJERO='P0012'
```

CASO DESARROLLADO N° 3.8:

Script que permita replicar todos los registros de la tabla PASAJERO a una nueva tabla MISPASAJEROS y que por intermedio de una tabla temporal llamada PASAJEROTEM inserte los registros usando la instrucción WITH.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

Primero, se debe crear la tabla MISPASAJEROS con las columnas necesarias para la réplica de los valores de la tabla PASAJERO.

8 views

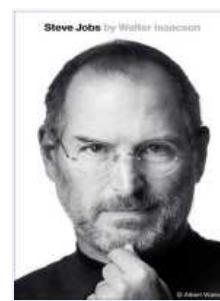
1 like

0 comments

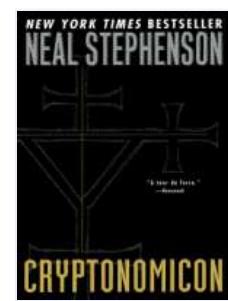
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

96

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Seguidamente, se implementará la instrucción **WITH** que permite crear una tabla temporal con los valores de la tabla **PASAJERO** que luego serán replicados a la tabla destino. Para este caso se deben crear columnas para la tabla temporal sin necesidad de especificar el tipo de dato.

```

WITH PASAJEROSTEMP(
    IDPAS,NOMBR,>IDPAI,
    FONO,CORREO)
AS (
    SELECT * FROM PASAJERO
)
INSERT INTO MISPASAJEROS
SELECT * FROM PASAJEROSTEMP
GO

```

En ambas instrucciones tanto **WITH** como **INSERT INTO** se puede controlar la forma en que se insertan los datos usando la instrucción **WHERE**, para este caso no era necesario pero lo podría implementar. No se olvide de verificar si los datos se han replicado dentro de la tabla **MISPASAJEROS** ejecutando la consulta **SELECT * FROM MISPASAJEROS**.

3.3. MODIFICACIÓN Y ACTUALIZACIÓN DE DATOS DE UNA TABLA CON UPDATE

La sentencia **UPDATE** permite modificar o actualizar un conjunto de registros de una tabla dependiendo de una condición.

Sintaxis:

LMD
UPDATE

```

UPDATE TABLA_O_VISTA
[SET] { column_name = { expression | DEFAULT | NULL }
        { udt_column_name.{ { property_name = expression
                                field_name = expression }
                                method_name ( argument [ ,...n ] )
                            }
        }
    }
[WHERE <search_condition>]

```

Donde:

- **TABLA O VISTA:** aquí se especifica el nombre de la tabla o vista que necesite ser actualizada.
- **SET:** aquí se especifica la lista de nombres de variables o de columnas que se actualizarán en la determinada tabla.

8 views

1 like

0 comments

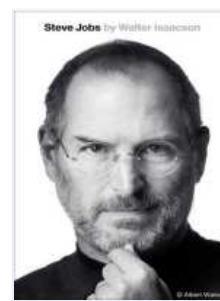
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

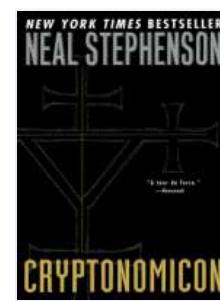
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

- -= Restar y asignar
- *= Multiplicar y asignar
- /= Dividir y asignar
- **WHERE:** especifica la o las condiciones que limitaran los valores que se actualizarán. Todo de la condición planteada, los operadores dentro de la instrucción where dependerán datos de la columna.

Consideraciones generales al actualizar registros:

- Si la sentencia de actualización infringe una restricción, una regla o si el nuevo valor es de datos incompatible al declarado en la tabla; se cancela la instrucción dando como mensaje error y no actualiza ningún registro de la tabla.
- La sentencia UPDATE es susceptible a errores aritméticos que podría ser un error de desbordamiento o división por cero durante la evaluación de la expresión, la actualización no se muestra en este caso un mensaje de error y corta la actualización desde el punto de diferencia del primer punto este si actualiza los registros hasta donde se ocasionó el error.
- Se pueden implementar la sentencia **UPDATE** dentro de funciones definidas por el usuario teniendo en cuenta que la tabla que se va a modificar sea una variable de tipo **TABLE**.
- Si en la actualización participan columnas con definición de tipo char y nchar se rellenan espacios vacíos a la derecha hasta la longitud definida en la tabla.

CASO DESARROLLADO N° 3.9:

Script que permite actualizar los valores de la columna IMPUESTO por el valor 11 a todos los registros de la tabla TARIFA.

TARIFA		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
CLASE	varchar(20)	No
PRECIO	money	No
IMPUESTO	money	No

```
UPDATE TARIFA
SET IMPUESTO=11
GO
```

El script anterior permite actualizar todos los registros de la tabla **TARIFA** cambiando el valor de IMPUESTO por 11. Esto ocurre porque dentro de la sentencia **UPDATE** no se especificó ninguna condición.

CASO DESARROLLADO N° 3.10:

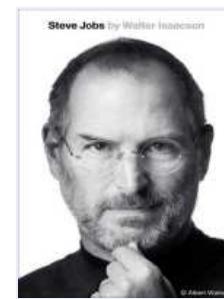
8 views

1 0

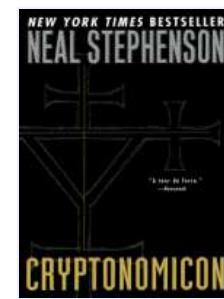
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

98

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Primero verificamos los registros de la tabla TARIFA:

	CLASE	PRECIO	IMPUESTO
1	SUPER VIP	1200.00	11.00
2	VIP	1000.00	11.00
3	NACIONAL	800.00	11.00
4	ECONOMICO	500.00	11.00

```
UPDATE TARIFA
SET IMPUESTO+=2
GO
```

En el script anterior se especifica la actualización con la instrucción **SET IMPUESTO+=2** c
podría ser implementado como **SET IMPUESTO=IMPUESTO+2**.

	CLASE	PRECIO	IMPUESTO
1	SUPER VIP	1200.00	13.00
2	VIP	1000.00	13.00
3	NACIONAL	800.00	13.00
4	ECONOMICO	500.00	13.00

Como notará la columna IMPUESTO de todos los registros se han actualizado por 13 aum
2 al valor anterior (11).

CASO DESARROLLADO N° 3.11:

Script que permita asignar el impuesto a cero sólo a los registros cuya **CLASE** sea **ECONOC**
tabla **TARIFA**.

TARIFA		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
CLASE	varchar(20)	No
PRECIO	money	No
IMPUESTO	money	No

Primero verificamos los registros de la tabla TARIFA:

	CLASE	PRECIO	IMPUESTO
1	SUPER VIP	1200.00	13.00
2	VIP	1000.00	13.00
3	NACIONAL	800.00	13.00
4	ECONOMICO	500.00	13.00

8 views

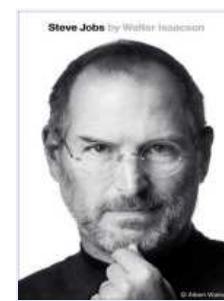
1 like

0 comments

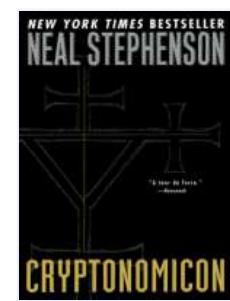
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

Hay que notar que la sentencia **WHERE** limita la actualización de los registros, haciendo sólo a los registros que cumplan con dicha condición. Una vez actualizado los registros mostrar de la siguiente forma:

	CLASE	PRECIO	IMPUESTO
1	SUPER VIP	1200.00	13.00
2	VIP	1000.00	13.00
3	NACIONAL	800.00	13.00
4	ECONOMICO	500.00	0.00

CASO DESARROLLADO N° 3.12:

Script que permita asignar el texto **SIN FONO** a los pasajeros cuyo país sea **BRASIL**, todo ser realizado en la tabla **PASAJERO**.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

Primero verificamos los registros de la tabla **TARIFA** considere que la única pasajera del paí encuentra en la fila 3 y su columna teléfono en este caso se encuentra vacía:

	IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1	P0001	ANGELA TORRES LAZARO	0001	957564526	ATORRES@HOTMAIL.COM
2	P0002	FERNANDA TORRES LAZARO	0001	957564526	FTORRES@HOTMAIL.COM
3	P0003	MARIA ZAMORA MEJIA	0005		MZAMORA@GMAIL.COM
4	P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5	P0005	LUZ LAZARO MENOR	0001	957564526	LLAZARO@GMAIL.COM
6	P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7	P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8	P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
9	P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
10	P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM
11	P0011	CONSUELO OSORIO ZEGARRA	0009	957564526	COSORIO@HOTMAIL.COM
12	P0012	KAREN CASTILLO GIRALDO	0002	957564526	KCASTILLO@HOTMAIL.COM

Mostraremos 2 script para el caso, el primero se implementará con **FROM** dentro de **UPDATE**.

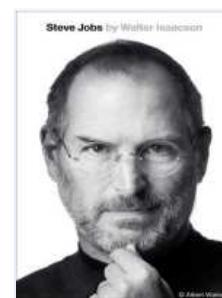
8 views

1 0

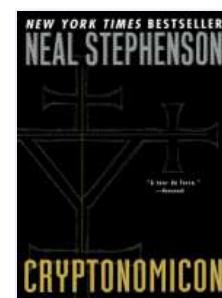
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

100

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Como se podrá visualizar no se aplicó **WHERE** sino se implementó **FROM** dentro de la sentencia principal es unir a las dos tablas involucradas, es decir, la tabla **PASAJERO** y **PAIS** para lo cual una columna en común llamada **IDPAIS** ese factor de unión se especifica en el siguiente script:

```
FROM PASAJERO AS PAS
JOIN PAIS AS PAI
ON PAS.IDPAIS=PAI.IDPAIS
```

Se le asignó un alias a ambas tablas para que no ocurra un error de ambigüedad entre las tablas ya que ambos cuentan con el mismo nombre de columna, es decir, **IDPAIS**. Entonces, para que se le asignó el alias **PAS** y en **PAIS** el alias **PAI**. Luego se tendrá que especificar el nombre de la columna que va a salir de la tabla **PASAJERO** como notará el nombre del país sólo se encuentra dentro de la tabla **PAIS**.

Para poder encontrar el país respectivo se tendrá que condicionar la columna **IDPAIS** de la tabla **PASAJERO** a que tenga el código del país a buscar; en este caso BRASIL tiene el código 0005. El siguiente muestra la forma de capturar el código sin salir de la tabla **PASAJERO**:

```
AND PAS.IDPAIS = (SELECT IDPAIS
                    FROM PAIS
                    WHERE PAIS='BRASIL')
```

Desde aquí se puede observar que **PAS.IDPAIS** está esperando el valor 0005 que será proporcionado por la consulta encerrada entre paréntesis, dicha consulta devuelve el código del país si cumple con la condición especificada en el **WHERE**. Para actualizar a otro país solamente modifique el código.

Ahora mostramos la segunda versión del mismo caso, esta vez usando la sentencia **UPDATE** con **SUBCONSULTAS**.

```
UPDATE PASAJERO
SET TELEFONO='SIN FONO'
WHERE IDPAIS = (SELECT IDPAIS
                  FROM PAIS
                  WHERE PAIS='BRASIL')
GO
```

Como notará es mucho más corto que la solución anterior pero ambas dan el mismo resultado.

CASO DESARROLLADO N° 3.13:

Script que permita actualizar el número de teléfono por **SIN FONO** a aquellos pasajeros cuyo IDPASAJERO sea **9999999999**, todo esto se realizará tomando una muestra de 3 pasajeros, el resultado se grabado dentro de una variable de tipo **TABLE** con los campos **IDPASAJERO**, **NOMBRES**

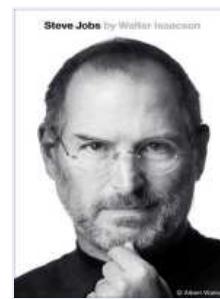
8 views

1 0

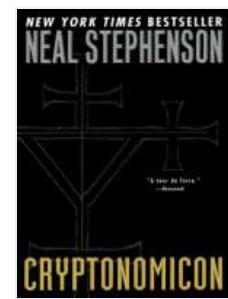
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

Primero verificamos el contenido de la tabla PASAJEROS:

IDPASAJERO	NOMBRES	DPAÍS	TELEFONO	EMAIL
P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
P0005	LUZ LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM
P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL...
P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.C...
P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM

Como notará hay 3 pasajeros que cumplen con la condición de teléfono **99999999** como si **P0002** y el **P0005**.

Ahora, implementaremos el script que permitirá actualizar las filas y grabarlos en la variable **@MISPASAJEROS**.

```

DECLARE @MISPASAJEROS TABLE(IDPAS CHAR(5),NOM VARCHAR(40),
                             TELA CHAR(15),TELN CHAR(15))
UPDATE TOP(3) PASAJERO
    SET TELEFONO='SIN FONO'
    OUTPUT INSERTED.IDPASAJERO,
        INSERTED.NOMBRES,
        DELETED.TELEFONO,
        INSERTED.TELEFONO
        INTO @MISPASAJEROS
        WHERE TELEFONO='999999999'

SELECT IDPAS,NOM,TEL,A,TELN FROM @MISPASAJEROS
GO

```

En el script anterior se preparó una variable de tipo **TABLE** llamada **@MISPASAJEROS** que campos **IDPAS** que representará el valor del **IDPASAJERO**, **NOM** que representará el pasajero, **TEL** representará al teléfono anterior a la modificación y **TELN** representará luego de la actualización.

Según el caso se tiene que especificar que la actualización solo se realizará a los tres primeros que cumplen con la condición del caso, para lograr esto se implementó **UPDATE TOP(3) PASAJERO** se asigna el valor a cambiar con **SET TELEFONO='SIN FONO'**, para poder enviar los datos del script hacia la variable de tipo **TABLE** se usa la cláusula **OUTPUT** donde se especifica los valores a enviar a la variable **@MISPASAJEROS** observe que esto se realiza especificando lo la tabla **INSERTED** con **INSERTED.IDPASAJERO**, **INSERTED.NOMBRES**, **INSERTED.TELEFONO**

8 views

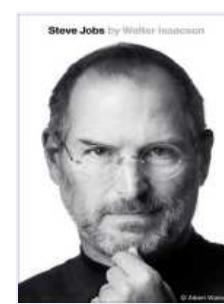
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

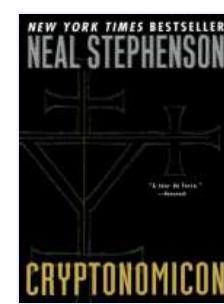
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in History

102 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Finalmente, se realiza una consulta sobre la variable de tipo **TABLE @MISPASAJEROS IDPAS,NOM,TELA,TELN FROM @MISPASAJEROS**. No se olvide que para probar el script al seleccionar desde la declaración de la variable de tipo **TABLE** hasta la consulta de la misma.

CASO DESARROLLADO N° 3.14:

Script que permite actualizar los costos de la tabla **RESERVA** disminuyendo en 50 a los registros que se realizó en el año 2011, usar el valor de actualización por medio de una variable

RESERVA					
Column Name	Condensed Type	Nullable	Identity	Default Value	
IDRESERVA	int	No	<input checked="" type="checkbox"/>		
COSTO	money	Yes	<input type="checkbox"/>	((0))	
FECHA	date	Yes	<input type="checkbox"/>	(getdate())	

Primero verificaremos cuales son los valores iniciales de la tabla **RESERVA**.

IDRESERVA	COSTO	FECHA
1	500.00	2011-10-01
2	900.00	2011-10-06
3	500.00	2011-11-14
4	1200.00	2011-11-16
5	1500.00	2011-12-12
6	590.00	2011-12-17
7	400.00	2012-01-14
8	1300.00	2012-01-15
9	1000.00	2012-02-01
10	1800.00	2012-04-02
11	1200.00	2012-04-09
12	400.00	2012-08-01
13	800.00	2012-08-15

Entonces observaremos que hay 6 registros que fueron ingresados en el año 2011, la actualización solo le debe ocurrir a dichas filas. Para eso implementaremos el siguiente script:

```
DECLARE @MONTO MONEY=50
UPDATE RESERVA
    SET COSTO-=@MONTO
    WHERE YEAR(FECHA)=2011
GO
```

Antes de actualizar se declara una variable de tipo **MONEY** e inicializada con 50 de forma **DECLARE @MONTO MONEY=50**. Seguidamente se actualiza el costo usando el comando **SET COSTO-=@MONTO** y finalmente se ejecuta el comando **GO**.

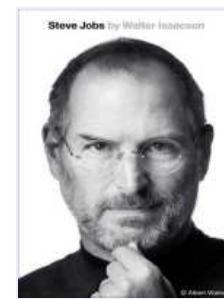
8 views

1 0

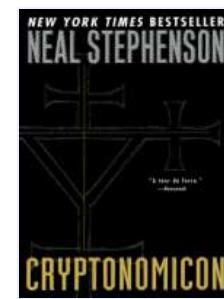
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

Ahora visualizaremos los cambios realizados a los registros de la tabla RESERVA.

IDRESERVA	COSTO	FECHA
1	450.00	2011-10-01
2	850.00	2011-10-06
3	450.00	2011-11-14
4	1150.00	2011-11-16
5	1450.00	2011-12-12
6	540.00	2011-12-17
7	400.00	2012-01-14
8	1300.00	2012-01-15
9	1000.00	2012-02-01
10	1800.00	2012-04-02
11	1200.00	2012-04-09
12	400.00	2012-08-01
13	800.00	2012-08-15

3.4. ELIMINACIÓN DE REGISTROS DE UNA TABLA CON DELETE

La sentencia **DELETE** permite eliminar todos los registros especificados en una determinada tabla.

Sintaxis:

```
LMD
DELETE

DELETE
[ TOP ( expression ) [ PERCENT ] ]
FROM TABLE
{ <object> | rowset_function_limited
[ WITH ( <table_hint_limited> [ ...n ] ) ]
}
[ <OUTPUT Clause> ]
[ FROM <table_source> [ ,...n ] ]
[ WHERE { <search_condition>
| { [ CURRENT OF
{ { [ GLOBAL ] cursor_name }
| cursor_variable_name
}
]
}
}
]
```

Donde:

- **TOP:** especifica una muestra en número o porcentaje de registros posibles a eliminar.
- **TABLA O VISTA:** aquí se especifica el nombre de la tabla o vista que necesite ser actualizada.

8 views

1 like

0 comments

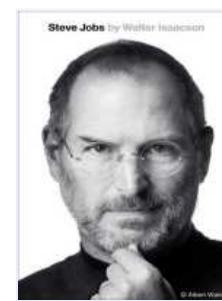
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

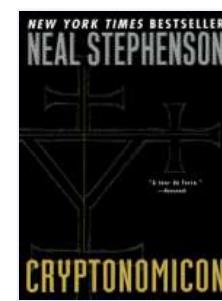
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

104 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

- Cuando la sentencia **DELETE** determina un error aritmético en su proceso el motor de datos cancela todo el proceso y envía un mensaje de error al usuario ya que activa la **ARITHABORT ON**.
- También puede optar por la sentencia **TRUNCATE TABLE** cuando necesite eliminar todos los datos de una determinada tabla sin especificar la cláusula WHERE. En este caso TRUNCATE usa menos recursos que DELETE, por tanto es mucho más rápido la transacción.

CASO DESARROLLADO N° 3.15:

Script que permite eliminar todos los registros de la tabla **AEROLINEA**.



```
DELETE AEROLINEA
GO
```

El script anterior permite eliminar todos los registros de la tabla AEROLINEA, también se puede utilizar el siguiente script **DELETE FROM AEROLINEA**.

CASO DESARROLLADO N° 3.16:

Script que permite eliminar el registro de la tabla **PASAJERO** cuyo **IDPASAJERO** sea **P0010**.

Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

```
DELETE FROM PASAJERO
WHERE IDPASAJERO='P0010'
GO
```

En el script anterior se hace referencia a la tabla **PASAJERO** condicionando al pasajero **P0010**. También se pudo especificar de la siguiente forma:

8 views

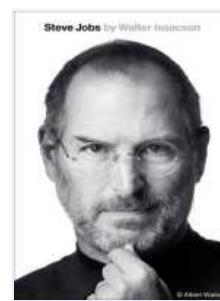
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

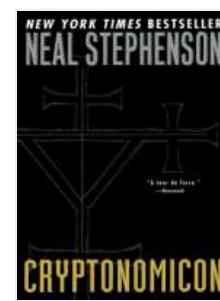
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

CASO DESARROLLADO N° 3.17:

Script que permita eliminar el registro de la tabla PASAJERO cuyo país sea ECUADOR use subconsultas para el proceso.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

```

DELETE FROM PASAJERO
WHERE IDPAIS=
    (
        SELECT IDPAIS
        FROM PAIS
        WHERE PAIS='ECUADOR'
    )
GO

```

En el script anterior note que la sentencia **DELETE** condiciona al **IDPAIS** según el resultado de la subconsulta, hay que tener en cuenta que dicha subconsulta sólo debe enviar un valor, si se implementa de forma errada emitirá más de un registro; el motor de base de datos retorna un mensaje de error al usuario anulando el proceso de eliminación.

CASO DESARROLLADO N° 3.18:

Script que permita eliminar el 5% de los registros de la tabla **PASAJERO** cuyo país sea ECUADOR use subconsultas para el proceso.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

```

DELETE TOP(5) PERCENT
FROM PASAJERO
WHERE IDPAIS=
    (
        SELECT IDPAIS
        FROM PAIS
    )

```

8 views

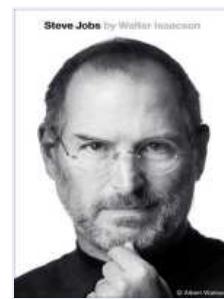
1 like

0 comments

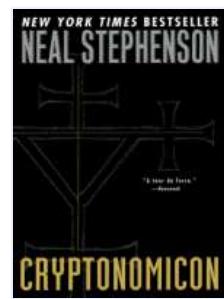
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

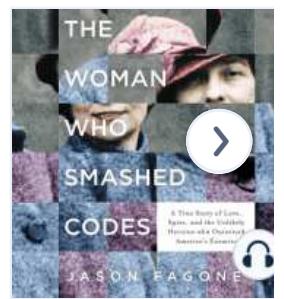
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

106 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.19:

Script que permita eliminar el 5% de los registros de la tabla **PASAJERO** cuyo país sea subconsultas para el proceso. Use el operador **OUTPUT**.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

```

DECLARE @MISPASAJEROS TABLE (
    IDPAS CHAR(5),
    NOMBR VARCHAR(40),
    TELEF CHAR(15)
)

DELETE PASAJERO
    OUTPUT deleted.IDPASAJERO,
           deleted.NOMBRES,
           deleted.TELEFONO
        INTO @MISPASAJEROS
    FROM PASAJERO P
    WHERE IDPAIS=(

        SELECT IDPAIS
        FROM PAIS
        WHERE PAIS='BRASIL')

SELECT IDPAS, NOMBR, TELEF FROM @MISPASAJEROS
GO

```

En el script anterior se declaró **@MISPASAJEROS** como la variable de tipo **TABLE** que permite temporalmente los registros de los pasajeros eliminados desde la sentencia **DELETE**, como usamos la cláusula **OUTPUT** donde se dirige los valores eliminados a la variable tipo **MISPASAJEROS**, la cláusula **FROM** especifica los registros a eliminar aquí se está condicionando dichos pasajeros sean del país **BRASIL**.

Finalmente, se muestran los valores registrados en la variable **@MISPASAJEROS**, recuerda ejecutar el script se tiene que seleccionar desde la declaración de la variable **TABLE** hasta la **SELECT** que muestra el contenido de dicha variable.

8 views

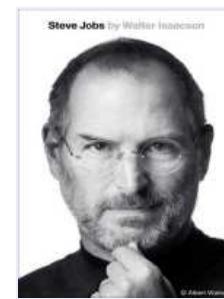
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

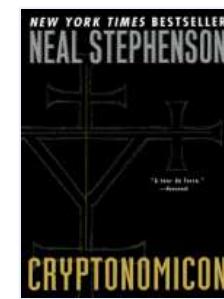
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Unlikely Heroine Who Outwitted America's Smartest Codebreakers

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

3.5. DECLARACIÓN GENERAL DEL COMANDO SELECT PARA LA RECUPERACIÓN DE REGISTROS

Uno de los propósitos de la gestión de base de datos es almacenar información lógica dentro de tablas y obtener información de la misma, para lograr este propósito usaremos SELECT y sus variadas formas de recuperar información desde la tabla de una base de datos.

Sintaxis:

LMD
SELECT
<pre> SELECT [ALL DISTINCT] [TOP numero [PERCENT]] < select_list > [INTO nueva_tabla] [FROM { <tabla_fuente> } [,...n]] [WHERE <condicion>] [<GROUP BY>] [HAVING < condicion >] [ORDER BY columna [ASC DESC]] </pre>

Donde:

- **ALL:** especifica que el conjunto de filas devueltas por la consulta puede incluir filas dígitos iguales a usar el operador * en la consulta.
- **DISTINCT:** especifica que el conjunto de filas devueltas por la consulta no sean repetidas, estas filas deben ser únicas. Los valores asignados con **NULL** se consideran igual desde la vista de la cláusula **DISTINCT**.
- **TOP:** especifica que el conjunto de filas devueltas por la consulta puede ser controlado en cantidad o en porcentaje. Considere que la muestra de registros devueltas siempre será el primer número de filas especificadas. Para especificar el porcentaje sólo agregue la palabra **PERCENT**.
- **SELECT_LIST:** especifica que columnas participarán en el conjunto de resultados o la sentencia SELECT. Para poder especificar varias columnas se debe separar por comas, el número máximo de columnas es de 4096.
- **INTO:** Permite crear un nueva tabla y envíe las filas resultantes del proceso de consulta.
- **FROM:** permite especificar las tablas que están involucradas en la consulta, hay que tener en cuenta que aquí es donde se definen los alias a las tablas.
- **WHERE:** esta cláusula permite condicionar el resultado de una consulta.
- **GROUP BY:** permite agrupar un conjunto de registros en forma de resumen con especificados de una o más columnas.
- **ORDER BY:** permite ordenar los registros de acuerdo a una columna específica, no tiene que haber una cláusula GROUP BY.

8 views

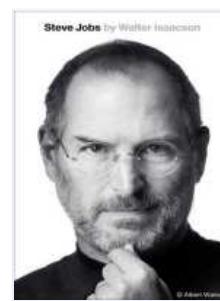
1 like

0 comments

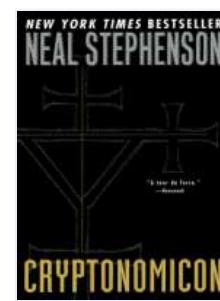
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

108 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Consideraciones generales de la sentencia SELECT:

- La sintaxis completa de la instrucción **SELECT** es un poco compleja, pero podemos considerar las cláusulas principales a los siguientes:
 - **WITH <common_table_expression>**
 - **SELECT select_list [INTO new_table]**
 - **FROM table_source] [WHERE search_condition]**
 - **GROUP BY group_by_expression**
 - **HAVING search_condition**
 - **ORDER BY order_expression [ASC | DESC]**
- Los operadores **UNION**, **EXCEPT** e **INTERSECT** se pueden utilizar entre consultas para comparar valores en un conjunto de resultados.
- El operador ***** es usado como una atajo para designar las columnas a mostrar en la consulta.
- En el caso de combinaciones de tabla dentro de una misma consulta deberá diferenciar a cada tabla por medio de un alias asignada a cada tabla. Desde ese momento podrá especificar los campos de cada tabla por medio del alias asignada a la misma.

CASO DESARROLLADO N° 3.20:

Script que permita mostrar los registros de la tabla PASAJERO que use todos los predicados para el mismo proceso.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

Primera forma:

```
SELECT * FROM PASAJERO
GO
```

Segunda forma:

```
SELECT ALL * FROM PASAJERO
GO
```

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
------------	---------	--------	----------	-------

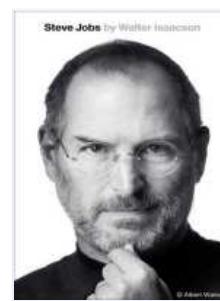
8 views

1 0

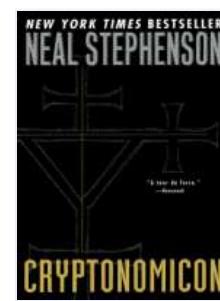
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

En el script anterior se usó el operador * para especificar todas las columnas y todas las filas **PASAJERO**. En la segunda solución el predicado **ALL** especifica también todas las columnas mientras que * representa a todas las filas de la tabla **PASAJERO**.

Ahora, realizaremos el mismo proceso pero especificando las columnas de la tabla **PASAJERO**:

```
SELECT IDPASAJERO, NOMBRES, IDPAIS, TELEFONO, EMAIL
FROM PASAJERO
GO
```

En el script anterior el resultado es el mismo que cuando se usó el operador * y el predicho pero podríamos aprovechar en especificar un orden a las columnas. Entonces aplicaremos la consulta pero en un orden distinto al original. Veamos el siguiente script:

```
SELECT NOMBRES, EMAIL, TELEFONO
FROM PASAJERO
GO
```

Como notará no es necesario especificar todos las columnas que contiene la tabla **PASAJERO**, lo que lo hace más específico el usuario. Aquí listamos el resultado:

	NOMBRES	EMAIL	TELEFONO
1	ANGELA TORRES LAZARO	ATDRRES@HOTMAIL.COM	9999999999
2	FERNANDA TORRES LAZARO	FTORRES@HOTMAIL.COM	9999999999
3	MARIA ZAMORA MEJIA	MZAMORA@GMAIL.COM	957564526
4	GUADALUPE ACOSTA FERRER	GACOSTA@HOTMAIL.COM	957564526
5	LUZ LAZARO MENDOZA	LLAZARO@GMAIL.COM	9999999999
6	KARLA GALLEGOS SILVA	KGALLEGOS@HOTMAIL.COM	957564526
7	NERY CALLE DE LA CRUZ	NCALLE@GMAIL.COM	957564526
8	HEIDI RENGIFO REATEGUI	HRENGIFO@HOTMAIL.COM	957564526
9	MARISOL DIAZ ZAMBRANO	MDIAZ@GMAIL.COM	957564526
10	LINDA TUME VARAS	LTUME@HOTMAIL.COM	957564526

Ahora podemos implementar la misma consulta definiendo una cabecera personalizada por el usuario. A continuación se muestra la cabecera de una consulta simple de la tabla **PASAJERO**.

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
------------	---------	--------	----------	-------

Veamos la primera implementación de este proceso:

```
SELECT IDPASAJERO AS 'CÓDIGO',
NOMBRES AS 'PASAJERO'.
```

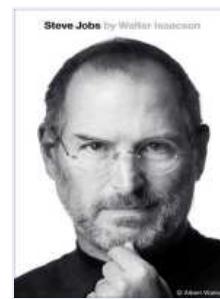
8 views

1 0

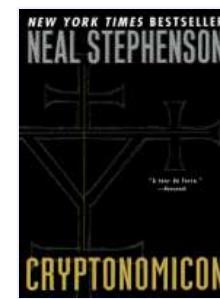
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

110 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Como notará el uso de la palabra **AS** define la cabecera de cada columna como **IDPAIS**, **'CODIGO'** aquí la columna **IDPASAJERO** tiene como cabecera o título **CODIGO**. Ahora vis la columna **TELEFONO** el cual no se especificó la cabecera, por tanto al no definirlo la cabe como título el mismo nombre de la columna.

CODIGO	PASAJERO	CODIGO PAIS	TELEFOND	CORREO ELECTRONICO
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2 P0002	FERNANDA TORRES LAZARO	0001	989999999	FTORRES@HOTMAIL.COM
3 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4 P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5 P0005	LUZ LAZARO MENDR	0001	999999999	LLAZARO@GMAIL.COM
6 P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7 P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8 P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
9 P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
10 P0010	LINDA TUNE VARAS	0008	957564526	LTUME@HOTMAIL.COM

La segunda implementación sería de la siguiente manera:

```
SELECT IDPASAJERO AS [CODIGO],
       NOMBRES AS [PASAJERO],
       IDPAIS AS [CODIGO PAIS],
       TELEFONO,
       EMAIL AS [CORREO ELECTRONICO]
  FROM PASAJERO
     GO
```

Como notará en el script anterior sólo hemos reemplazado las comillas simples por corchet podríamos suprimir la palabra **AS** en la asignación de la cabecera.

```
SELECT IDPASAJERO [CODIGO],
       NOMBRES [PASAJERO],
       IDPAIS [CODIGO PAIS],
       TELEFONO,
       EMAIL [CORREO ELECTRONICO]
  FROM PASAJERO
     GO
```

En la tercera implementación se asigna la cabecera con la columna de la tabla de la siguie

```
SELECT CODIGO=IDPASAJERO,
       PASAJERO=NOMBRES,
       [CODIGO PAIS]=IDPAIS,
       TELEFONO,
       [CORREO ELECTRONICO]=EMAIL
  FROM PASAJERO
     GO
```

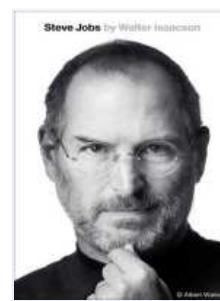
8 views

1 0

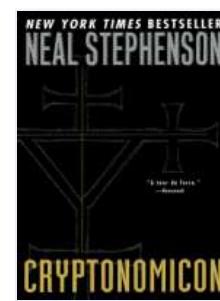
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

Finalmente, podríamos listar los registros de la tabla PASAJERO usando un alias a la tabla de forma:

```
SELECT P.IDPASAJERO, P.NOMBRES,
       P.IDPAIS,P.TELEFONO,P.EMAIL
  FROM PASAJERO P
     GO
```

En el script anterior se asigna el alias P a la tabla PASAJERO con el objetivo de definir un de las columnas, por tanto P.IDPASAJERO significa que la columna IDPASAJERO pertenece con alias P. Desde la implementación básica de la consulta de la tabla PASAJERO podríamos siguiente script:

```
SELECT PASAJERO.IDPASAJERO, PASAJERO.NOMBRES,
       PASAJERO.IDPAIS,PASAJERO.TELEFONO,PASAJERO.EMAIL
  FROM PASAJERO
     GO
```

Pero podríamos usar un alias y el operador * al mismo tiempo de la siguiente forma:

```
SELECT P.*
  FROM PASAJERO P
     GO
```

CASO DESARROLLADO N° 3.21:

Script que permita mostrar los códigos registrados en la tabla PASAJERO sin repetirlos, us **DISTINCT**.

PASAJERO		
	Nombre de columna	Tipo comprimido
		Aceptación de valores NULL
1	IDPASAJERO	char(5)
	NOMBRES	varchar(50)
	IDPAIS	char(4)
	TELEFONO	char(15)
	EMAIL	varchar(50)

Antes de implementar el script del proceso visualizaremos los códigos registrados sin rest la tabla **PASAJERO**.

IDPAIS
1
2
3

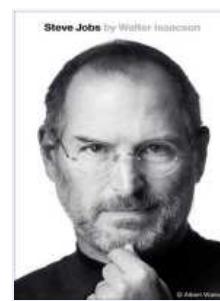
8 views

1 0

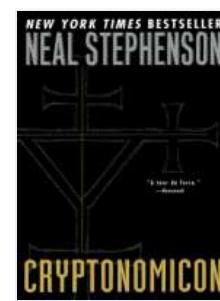
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

112 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Como notará en la imagen anterior los código 0001 se repite tres veces, así como el código 0002 se repite dos veces. Ahora mostraremos el script que omite los códigos duplicados haciendo uso de la cláusula **DISTINCT**.

```
SELECT DISTINCT IDPAIS
  FROM PASAJERO
 GO
```

En el script anterior sólo se especifica la columna **IDPAIS** ya que la distinción sólo puede ser realizada sobre una columna de la tabla. El resultado de este script sería como se muestra a continuación:

IDPAIS
0001
0002
0004
0005
0007
0008
0010

CASO DESARROLLADO N° 3.22:

Script que permita mostrar los registros de la tabla **PASAJERO** ordenados por sus nombres de forma ascendente use la cláusula **ORDER BY**.

PASAJERO			
	Nombre de columna	Tipo comprimido	Aceptación de valores NULL
1	IDPASAJERO	char(5)	No
2	NOMBRES	varchar(50)	No
3	IDPAIS	char(4)	No
4	TELEFONO	char(15)	No
5	EMAIL	varchar(50)	No

El script para lograr este proceso se define de la siguiente manera:

```
SELECT P./*
   FROM PASAJERO P
 ORDER BY P.NOMBRES
 GO
```

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3 P0004	GUADALUPE ACOSTA FRRFR	0002	957564526	GACOSTA@HOTMAIL.COM

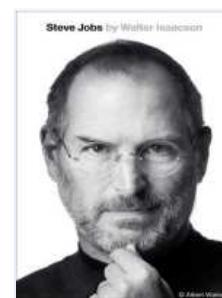
8 views

1 0

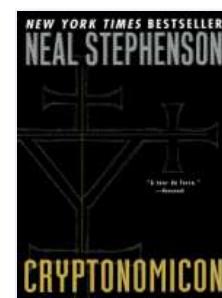
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

Como habrá notado en el script no se especificó el orden ascendente dentro de la sentencia `ORDER BY`, al no especificarlo siempre se ordenará en forma ascendente, también se podría especificar el orden de la siguiente manera:

```
SELECT P.*  
FROM PASAJERO P  
ORDER BY P.NOMBRES ASC  
GO
```

CASO DESARROLLADO N° 3.23:

Script que permita mostrar los registros de la tabla **PASAJERO** ordenados por su **IDPA** ascendente y a la duplicidad de filas ordenarlos por **IDPASAJERO** en forma descendente. Usar la cláusula **ORDER BY**.

PASAJERO			
	Nombre de columna	Tipo comprimido	Aceptación de valores NULL
1	IDPASAJERO	char(5)	No
	NOMBRES	varchar(50)	No
2	IDPAIS	char(4)	No
	TELEFONO	char(15)	No
3	EMAIL	varchar(50)	No

En el siguiente script listaremos los registros ordenados por el **IDPAIS** con las siguientes columnas: **IDPAIS, IDPASAJERO, NOMBRES, TELEFONO, EMAIL**. Implemente el siguiente código:

```
SELECT P.IDPAIS, IDPASAJERO, P.NOMBRES, P.TELEFONO, P.EMAIL  
FROM PASAJERO P  
ORDER BY P.IDPAIS  
GO
```

IDPAIS	IDPASAJERO	NOMBRES	TELEFONO	EMAIL
1	P0001	ANGELA TORRES LAZARO	999999999	ATORRES@HOTMAIL.COM
2	P0001	FERNANDA TORRES LAZARO	999999999	FTORRES@HOTMAIL.COM
3	P0005	LUZ LAZARO MENOR	999999999	LLAZARO@GMAIL.COM
4	P0004	GUADALUPE ACOSTA FERRER	957564526	GACOSTA@HOTMAIL.COM
5	P0008	HEIDI RENGIFO REATEGUI	957564526	HRENGIFO@HOTMAIL.COM
6	P0009	MARISOL DIAZ ZAMBRANO	957564526	MDIAZ@GMAIL.COM
7	P0003	MARIA ZAMORA MEJIA	957564526	MZAMORA@GMAIL.COM
8	P0006	KARLA GALLEGOS SILVA	957564526	KGALLEGOS@HOTMAIL.COM
9	P0010	LINDA TUÑE VARAS	957564526	LTUNE@HOTMAIL.COM
10	P0007	NERY CALLE DE LACRUZ	957564526	NCALLE@GMAIL.COM

Pero notará que los códigos del **IDPASAJERO** no se encuentran ordenados en forma descendente.

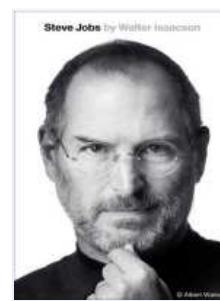
8 views

1 0

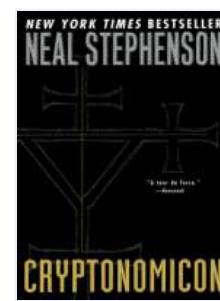
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

114

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

IDPAIS	IDPASAJERO	NOMBRES	TELEFONO	EMAIL
1	P0005	LUZ LAZARO MENOR	999999999	LLAZARO@GMAIL.COM
2	P0002	FERNANDA TORRES LAZARO	999999999	FTORRES@HOTMAIL.COM
3	P0001	ANGELA TORRES LAZARO	999999999	ATORRES@HOTMAIL.COM
4	P0004	GUADALUPE ACOSTA FERRER	957564526	GACOSTA@HOTMAIL.COM
5	P0009	MARISOL DIAZ ZAMBRANO	957564526	MDIAZ@GMAIL.COM
6	P0008	HEIDI RENGIFO REATEGUI	957564526	HRENGIFO@HOTMAIL.COM
7	P0003	MARIA ZAMORA MEJIA	957564526	MZAMORA@GMAIL.COM
8	P0006	KARLA GALLEGOS SILVA	957564526	KGALLEGOS@HOTMAIL.COM
9	P0010	LINDA TUME VARAS	957564526	LTUME@HOTMAIL.COM
10	P0007	NERY CALLE DE LA CRUZ	957564526	NCALLE@GMAIL.COM

Observe que la columna **IDPASAJERO** se encuentra ordenada en forma descendente dependiendo de la columna **IDPAIS**. También se pudo ordenar por los nombre de los pasajeros de la siguiente manera:

```
SELECT P.IDPAIS,P.NOMBRES,P.IDPASAJERO,P.TELEFONO,P.EMAIL
FROM PASAJERO P
ORDER BY P.IDPAIS,P.NOMBRES DESC
GO
```

IDPAIS	NOMBRES	IDPASAJERO	TELEFONO	EMAIL
1	LUZ LAZARO MENOR	P0005	999999999	LLAZARO@GMAIL.COM
2	FERNANDA TORRES LAZARO	P0002	999999999	FTORRES@HOTMAIL.COM
3	ANGELA TORRES LAZARO	P0001	999999999	ATORRES@HOTMAIL.COM
4	GUADALUPE ACOSTA FERRER	P0004	957564526	GACOSTA@HOTMAIL.COM
5	MARISOL DIAZ ZAMBRANO	P0009	957564526	MDIAZ@GMAIL.COM
6	HEIDI RENGIFO REATEGUI	P0008	957564526	HRENGIFO@HOTMAIL.COM
7	MARIA ZAMORA MEJIA	P0003	957564526	MZAMORA@GMAIL.COM
8	KARLA GALLEGOS SILVA	P0006	957564526	KGALLEGOS@HOTMAIL.COM
9	LINDA TUME VARAS	P0010	957564526	LTUME@HOTMAIL.COM
10	NERY CALLE DE LA CRUZ	P0007	957564526	NCALLE@GMAIL.COM

CASO DESARROLLADO N° 3.24:

Script que permita mostrar los 5 primeros registros de la tabla PASAJERO, use la cláusula `TOP`:

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

8 views

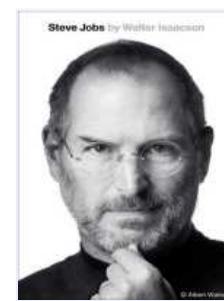
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

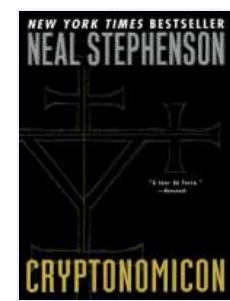
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4 P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5 P0005	LUIZ LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM

De todos los registros almacenados en la tabla solo se muestra una porción de registros e sólo los cinco primeros registros.

CASO DESARROLLADO N° 3.25:

Script que permite mostrar los 5 últimos registros de la tabla PASAJERO, use la cláusula **TOP BY** asociados.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

```
SELECT TOP 5 *
FROM PASAJERO
ORDER BY IDPASAJERO DESC
GO
```

Para mostrar los 5 últimos registros debemos contar con una columna que defina el criterio de ordenamiento ya que si tomamos como criterio de orden a la columna **NOMBRES** sólo podríamos ordenarlos en forma ascendente y descendente por su letra inicial, pero no indicaría que fue el primero o el último registro; por tanto la única columna podría ser **IDPASAJERO** que tiene una serie de registros numerados donde el 0010 indica que dicho registro es el décimo en la tabla **PASAJERO**. A continuación se muestra el resultado del script.

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0010	LINDA TUME VARAS	0003	957564526	LTUME@HOTMAIL.COM
2 P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
3 P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
4 P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
5 P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM

Ahora podríamos implementar el mismo resultado con otro script de la siguiente forma:

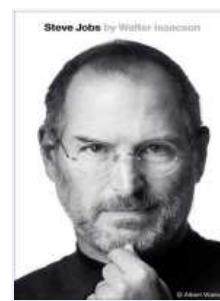
8 views

1 0

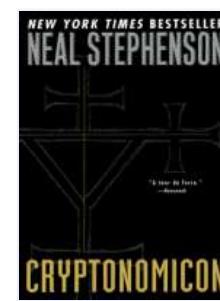
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

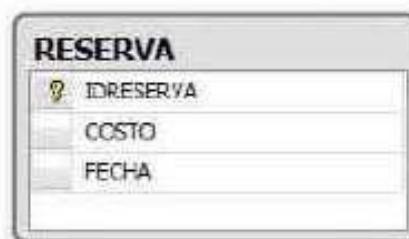
116 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En la imagen siguiente se muestra el resultado del script anterior donde notará que hay un número llamada **NÚMERO** donde muestra el número de fila asignada a los valores de **ROW_NUMBER** es una función que permite devolver el número secuencial de filas de un resultado comenzando siempre en 1 al resultar la consulta. La cláusula **OVER** define el contexto de los registros en la consulta.

NUMERO	IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1	P0010	LINDA LTUME VARAS	0008	957564526	LTUME@HOTMAIL.COM
2	P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
3	P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
4	P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
5	P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM

CASO DESARROLLADO N° 3.26:

Script que permita mostrar el 30% de los primeros registros de la tabla RESERVA, use la cláusula **PERCENT**.



Inicialmente los registros almacenados en la tabla RESERVA son:

IDRESERVA	COSTO	FECHA
1	450.00	2011-10-01
2	850.00	2011-10-06
3	450.00	2011-11-14
4	1150.00	2011-11-16
5	1450.00	2011-12-12
6	540.00	2011-12-17
7	400.00	2012-01-14
8	1300.00	2012-01-15
9	1000.00	2012-02-01
10	1800.00	2012-04-02
11	1200.00	2012-04-09
12	400.00	2012-08-01
13	800.00	2012-08-15

Si deseamos mostrar solo el 30% de los registros, entonces el script sería como sigue:

```
SELECT TOP 30 PERCENT *
FROM RESERVA
GO
```

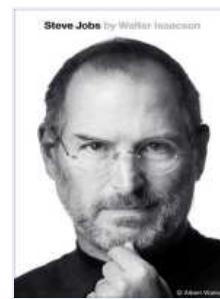
8 views

1 0

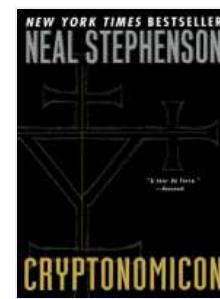
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

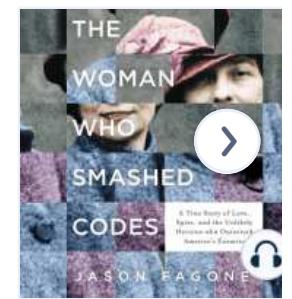
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

CASO DESARROLLADO Nº 3.27:

Script que permite replicar los registros de la tabla PASAJERO en una nueva tabla llamada MIS. Use la cláusula INTO.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

Hay que tener en cuenta que la cláusula **INTO** permite crear la tabla y además definir si en base a las columnas especificadas en la consulta. A continuación se muestra el script para realizar el proceso de réplica.

```

IF OBJECT_ID('MISPASAJEROS') IS NOT NULL
BEGIN
    DROP TABLE MISPASAJEROS
END
GO

SELECT P.*
    INTO MISPASAJEROS
    FROM PASAJERO P
GO

SELECT * FROM MISPASAJEROS
GO

```

En el primer bloque se tiene que verificar que la tabla no existe para poder continuar con el proceso de creación y asignación de valores, caso contrario se tendrá que eliminar la tabla. En este caso se usa la función **OBJECT_ID** para verificar si existe la tabla **MISPASAJEROS** al asignarle **IS NOT NULL**, indicando que la tabla si existe, dentro de la condicional se especifica la eliminación de la tabla con la sentencia **DROP TABLE**.

En el segundo bloque se hace referencia a todas las columnas de la tabla de origen con **P.*** y el alias de la tabla **PASAJERO**, luego se especifica la nueva tabla con **INTO MISPASAJEROS**. En el primer bloque se verificó que la tabla no exista, entonces esta línea no ocasionará error. Finalmente se especifica de donde provienen los datos de origen con **FROM PASAJERO P**.

En el último bloque se hace la consulta en referencia a la nueva tabla creada por consecuencia de la ejecución de la consulta del segundo bloque.

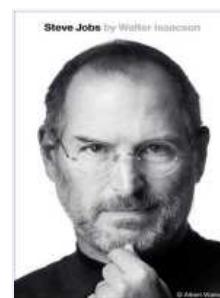
8 views

1 0

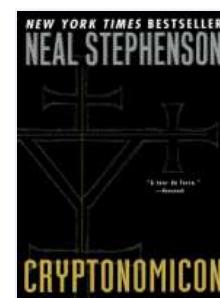
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

118 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

```

IF OBJECT_ID('MISPASAJEROS') IS NOT NULL
BEGIN
    DROP TABLE MISPASAJEROS
END
GO

SELECT P.*
INTO MISPASAJEROS
FROM PASAJERO P
WHERE P.EMAIL LIKE '%HOTMAIL%'
GO

SELECT * FROM MISPASAJEROS
GO

```

Como notará en el segundo bloque se hace referencia a la condición de sólo email de HOTMAIL en la cláusula **WHERE** que explicaremos en los siguientes casos.

En ambos casos para ejecutar las sentencias tiene que seleccionar los tres bloques desde la creación de la existencia de la tabla hasta el listado de los registros con **SELECT * FROM MISPASAJEROS**.

CASO DESARROLLADO N° 3.28:

Script que permite mostrar los registros de las tablas PASAJERO y PAIS combinados.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

PAIS	
IDPAIS	PAIS

Hay que tener en cuenta que los registros de la tabla PASAJERO y PAIS tienen una relación de uno a muchos gracias a la columna IDPAIS, por tanto mostrar los registros de ambos al mismo tiempo no es lograr, pero con el defecto de mostrar la multiplicación de sus registros, es decir, para este caso tendría 130 registros, 13 países por 10 pasajeros registrados.

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL	IDPAIS	PAIS
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	0001	PERU
2 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	0002	ARGENTINA
3 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	0003	CHILE
4 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	0004	ECUADOR
5 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	0005	BRASIL
6 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	0006	VENEZUELA

Consulta ejecutada correctamente

MTORES-PC (10.0 RTM) Mtorres-PC/Mtorres (56) AGENCIA 00:00:00

8 views

1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)

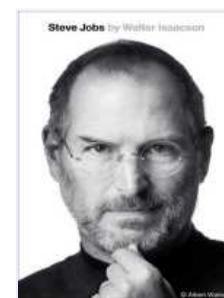
Save

Embed

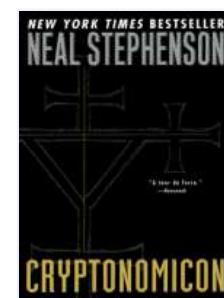
Share

Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, andrea

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

En el script anterior se asignó un alias a cada tabla ya que al tener columnas ambiguas genera error en la consulta.

Para no mostrar filas replicadas en la consulta anterior se tiene que implementar el siguiente script:

```
SELECT PAS.* ,PAI.*  
FROM PASAJERO PAS,PAIS PAI  
WHERE PAS.IDPAIS=PAI.IDPAIS  
GO
```

En vista que ambas tablas cuentan con la columna IDPAIS y es lo único que los asocia, se condiciona dicha unión para no repetir las filas por eso se añadió la siguiente línea de código anterior **WHERE PAS.IDPAIS=PAI.IDPAIS**. A continuación se muestran los registros y con una cantidad de filas en este caso es 10 y no 130 como se mostró en la imagen anterior.

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL	IDPAIS	PAIS
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	0001	PERU
2 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM	0001	PERU
3 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM	0005	BRASIL
4 P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM	0002	ARGENTINA
5 P0005	LUZ LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM	0001	PERU
6 P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM	0007	PARAGUAY
7 P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM	0010	MEXICO
8 P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM	0004	ECUADOR
9 P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM	0004	ECUADOR
10 P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM	0008	URUGUAY

Consulta ejecutada correctamente

MTORRES-PC (10.0 RTM)

Mtorres-PC\mtorres (56)

AGENCIA

00:00:00

Si cambiamos el orden de las columnas entonces se vería de la siguiente manera:

```
SELECT PAI.* ,PAS.*  
FROM PASAJERO PAS,PAIS PAI  
WHERE PAS.IDPAIS=PAI.IDPAIS  
GO
```

IDPAIS	PAIS	IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 0001	PERU	P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2 0001	PERU	P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3 0005	BRASIL	P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4 0002	ARGENTINA	P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5 0001	PERU	P0005	LUZ LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM
6 0007	PARAGUAY	P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7 0010	MEXICO	P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8 0004	ECUADOR	P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM

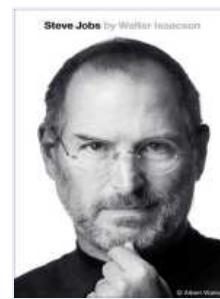
8 views

1 0

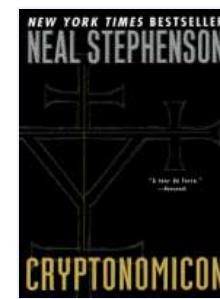
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

120 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.29:

Script que permite mostrar los registros de los pasajeros cuyo país registrado sea ECU subconsultas en la cláusula FROM.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

PAÍS	
IDPAIS	PAÍS
0004	ECUADOR

En la primera implementación se especifica el código del país y no el nombre del mismo:

```
SELECT *
FROM (SELECT P. *
      FROM PASAJERO P
     WHERE P.IDPAIS='0004') X
GO
```

En el script anterior la subconsulta se implementa dentro del FROM especificando todos los campos de la tabla PASAJERO representada con el alias P y condicionando el código del país al final se especifica el nombre de X ya que la sintaxis de la sentencia SELECT exige el nombre de una tabla. Por tanto se accede a todos los campos referenciados de la subconsulta. El resultado se muestra en la siguiente imagen:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
P0008	HEIDI RENEGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM

En la segunda implementación se especifican los campos IDPASAJERO, NOMBRES Y EMAIL desde el alias de la subconsulta.

```
SELECT X.IDPASAJERO,X.NOMBRES,X.EMAIL
FROM (SELECT P. *
      FROM PASAJERO P
     WHERE P.IDPAIS='0004') X
GO
```

En el script anterior se especifican las columnas desde el alias de la tabla X con la siguiente consulta. Se especifica X.IDPASAJERO recordando que X almacena los registros de la subconsulta. La siguiente imagen muestra el resultado de la consulta:

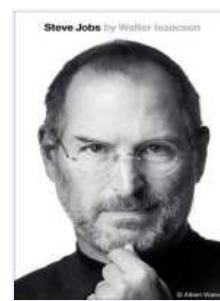
8 views

1 0

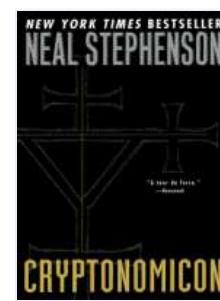
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

```

SELECT X.IDPASAJERO,X.NOMBRES,X.EMAIL
      FROM (SELECT P.* 
            FROM PASAJERO P
           WHERE P.IDPAIS=(SELECT PA.IDPAIS
                            FROM PAIS PA
                           WHERE PA.PAIS='ECUADOR' ))
GO
  
```

Observe que en la subconsulta inicial se condiciona el IDPAIS, dentro de el también se especifica una subconsulta donde se especifica el nombre del país desde la tabla PAIS. A continuación se muestra el resultado de la ejecución de la consulta.

IDPASAJERO	NOMBRES	EMAIL
P0008	HEIDI RENGIPO REATEGUI	HRENGIPO@HOTMAIL.COM
P0009	MARISOL DIAZ ZAMBRANO	MDIAZ@GMAIL.COM

3.6. Los OPERADORES EN SQL SERVER 2012

Un operador es un símbolo que especifica una acción que es realizada por una o más expresiones. En las siguientes tablas se incluyen las categorías de operadores que utiliza SQL Server.

A. ARITMÉTICOS

Estos operadores permiten realizar operaciones matemáticas entre expresiones del tipo real o necesariamente iguales.

+ sumar	Operador de suma numérica, concatenación de columnas en una consulta y para los tipos caracteres.
- Restar	Operador de restas numéricas también representa a los números negativos.
* Multiplicar	Operador de multiplicación.
/ Dividir	Operador de división entera y fraccionaria.
% Modulo	Operador que devuelve el resto de una división.

CASO DESARROLLADO N° 3.30:

Script que permite mostrar el resultado de la expresión $2+(4+2)$ use la sentencia SELECT y

Solución con SELECT:

```
SELECT 2+(4+2) AS [EXPRESIÓN]
```

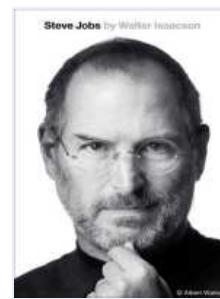
8 views

1 0

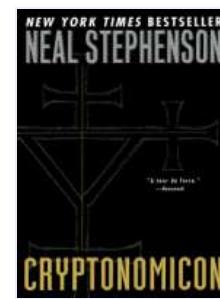
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

122 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

La imagen siguiente muestra el resultado de la consulta. Recuerde que en toda expresión las paréntesis tienen más prioridad que los demás operadores.

EXPRESIÓN	RESULTADO
1 2+4*2	8

Solución con **PRINT**:

```
PRINT 'LA EXPRESION 2+(4+2) TIENE COMO RESULTADO' +STR(2+(4+2))
GO
```

En el script se observa que la forma de implementar la solución es distinta por el hecho de que la sentencia **PRINT** tiene otra sintaxis. Aquí el operador **+** se utilizó para unir 2 expresiones la primera es el texto **LA EXPRESION 2+(4+2) TIENE COMO RESULTADO** y el segundo es el resultado de la expresión **STR(2+(4+2))**, como notará aquí se está aplicando la función **STR** que permite convertir un número a cadena (String) hay que recalcar que sino se convierte a cadena el motor de base de datos mostraría el siguiente mensaje de error:

```
Mens. 245, Nivel 16, Estado 1, Línea 1
Error de conversión al convertir el valor varchar 'LA EXPRESION 2+(4+2) TIENE COMO RESULTADO' al tipo de d
```

Donde el error es claro en mencionar que la expresión no puede concatenarse con un valor (entero). Más adelante explicaremos las funciones **CAST** y **CONVERT** para realizar conversiones entre tipos de datos.

CASO DESARROLLADO N° 3.31:

Script que permita mostrar el resultado de la expresión $\text{Celsius} = (\text{Fahrenheit} - 32) * 5/9$. Use la sentencia **SELECT** y el **PRINT**.

Solución con **SELECT**:

```
SELECT '100' [CELSIUS],
       (100-32)*5/9 [FAHRENHEIT]
GO
```

En el script se define dos columnas: en la primera, se mostrará los grados a convertir y en la segunda, se imprime el resultado de la expresión. Tenga en cuenta que en la sentencia **SELECT** se usa la palabra **AS** para definir las cabeceras.

	CELSIUS	FAHRENHEIT
1	100	37

8 views

1 like

0 comments

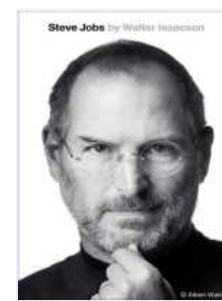
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

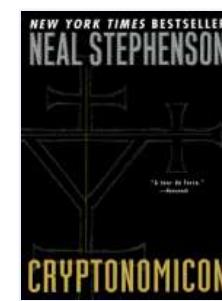
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

En la solución con PRINT el resultado de la expresión tiene que ser convertida a cadena | usó la función STR, pero al aplicarlo este genera espacios en blanco en el lado izquierdo de convertida, por tanto se aplicó la función LTRIM que permite eliminar espacios en blanco izquierdo (L). Tenga en cuenta que el paréntesis es el operador con más alta prioridad.

CASO DESARROLLADO N° 3.32

Script que permita mostrar el revés del número 45. Use la sentencia SELECT y el PRINT.

Solución con SELECT:

```
SELECT '45' AS [NUMERO],
       45/10 + (45%10)*10 AS [REVES]
GO
```

En el script se hace uso del operador % para poder obtener el resto de la división, explicare resolviendo la expresión como lo haría el motor de base de datos:

$$45/10 + (45\%10)*10$$

- **45/10:** da como resultado 4 ya que se está dividiendo dos números enteros, en otro caso que la expresión devuelva decimales tendría que colocarlo de la siguiente manera: resultado de esta expresión es 4.5 para lo cual no nos serviría de nada dicha solución.
- **(45%10)*10:** da como resultado $(5) \times 10 \rightarrow 50$, ya que el resto de la división de 45 con 10 si sumamos ambos resultados obtenemos $50+4$ que es el revés del número 45.

Solución con PRINT:

```
PRINT 'EL NUMERO 45 AL REVES ES >> '+LTRIM(STR(45/10 + (45%10)*10))
GO
```

B. ASIGNACIÓN

SQL Server sólo cuenta con un operador universal para la asignación de valores. El operador tendrá que colocar en cualquier expresión que necesite asignar un valor de cualquier tipo

CASO DESARROLLADO N° 3.33:

Script que permita asignar el valor de PI=3.1415 a una variable.

Solución con SELECT:

8 views

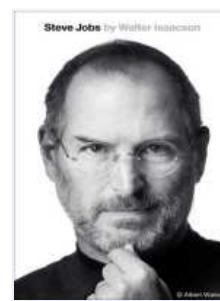
1 like

0 comments

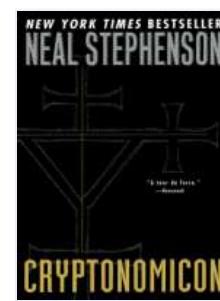
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

124

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el script se declara una variable de tipo Float, cada vez que se declara una variable a nivel SQL debe añadirle un arroba (@) al iniciar su declaración seguidamente se declara como FL el valor a almacenar es decimal.

En la segunda línea se asigna el valor 3.1415 a la variable @PI, dentro de TRANSACT-SQL lo se realiza con **SET**. La impresión de la variable con la sentencia **SELECT** se tiene que re mismo símbolo con el que se declaró, es decir, el arroba (@) no debe despegarse de dic tenga en cuenta que si no coloca este operador la sentencia **SELECT** asumirá que PI sin ar columna de alguna tabla.

Solución con **PRINT**:

```
DECLARE @PI FLOAT
SET @PI=3.1415
PRINT 'EL VALOR DE PI ES: '+LTRIM(CAST(@PI AS FLOAT))
GO
```

En el script con **PRINT** ocurre un desperfecto al momento de imprimir el valor de la vari usamos la función **STR** como en los ejercicios anteriores ocurriría que la respuesta no tomar los decimales, es decir, el resultado sería 3. En este caso se tiene que usar una función e en conversiones como lo es **CAST** que permite convertir a un tipo específico de datos, n definiremos dicha función.

CASO DESARROLLADO N° 3.34:

Script que permita capturar en una variable el correo electrónico del pasajero con código variables TRANSACT-SQL, **PRINT** y **SELECT** para la impresión del correo.

Solución con **PRINT**:

```
DECLARE @CORREO VARCHAR(50)
SELECT @CORREO=EMAIL
FROM PASAJERO
WHERE IDPASAJERO='P0004'
PRINT 'EL EMAIL DEL PASAJERO CON CODIGO P0004 ES: >> ' + @CORREO
GO
```

En el script se declara una variable de tipo **VARCHAR(50)** para poder almacenar el correo del pasajero, ahora la sentencia **SELECT** se comportará como asignador de valor, es decir, d variable @CORREO hay que tener en cuenta que la sentencia **SELECT** debe emitir un solo ' que si la consulta se implementa mal y devolviera varios registros la variable no lo soport sólo permite asignar valores de tipo **VARCHAR**.

8 views

1 like

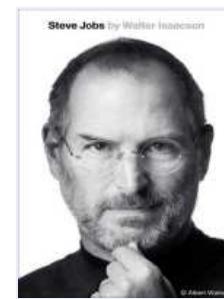
0 comments

eMacro - Programación Transact SQL Server 2012-4.pdf

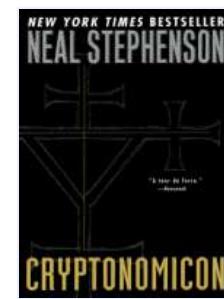
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

Solución con SELECT:

```
DECLARE @CORREO VARCHAR(50)
SELECT @CORREO=EMAIL
FROM PASAJERO
WHERE IDPASAJERO='P0004'
SELECT 'P0004' AS [IDPASAJERO],
@CORREO AS [EMAIL]
GO
```

En este script se invoca a otro sentencia SELECT la primero asignara el correo a la variable y el segundo mostrara en pantalla el resultado.

C. LÓGICOS

Los operadores lógicos tienen por misión comprobar la veracidad de alguna condición. Los operadores de comparación, devuelven el tipo de datos BOOLEAN con el valor Verdadero(TRUE) o Falso(FALSE) o Desconocido(UNKNOWN).

AND	Representa a la Y lógica en la cual dos expresiones deben ser TRUE para poder devolver TRUE.
ANY	Devuelve TRUE si alguna expresión del conjunto de expresiones es TRUE.
BETWEEN	Devuelve TRUE si el valor se encuentra dentro de un rango ya sea numérico o de cadena.
EXISTS	Devuelve TRUE si una determinada subconsulta devuelve por lo menos una fila de registros.
IN	Devuelve TRUE si el operando se encuentra dentro de una lista de valores específicos.
LIKE	Devuelve TRUE si el operando coincide a lo más con un patrón específico. Dicho patrón contiene la cadena de caracteres que se va a buscar en una expresión, los comodines que se pueden usar son: % Representa uno o más caracteres. Puede ser cualquier tipo de carácter ya sea numérico, textual o símbolo. _ Representa a un solo carácter de cualquier tipo. [] Representa cualquier carácter individual dentro de un intervalo o conjunto de caracteres. [^] Representa cualquier carácter individual fuera del intervalo especificado.
IS NOT NULL	Representa que el contenido de una columna no este vacía.
NOT	Invierte el valor booleano de una expresión del mismo tipo.
OR	Representa la O lógica en la cual dos expresiones sólo serán de tipo FALSE cuando ambas sean FALSE.
SOME	Devuelve TRUE si alguna de las comparaciones de un conjunto de comparaciones es TRUE.

En la mayoría de los casos cuando se usa un operador lógico se necesitará también síntesis de operadores.

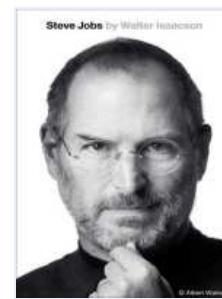
8 views

1 0

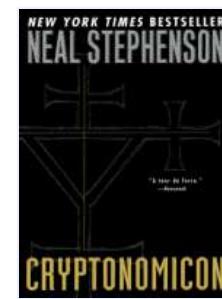
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

126 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.35:

Script que permita mostrar los pasajeros que sean de código de país 0001 y tengan correo en HOTMAIL. Use el operador lógico AND.

PASAJERO		
	Nombre de columna	Tipo comprimido
1	IDPAJERO	char(5)
	NOMBRES	varchar(50)
	IDPAIS	char(4)
	TELEFONO	char(15)
	EMAIL	varchar(50)

Para mostrar los registros hay 2 condiciones; por lo tanto, debemos unir ambas expresiones de un operador lógico en este caso usaremos AND ya que está obligado a ser del país 0001 sea de correo HOTMAIL. Veamos la lista de pasajeros:

	IDPAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1	P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2	P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3	P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4	P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5	P0005	LUZ LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM
6	P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7	P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8	P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
9	P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
10	P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM

```
SELECT *
FROM PASAJERO
WHERE IDPAIS='0001' AND EMAIL LIKE '%HOTMAIL%'
GO
```

Como visualizará en el script cada expresión de la cláusula WHERE debe tener un resultado que el operador AND pueda devolver una respuesta lógica y poder listar los registros dadas condiciones. Analicemos la condicional con el primer registro:

P0001 ANGELA TORRES LAZARO 0001 999999999 ATORRES@HOTMAIL.COM.

1	IDPAIS='0001'	AND	EMAIL LIKE '%HOTMAIL%'
2	'0001'='0001'	AND	'ATORRES@HOTMAIL.COM' LIKE '%HOTMAIL%'
3	TRUE	AND	TRUE

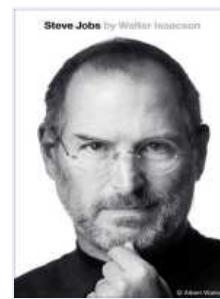
8 views

1 0

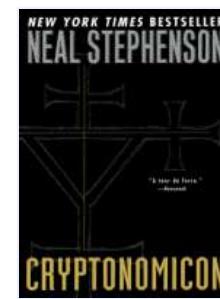
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

Entonces la imagen siguiente muestra los registros que cumplen con la condición:

	IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1	P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2	P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM

En la imagen se muestran sólo 2 registros, analicemos porque el registro 5 no aparece en anterior:

P0005 LUZ LAZARO MENOR 0001 99999999 LLAZARO@GMAIL.COM

1	IDPAIS='0001'	AND	EMAIL LIKE '%HOTMAIL%'
2	'0001'='0001'	AND	'LLAZARO@GMAIL.COM' LIKE '%HOTMAIL%'
3	TRUE	AND	FALSE
4			FALSE

Según la tabla de comparaciones el resultado de una de las expresiones es FALSE; por tanto, final es FALSE y no llega a mostrarse en el resultado de la consulta.

CASO DESARROLLADO N° 3.36:

El administrador de la aerolínea anualmente hace una comparación entre el total de costos en la reserva de acuerdo a un monto establecido como meta. Se pide implementar un script que muestre un mensaje de **EL MONTO NO SUPERA AL BASE ANUAL** solo en caso que el año a comparar los costos no superara el valor programado, caso contrario mostrará el mensaje **EL MONTO EXcede LA BASE ANUAL**. Considere que los valores de la base anual y el año a comparar deben ser ingresados por el usuario. Use operadores Transact-SQL.

RESERVA		
IDRESERVA	COSTO	FECHA

Solución con ANY:

```

DECLARE @AÑO INT=2012
DECLARE @MONTO MONEY=10000

IF @MONTO>ANY(
    SELECT SUM(COSTO)
    FROM RESERVA
    GROUP BY YEAR(FECHA)
    HAVING YEAR(FECHA)=@AÑO
)

```

8 views

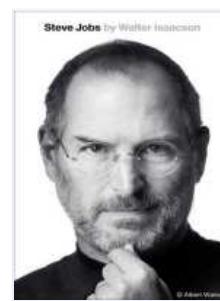
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

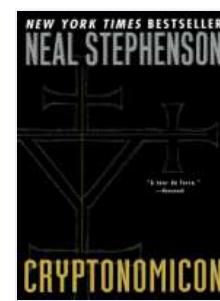
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True

128 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Solución con **SOME**:

```

DECLARE @AÑO INT=2012
DECLARE @MONTO MONEY=10000

IF @MONTO>SOME(
    SELECT SUM(COSTO)
    FROM RESERVA
    GROUP BY YEAR(FECHA)
    HAVING YEAR(FECHA)=@AÑO
)
PRINT 'EL MONTO NO SUPERA LA BASE ANUAL'
ELSE
PRINT 'EL MONTO CUMPLE CON LA BASE ANUAL'
GO

```

En el script se declaró dos variables **@AÑO** y **@MONTO** con valores predeterminados; para modificar dichos valores para probar el script.

Luego se compara el **@MONTO** con una consulta escalar (se dice escalar cuando el resultado de la consulta es un solo valor), analizaremos la consulta desde el **FROM** que hace referencia a la tabla **RESERVA** de donde obtendremos los costos a comparar, **GROUP BY YEAR(FECHA)** agrupa los registros por años ya que el caso solicita comparaciones por años, **HAVING YEAR(@AÑO)** permite condicionar el resultado de la agrupación por año es por eso que se iguala el valor ingresado por el usuario, finalmente **SUM(COSTO)** permite acumular los costos según la agrupación de la sentencia.

SOME o **ANY** devuelve **TRUE** cuando la comparación especificada también es **TRUE** entre **@MONTO>SOME**; en caso sea verdadero el mensaje será **EL MONTO NO SUPERA LA BASE ANUAL**, en caso contrario mostrará el mensaje **EL MONTO CUMPLE CON LA BASE ANUAL**.

Particularmente si asignamos **2012** a la variable **@AÑO** y el monto base es **10000**, el resultado es:

EL MONTO CUMPLE CON LA BASE ANUAL

El acumulado de costos en el año 2012 es **13800.00** que supera el monto base anual, para obtener este resultado el siguiente script:

```

SELECT SUM(COSTO)
FROM RESERVA
GROUP BY YEAR(FECHA)
HAVING YEAR(FECHA)=2012
GO

```

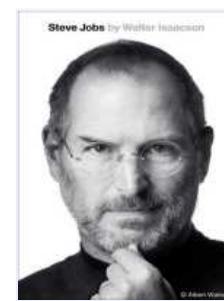
8 views

1 0

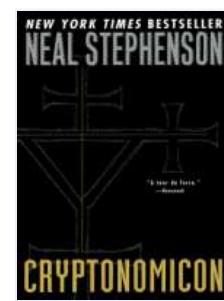
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CASO DESARROLLADO N° 3.37:

Script que permita mostrar las reservas cuyo costo se encuentre entre 400 y 700 inclusive e Use el operador lógico BETWEEN.

RESERVA		
IDRESERVA		
COSTO		
FECHA		

```
SELECT *
FROM RESERVA
WHERE COSTO BETWEEN 400 AND 700
GO
```

En el script se condiciona la columna **COSTO** para que muestre aquellas reservas que se entre 400 y 700. También se pudo implementar obviando el operador **BETWEEN** de la sigue

```
SELECT *
FROM RESERVA
WHERE COSTO>=400 AND COSTO<=700
GO
```

CASO DESARROLLADO N° 3.38:

Script que permita mostrar los pasajeros cuya letra inicial de su nombre se encuentre ent el operador lógico BETWEEN.

PASAJERO			
Nombre de columna	Tipo comprimido	Aceptación de valores NULL	
IDPASAJERO	char(5)	No	
NOMBRES	varchar(50)	No	
IDPAIS	char(4)	No	
TELEFONO	char(15)	No	
EMAIL	varchar(50)	No	

Esta vez veremos como BETWEEN también especifica rangos de tipo carácter. Se mostraremos el script que permite dar solución al caso:

8 views

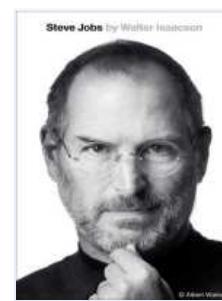
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

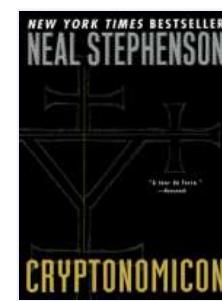
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Unlikely Heroine Who Outwitted America's Enemies

130 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el script usamos la función LEFT que permite obtener el primer carácter de la columna para poder comparar si dicha letra se encuentra en el rango de A hasta L. Mostramos a continuación otra forma de poder implementar la misma consulta:

```
SELECT *
FROM PASAJERO
WHERE LEFT(NOMBRES,1) LIKE '[A-L]'
ORDER BY NOMBRES
GO
```

Esta vez usamos el operador **LIKE** que permite comparar sólo cadenas de caracteres, a **'[A-L]'** hacemos referencia que las letras se encuentren entre **A** y **L** inclusive ellos mismos. La siguiente muestra el resultado de la consulta:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3 P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
4 P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
5 P0005	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
6 P0010	LINDA TUIME VARAS	0008	957564526	LTUIME@HOTMAIL.COM
7 P0005	LUZ LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM

También podríamos negar la condicional usando el operador NOT para poder listar a los pasajeros cuya letra inicial no se encuentre entre A y L. El script sería de la siguiente forma:

```
SELECT *
FROM PASAJERO
WHERE NOT LEFT(NOMBRES,1) BETWEEN 'A' AND 'L'
ORDER BY NOMBRES
GO
```

Como notará es la misma implementación que se realizó para mostrar los pasajeros cuya letra inicial se encuentre entre A y L sólo que aquí se niega el resultado lógico e invierte el contenido del resultado siguiente:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
2 P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
3 P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM

CASO DESARROLLADO N° 3.39:

8 views

1 like

0 comments

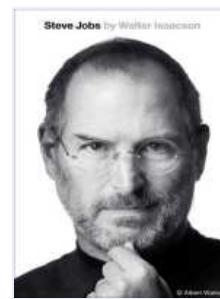
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

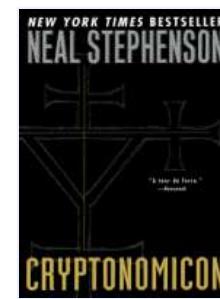
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

```
SELECT *
FROM RESERVA
WHERE FECHA BETWEEN '01/01/2011' AND '31/12/2011'
GO
```

En el script anterior se muestra como se puede implementar el operador BETWEEN en consulta con datos de tipo fecha, para este caso al referenciar fechas se tienen que encerrar en simples.

También podríamos resolver el mismo caso pero usando el operador AND de la siguiente forma:

```
SELECT *
FROM RESERVA
WHERE FECHA>='01/01/2011' AND FECHA<='31/12/2011'
GO
```

Y otra opción mucho más corta que las anteriores podría ser el uso de la función YEAR que nos devuelva los rangos en un solo valor, así podríamos comprobar por medio de un solo valor numérico.

```
SELECT *
FROM RESERVA
WHERE YEAR(FECHA)=2011
GO
```

CASO DESARROLLADO N° 3.40:

Script que permita verificar si un determinado PAÍS fue o no registrado en su tabla de origen y devolver un mensaje para cada caso.

PAÍS	
IDPAÍS	PAÍS
1	PERU
2	ARGENTINA
3	CHILE
4	ECUADOR
5	BRAZIL

Primero debemos verificar los registros de la tabla PAÍS.

IDPAÍS	PAÍS
1	PERU
2	ARGENTINA
3	CHILE
4	ECUADOR
5	BRAZIL

8 views

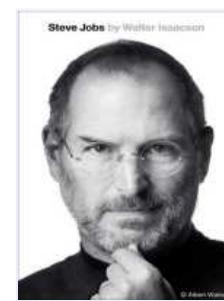
1 like

0 comments

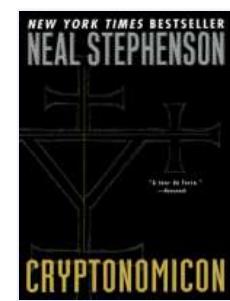
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

132

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

```

DECLARE @PAIS VARCHAR(40)='CHINA'
IF EXISTS(SELECT P.IDPAIS
          FROM PAIS P
          WHERE P.PAIS=@PAIS)
    PRINT 'EL PAIS YA SE ENCUENTRA REGISTRADO'
ELSE
    PRINT 'EL PAIS NO SE ENCUENTRA REGISTRADO'
GO

```

La variable @PAIS esta inicializada con el valor CHINA que no se encuentra registrada dentro de la tabla PAIS, entonces el operador EXISTS evalúa el resultado de la consulta si tiene un registro con el valor CHINA en este caso mostrará el mensaje EL PAIS YA SE ENCUENTRA REGISTRADO, caso contrario mostrará el mensaje EL PAIS NO SE ENCUENTRA REGISTRADO como sucede en el caso del país CHIN, que no se encuentra registrado dentro de la tabla PAIS.

El operador EXISTS sólo evalúa la existencia de por lo menos un registro para poder dar como resultado una expresión determinada.

CASO DESARROLLADO N° 3.41:

Script que permita mostrar los pagos realizados en un determinado año. Use el operador

PAGO	
IDRESERVA	
IDPASAJERO	
FECHA	
MONTO	

```

DECLARE @AÑO INT=2011
SELECT *
FROM PAGO
WHERE YEAR(FECHA) IN (@AÑO)
GO

```

En el script se declara la variable @AÑO de tipo entero que tiene por misión ser el valor de la columna FECHA dentro de la consulta. En este caso se inicializa con 2011 para buscar los pagos realizados en el año 2011.

Al especificar la condición se usó la función YEAR para determinar el año de la columna FECHA.

8 views

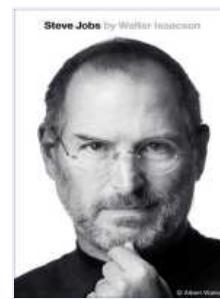
1 like

0 comments

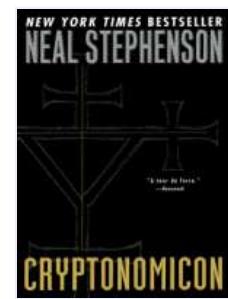
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CASO DESARROLLADO N° 3.42:

Script que permita mostrar los pasajeros condicionados a tres posibles países. Use el operador IN.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELÉFONO	char(15)	No
EMAIL	varchar(50)	No

```

DECLARE @PAIS1 VARCHAR(40)='PERU'
DECLARE @PAIS2 VARCHAR(40)='ECUADOR'
DECLARE @PAIS3 VARCHAR(40)='CHILE'

SELECT P.*
  FROM PASAJERO P
 WHERE P.IDPAIS IN (
    SELECT PA.IDPAIS
      FROM PAIS PA
     WHERE PA.PAIS IN (@PAIS1,@PAIS2,@PAIS3)
)
GO
  
```

En el script se declaran tres variables de tipo varchar que permitirán inicializar los tres posibles países para que la consulta pueda mostrar a los pasajeros que cumplan con dicha condición.

Esta vez la columna **IDPAIS** es la condición para mostrar los pasajeros, entonces se usa el operador **IN** para determinar si los códigos de los países registrados en la tabla **PASAJERO** se encuentran en la tabla **PAIS** en este caso se aplica una subconsulta para dicho proceso. En esta subconsulta se aplica nuevamente el operador **IN** para determinar si los países asignados a las variables se encuentran en la tabla **PAIS** de la siguiente forma **IN (@PAIS1,@PAIS2,@PAIS3)**.

CASO DESARROLLADO N° 3.43:

Script que permita mostrar los pasajeros cuyo nombre inicie con la letra A. Use el operador comodín %.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No

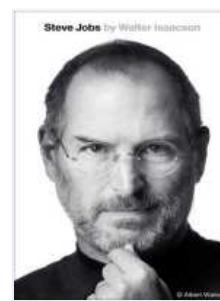
8 views

1 0

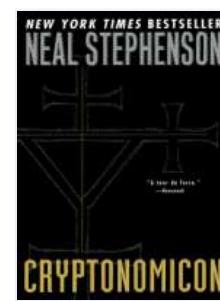
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

134

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012Primero visualizaremos los registros de la tabla **PASAJERO**.

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4 P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5 P0005	LULU LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM
6 P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7 P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8 P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
9 P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
10 P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM

```
SELECT P.*  
FROM PASAJERO P  
WHERE P.NOMBRES LIKE 'A%'  
GO
```

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM

En el script se condiciona la columna **NOMBRES** donde el primer carácter es la letra A cualquier carácter, esto está representado por el comodín % usando para este caso **LIKE** comparación de tipo carácter.

Ahora, haremos una pequeña modificación al caso para poder mostrar los pasajeros que tengan incluido la letra A en cualquier lugar.

```
SELECT P.*  
FROM PASAJERO P  
WHERE P.NOMBRES LIKE '%A%'  
GO
```

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4 P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5 P0005	LULU LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM
6 P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7 P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8 P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
9 P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
10 P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM

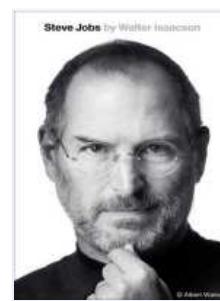
8 views

1 0

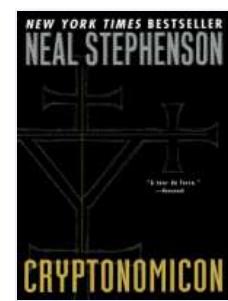
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

Para mostrar los pasajeros cuya segunda letra de su nombre sea la letra tendríamos que script de la siguiente manera:

```
SELECT P.*  
FROM PASAJERO P  
WHERE P.NOMBRES LIKE '_A%'  
GO
```

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM

En este caso usamos el comodín _ que representa a un solo carácter de cualquier tipo, se se colocó la letra A justamente en el segundo lugar, como lo solicita el caso no olvidarse el comodín % para representar los demás caracteres un error sería colocar de la siguiente forma que aquí se estaría buscando los nombres de los pasajeros que sólo tengan dos caracteres.

CASO DESARROLLADO N° 3.44:

Script que permite mostrar los pasajeros cuyo segundo carácter de su nombre sea la letra A el operador LIKE y el comodín % _ [].

PASAJERO			
Nombre de columna	Tipo comprimido	Aceptación de valores NULL	
IDPASAJERO	char(5)	No	
NOMBRES	varchar(50)	No	
IDPAIS	char(4)	No	
TELEFONO	char(15)	No	
EMAIL	varchar(50)	No	

```
SELECT P.*  
FROM PASAJERO P  
WHERE P.NOMBRES LIKE '_[AOU]%'  
GO
```

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
P0005	Luz Lazaro Menor	0001	999999999	LLAZARO@GMAIL.COM
P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM

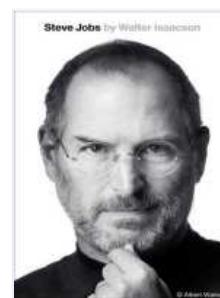
8 views

1 0

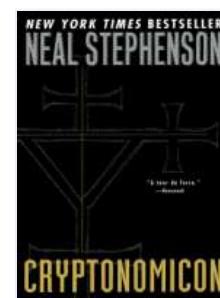
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

136 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.45:

Script que permita mostrar los pasajeros cuyo segundo carácter de su nombre sea una vocal A, O y U. Use el operador **LIKE** y el comodín **% ^ _ []**.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

```
SELECT P.*  
FROM PASAJERO P  
WHERE P.NOMBRES LIKE '_[^AOU]%'  
GO
```

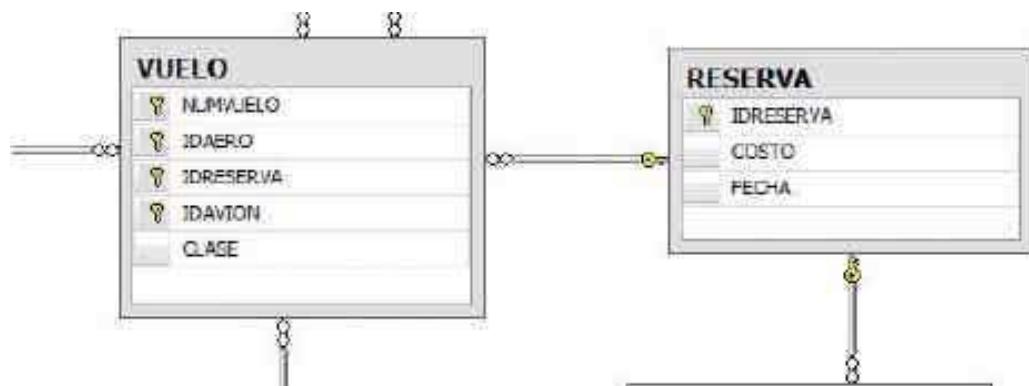
	IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1	P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM

En el script se especifica todas las vocales dentro del corchete y se les niega su partición condición colocando el comodín ^ dentro del corchete. También se puede implementar el haciendo uso del operador NOT así como se muestra en el siguiente script:

```
SELECT P.*  
FROM PASAJERO P  
WHERE NOT P.NOMBRES LIKE '_[AEIOU]%'  
GO
```

CASO DESARROLLADO N° 3.46:

Script que permita mostrar los pagos realizados por un determinado pasajero. Usar sus operadores T-SQL.



8 views

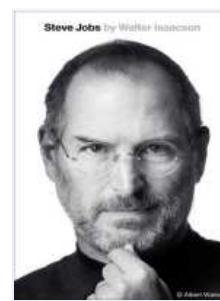
1 like

0 comments

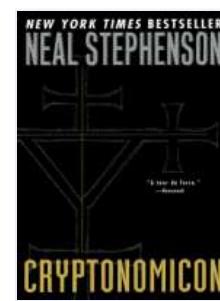
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

```
DECLARE @PASAJE VARCHAR(40)='FERNANDA TORRES LAZARO'
SELECT *
    FROM PAGO
    WHERE IDPASAJERO=
        (SELECT IDPASAJERO
            FROM PASAJERO
            WHERE NOMBRES=@PASAJE
        )
GO
```

En el script se declara la variable **@PASAJE** de tipo **VARCHAR(40)** ya que almacenará el nombre del pasajero determinado. A la vez se le asigna el valor a la misma variables obviando en este caso la cláusula **SET**. También permite asignar un valor. Mostramos una analogía entre el uso y la asignación directa de

```
DECLARE @PASAJE VARCHAR(40)='FERNANDA TORRES LAZARO'
```

o

```
DECLARE @PASAJE VARCHAR(40)
SET @PASAJE='FERNANDA TORRES LAZARO'
```

Luego se aplica la subconsulta dentro de la cláusula **WHERE** esperando un valor escalar a la columna **IDPASAJERO** desde dicha subconsulta. Tiene que evitar que la subconsulta genere más de un resultado ya que la variable **IDPASAJERO** espera un solo valor.

3.7. COMBINACIÓN DE TABLAS JOIN, LEFT JOIN, RIGHT JOIN

En determinada ocasión se tendrá que unir más tablas para poder combinar los valores y mostrarlos juntos en una consulta, aquí se tendrá que usar la combinación de tablas por medio de la cláusula **JOIN**. Hay que diferenciar que existen dos tipos de combinaciones las internas y las externas.

Las internas devuelven todas las filas que cumplen la combinación de la primera entidad con la segunda. En el caso no encontrarse coincidencia de valores no emite mensaje de error simplemente muestra ningún registro.

Las externas son combinaciones donde las filas resultantes no son directamente de la tabla que se combina, podrían ser de la izquierda, derecha o completa.

Sintaxis:

CLAUSULA JOIN
INNER JOIN <nombreTabla1> ON <nombreTabla2>
JOIN <nombreTabla1>

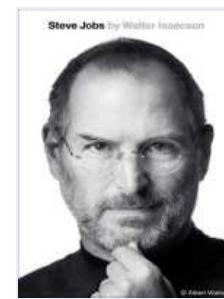
8 views

1 0

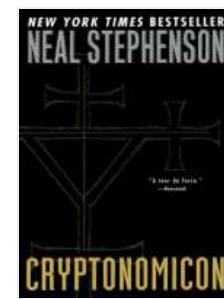
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

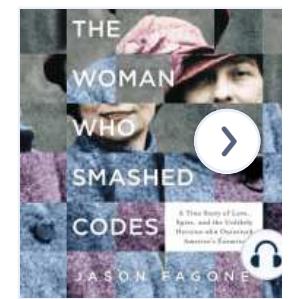
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and Violent Genius in the Heart of Silicon Valley

138 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.47:

Script que permite mostrar la combinación de la tabla PASAJERO y la tabla PAGO.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

PAGO		
IDRESERVA	IDPASAJERO	FECHA

```
SELECT *
FROM PASAJERO PAS
INNER JOIN PAGO PAG
ON PAS.IDPASAJERO=PAG.IDPASAJERO
GO
```

En el script se hace referencia a la primera tabla con la cláusula **FROM** y a la segunda tabla **JOIN**, a cada una de las tablas se le asigna un alias. Seguidamente se debe especificar cuál es de unión entre esas dos tablas como observará en las imágenes que muestran las columnas involucradas. La columna común en ambas es **IDPASAJERO**; por lo tanto, esa columna es el factor de unión especificado en **ON PAS.IDPASAJERO=PAG.IDPASAJERO**.

El resultado de la consulta es la multiplicación entre las filas de la tabla **PASAJERO** por la tabla **PAGO**, dando como resultado 130 filas repartidas entre 10 pasajeros registrados y 13 filas en la tabla **PAGO**, pero como notará en la imagen resultante sólo aparecen 13 filas es porque la definición de la columna de unión **IDPASAJERO**. En la siguiente imagen se muestra el resultado:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL	IDRESERVA	IDPASAJERO	FECHA	M
P0005	LUZ LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM	1	P0005	2011-10-01	5
P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM	2	P0003	2011-10-06	5
P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM	3	P0008	2011-11-14	5
P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM	4	P0002	2011-11-16	12
P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	5	P0001	2011-12-12	11
P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM	6	P0006	2011-12-17	5
P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM	7	P0003	2012-01-14	4
P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM	8	P0003	2012-01-15	1
P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM	9	P0008	2012-02-01	11
P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM	10	P0002	2012-04-02	11
P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	11	P0001	2012-04-09	12
P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM	12	P0006	2012-08-01	4
P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM	13	P0003	2012-08-20	3

Podríamos implementar el mismo caso sin especificar **INNER JOIN** en la solución de la siguiente forma:

```
SELECT *
FROM PASAJERO PAS, PAGO PAG
WHERE PAS.IDPASAJERO=PAG.IDPASAJERO
GO
```

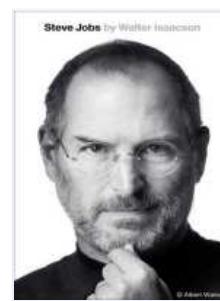
8 views

1 0

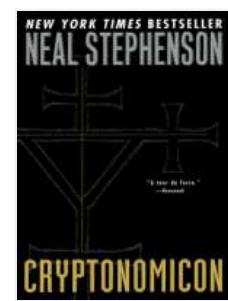
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

Ahora, como notó en todas las consultas mostradas para la solución de este caso se combina las columnas de la tabla **PASAJERO** así como todas las columnas de la tabla **PAGO**; esto se logró en la implementación se colocó *, podríamos modificar el script de la siguiente manera:

```
SELECT PAS.*  
      FROM PASAJERO PAS  
      JOIN PAGO PAG  
        ON PAS.IDPASAJERO=PAG.IDPASAJERO  
      GO
```

En el script se especifica sólo las columnas de la tabla **PASAJERO** mediante el alias **PAS**. El resultado sería como se muestra a continuación:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0005	Luz Lazaro Menor	0001	99999999	LLAZARO@GMAIL.COM
2 P0003	Maria Zamora Mejia	0005	957564526	MZAMORA@GMAIL.COM
3 P0008	Heidi Rengifo Reategui	0004	957564526	HRENGIFO@HOTMAIL.COM
4 P0002	Fernanda Torres Lazaro	0001	99999999	FTORRES@HOTMAIL.COM
5 P0001	Angela Torres Lazaro	0001	99999999	ATORRES@HOTMAIL.COM
6 P0006	Karla Gallegos Silva	0007	957564526	KGALLEGOS@HOTMAIL.COM
7 P0003	Maria Zamora Mejia	0005	957564526	MZAMORA@GMAIL.COM
8 P0003	Maria Zamora Mejia	0005	957564526	MZAMORA@GMAIL.COM
9 P0008	Heidi Rengifo Reategui	0004	957564526	HRENGIFO@HOTMAIL.COM
10 P0002	Fernanda Torres Lazaro	0001	99999999	FTORRES@HOTMAIL.COM
11 P0001	Angela Torres Lazaro	0001	99999999	ATORRES@HOTMAIL.COM
12 P0006	Karla Gallegos Silva	0007	957564526	KGALLEGOS@HOTMAIL.COM
13 P0003	Maria Zamora Mejia	0005	957564526	MZAMORA@GMAIL.COM

Ahora, con los campos de ambas tablas disponibles se podrá administrar la consulta de la misma manera. Se podrá seleccionar las columnas que crea conveniente para su consulta. Veamos el siguiente script que muestra los pasajeros con las siguientes columnas **IDPASAJERO**, **NOMBRES**, **PAIS** y **TELEFONO**:

```
SELECT PAS.IDPASAJERO,PAS.NOMBRES,PAI.PAIS,PAS.TELEFONO  
      FROM PASAJERO PAS  
      JOIN PAIS PAI  
        ON PAS.IDPAIS=PAI.IDPAIS  
      GO
```

En la siguiente imagen se muestra el resultado de la consulta:

IDPASAJERO	NOMBRES	PAIS	TELEFONO
1 P0001	ANGELA TORRES LAZARO	PERU	99999999
2 P0002	FERNANDA TORRES LAZARO	PERU	99999999
3 P0003	MARIA ZAMORA MEJIA	BRASIL	957564526
4 P0004	GUADALUPE ACOSTA FERRER	ARGENTINA	957564526
5 P0005	Luz Lazaro Menor	PERU	99999999
6 P0006	KARLA GALLEGOS SILVA	PARAGUAY	957564526
7 P0007	MERY CARMEN DE LA CRUZ	MEXICO	957564526

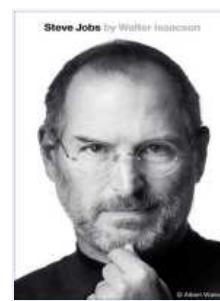
8 views

1 0

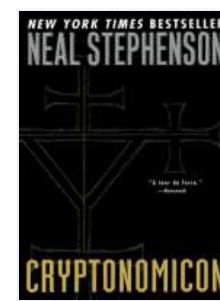
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Hidden Heroine Behind a Conspiracy

140 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.48:

Script que permita mostrar a los pasajeros con las siguientes columnas **IDPASAJERO, NOMBRE, FECHA DE PAGO, MONTO CANCELADO.**



```
SELECT PAS.IDPASAJERO, PAS.NOMBRES, PAI.PAIS, PAG.FECHA, PAG.MONTO
FROM PASAJERO PAS
JOIN PAGO PAG ON PAS.IDPASAJERO=PAG.IDPASAJERO
JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
GO
```

En el script participan tres tablas ya que el caso hace referencia a columnas de estas, ha cuidado al asignar el nombre del alias a cada tabla. Como notará la columna de unión entre **PAGO** es **IDPASAJERO**, mientras que entre **PASAJERO** y **PAIS** la columna de unión es **IDPAIS**.

La imagen siguiente muestra el resultado de la consulta:

	IDPASAJERO	NOMBRES	PAIS	FECHA	MONTO
1	P0005	Luz Lazaro Menor	PERU	2011-10-01	500.00
2	P0003	Maria Zamora Mejia	BRASIL	2011-10-06	900.00
3	P0008	Heidi Rengifo Reategui	ECUADOR	2011-11-14	500.00
4	P0002	Fernanda Torres Lazaro	PERU	2011-11-16	1200.00
5	P0001	Angela Torres Lazaro	PERU	2011-12-12	1500.00
6	P0006	Karla Gallegos Silva	PARAGUAY	2011-12-17	590.00
7	P0003	Maria Zamora Mejia	BRASIL	2012-01-14	400.00
8	P0003	Maria Zamora Mejia	BRASIL	2012-01-15	1300.00
9	P0008	Heidi Rengifo Reategui	ECUADOR	2012-02-01	1000.00
10	P0002	Fernanda Torres Lazaro	PERU	2012-04-02	1800.00
11	P0001	Angela Torres Lazaro	PERU	2012-04-03	1200.00
12	P0006	Karla Gallegos Silva	PARAGUAY	2012-08-01	400.00
13	P0003	Maria Zamora Mejia	BRASIL	2012-08-20	800.00

CASO DESARROLLADO N° 3.49:

Script que permita mostrar a los pasajeros de un determinado PAIS ingresado por el usuario en la cláusula JOIN.

8 views

1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)

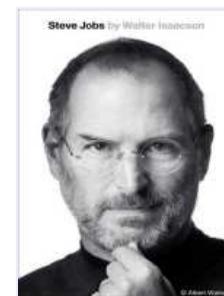
Save

Embed

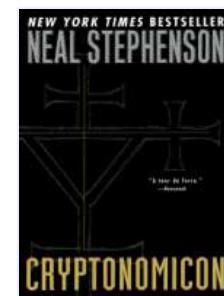
Share

Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

```
DECLARE @PAIS VARCHAR(50)=’PERU’
```

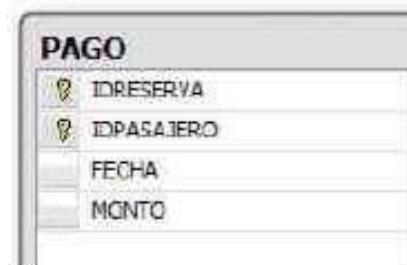
```
SELECT *
    FROM PASAJERO PAS
    JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
    WHERE PAI.PAIS=@PAIS
GO
```

Se declara la variable **@PAIS** para colocar el criterio de búsqueda en este caso buscar los pasajeros del país **PERÚ**. Se le adiciona la cláusula **WHERE** a la consulta para poder co país de los pasajeros que se necesitan mostrar como notará dentro del **WHERE** se puede r cualquiera de las columnas de ambas tablas, pero en este caso sólo será necesario indica del país por medio de **PAI.PAIS** que es el alias de la tabla **PAIS** y país representa el nombre

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL	IDPAIS	PAIS
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	0001	PERU
2 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM	0001	PERU
3 P0005	LULU LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM	0001	PERU

CASO DESARROLLADO N° 3.50:

Script que permita mostrar a los pasajeros con las siguientes columnas **IDPASAJERO**, **PAIS**, **FECHA DE PAGO**, **MONTO CANCELADO** sólo para aquellos pasajeros cuyo año de pa procedencia sea especificado por el usuario.



```
DECLARE @AÑO INT=2012
DECLARE @PAIS VARCHAR(40)=’PERU’
```

```
SELECT PAS.IDPASAJERO,PAS.NOMBRES,PAI.PAIS,PAG.FECHA,PAG.MONTO
    FROM PASAJERO PAS
    JOIN PAGO PAG ON PAS.IDPASAJERO=PAG.IDPASAJERO
    JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
    WHERE YEAR(PAG.FECHA)=@AÑO AND PAI.PAIS=@PAIS
GO
```

En el script se hace referencia a tres tablas y en la condicional se busca el año de la fe

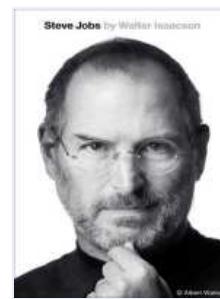
8 views

1 0

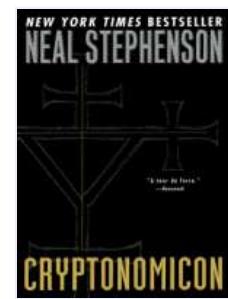
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon

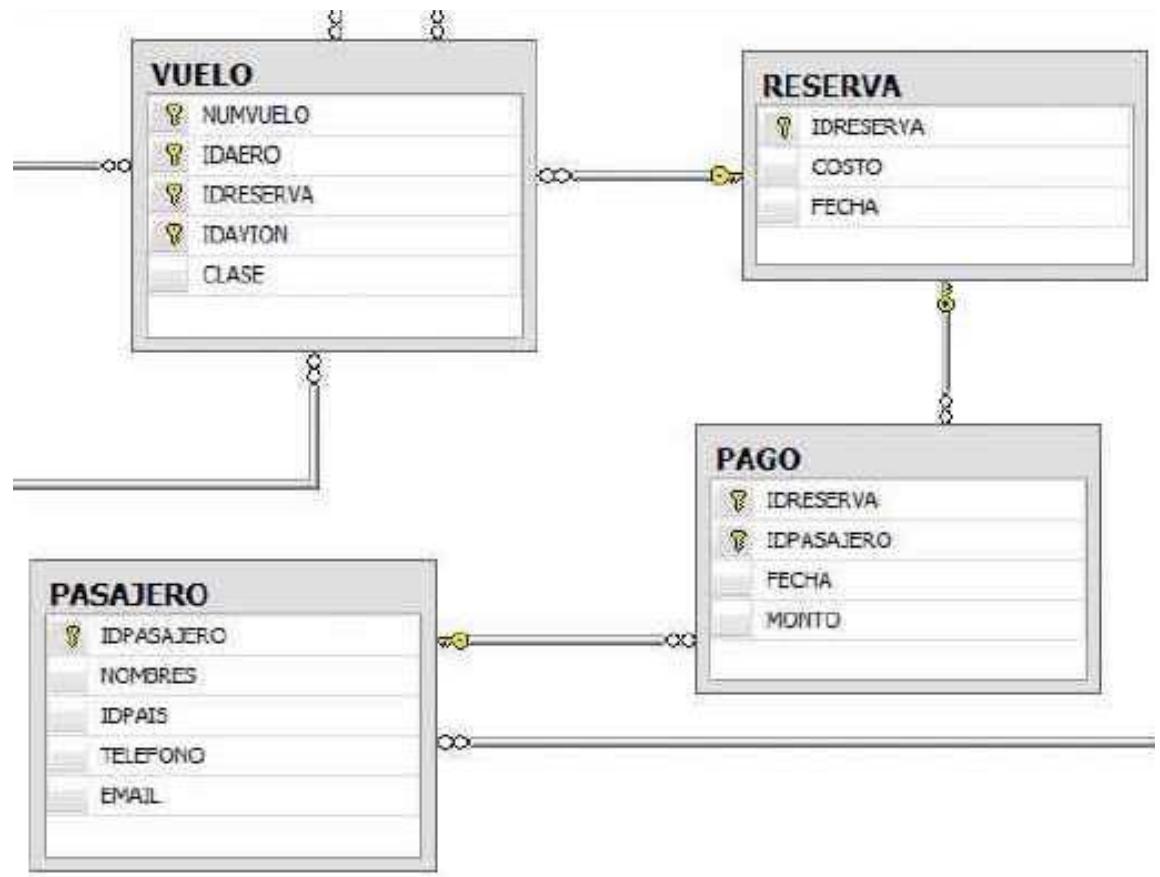


The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

142 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.51:

Script que permita mostrar las reservas de un determinado PASAJERO especificado por Finalmente, debe ordenar la fecha de reserva en forma descendente.



```

DECLARE @PASAJE VARCHAR(40)='ANGELA TORRES LAZARO'

SELECT RES.*
FROM PASAJERO PAS
JOIN PAGO PAG ON PAS.IDPASAJERO=PAG.IDPASAJERO
JOIN RESERVA RES ON PAG.IDRESERVA=RES.IDRESERVA
WHERE PAS.NOMBRES=@PASAJE
ORDER BY RES.FECHA DESC
GO
  
```

En el script se unen tres tablas **PASAJERO**, **PAGO** Y **RESERVA** que permitirá hacer comparaciones entre los valores a buscar. Note que no necesariamente se tienen que imprimir las columnas de la tabla **PASAJERO** que tiene asignada en el **FROM** cualquiera puede ser parte de la consulta, en este caso se especifican los campos de la tabla **RESERVA** por medio de su alias **RES.***.

La imagen a continuación muestra que la cliente **ANGELA TORRES LAZARO** tiene 2 reservas que cumplen con la condición especificada.

IDRESERVA	COSTO	FECHA
-----------	-------	-------

8 views

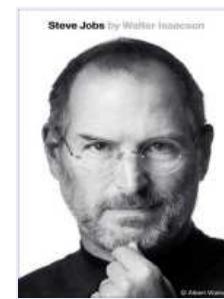
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

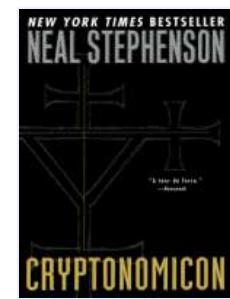
Uploaded by [anon_61286589](#)[Full description](#)

 Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

CASO DESARROLLADO N° 3.52:
 Script que permite mostrar todos los registros de la tabla pasajero y el país. Use LEFT JOIN

PASAJERO					
IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL	
P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	0001 PER
P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM	0001 PER
P0005	LUZ LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM	0001 PER
P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM	0002 ARG
NULL	NULL	NULL	NULL	NULL	0003 CHI
P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM	0004 ECU
P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM	0004 ECU
P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM	0005 BRA
NULL	NULL	NULL	NULL	NULL	0006 VEN
P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM	0007 PAR
P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM	0008 URU
NULL	NULL	NULL	NULL	NULL	0009 BOL
P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM	0010 MEX
NULL	NULL	NULL	NULL	NULL	0011 HON
NULL	NULL	NULL	NULL	NULL	0012 EEU
NULL	NULL	NULL	NULL	NULL	0013 PUE

PAÍS	
IDPAIS	PAÍS
0001	PER

```
SELECT *
  FROM PAÍS PAI
  LEFT JOIN PASAJERO PAS ON PAI.IDPAIS=PAS.IDPAIS
GO
```

En el script la tabla PASAJERO es referenciado a la unión externa izquierda eso quiere decir que la consulta deberá mostrar sus columnas en el lado izquierdo y forzar a la otra tabla a mostrar filas, eso significa que la tabla puede tener registros que no necesariamente se encuentren en la tabla izquierda. Cuando ocurre este caso la tabla izquierda mostrará NULL en los valores cruzados. La imagen muestra en la imagen siguiente:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL	IDPAIS	PAÍS
P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	0001	PER
P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM	0001	PER
P0005	LUZ LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM	0001	PER
P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM	0002	ARG
NULL	NULL	NULL	NULL	NULL	0003	CHI
P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM	0004	ECU
P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM	0004	ECU
P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM	0005	BRA
NULL	NULL	NULL	NULL	NULL	0006	VEN
P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM	0007	PAR
P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM	0008	URU
NULL	NULL	NULL	NULL	NULL	0009	BOL
P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM	0010	MEX
NULL	NULL	NULL	NULL	NULL	0011	HON
NULL	NULL	NULL	NULL	NULL	0012	EEU
NULL	NULL	NULL	NULL	NULL	0013	PUE

Si observa la imagen la interpretación que se puede dar aquí es que se muestran todos los países inclusive los que no participan dentro de la tabla PASAJERO. Chile, Venezuela, Bolivia, Honduras y Puerto Rico son países que aún no tienen pasajeros registrados.

Modificaremos el script para poder mostrar sólo a los países que aún no tienen pasajeros ya sabe entonces que los países de respuesta serían Chile, Venezuela, Bolivia, Honduras, EE. UU. y Puerto Rico. Veamos el script para la solución del caso:

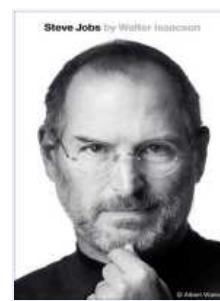
8 views

1 0

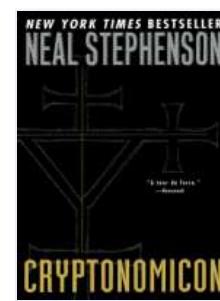
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Hidden Heroine Behind a Conspiracy

144

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Lo primero que se necesita es especificar que la consulta debe mostrar sólo las columnas **PAIS** esto se logra con **SELECT PAI.***, luego se condiciona que la columna **IDPAIS** de la tabla tenga un contenido nulo (NULL); eso quiere decir que el resultado será mostrar a todos los **IDPAIS** de la tabla **PASAJERO** sea **NULL**. La imagen siguiente muestra el resultado de la consulta:

IDPAIS	PAIS
0009	BOLIVIA
0003	CHILE
0012	EEUU
0011	HONDURAS
0013	PUERTO RICO
0006	VENEZUELA

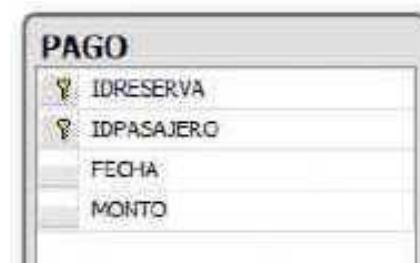
Mostraremos el script con la implementación de RIGHT JOIN reemplazando al anterior daba solución al caso propuesto, con esto queda demostrado que ambas pueden dar solución. Veamos el script:

```
SELECT PAI.*
FROM PASAJERO PAS
RIGHT JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
WHERE PAS.IDPAIS IS NULL
ORDER BY PAÍS
GO
```

En este caso la tabla derecha es PAIS; por tanto, la otra tabla deberá mostrar todas sus filas, así sólo registros de la tabla PAIS sólo cuando el IDPAIS de la tabla PASAJERO sea NULL.

CASO DESARROLLADO N° 3.53:

Script que permita mostrar los pasajeros que aún no han realizado ningún pago. Use LEFT JOIN.



```
SELECT PAS.*
FROM PASAJERO PAS
LEFT JOIN PAGO PAG ON PAS.IDPASAJERO=PAG.IDPASAJERO
```

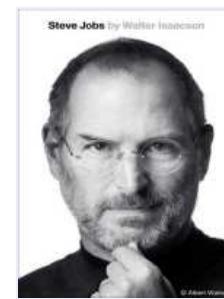
8 views

1 0

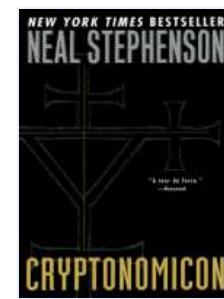
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

```
SELECT *
FROM PASAJERO PAS
LEFT JOIN PAGO PAG ON PAS.IDPASAJERO=PAG.IDPASAJERO
GO
```

	IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL	IDRESERVA	IDPASAJERO	FECHA
1	P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	5	P0001	2011
2	P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	11	P0001	2012
3	P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM	4	P0002	2011
4	P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM	10	P0002	2012
5	P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM	2	P0003	2011
6	P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM	7	P0003	2012
7	P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM	8	P0003	2012
8	P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM	13	P0003	2012
9	P0004	GLADALICE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM	NULL	NULL	NULL
10	P0005	LULU LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM	1	P0005	2011
11	P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM	6	P0006	2011
12	P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM	12	P0006	2012
13	P0007	NERLY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM	NULL	NULL	NULL
14	P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM	3	P0008	2011
15	P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM	9	P0008	2012
16	P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM	NULL	NULL	NULL
17	P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM	NULL	NULL	NULL

El resultado es que los pasajeros de código **P0004**, **P0007**, **P0009** y **P0010** son aquellos que aún no han realizado pago alguno, por eso la tabla **PAGO** lo muestra como **NULL** a dichos pasajeros. Para filtrar a estos pasajeros se debe condicionar cualquiera de las columnas de la tabla **PAGO** con la cláusula **WHERE PAG.IDPASAJERO IS NULL** o **WHERE PAG.IDRESERVA IS NULL** o **WHERE PAG.FECHA IS NULL**.

Seguidamente mostramos el script que modifica la cláusula LEFT JOIN por RIGHT JOIN para este caso.

```
SELECT PAS.*
FROM PAGO PAG
RIGHT JOIN PASAJERO PAS ON PAG.IDPASAJERO=PAS.IDPASAJERO
WHERE PAG.IDPASAJERO IS NULL
GO
```

Finalmente, podríamos haber usado la cláusula FULL JOIN que permite combinar LEFT JOIN y RIGHT JOIN dentro de una consulta. El script es como sigue:

```
SELECT PAS.*
FROM PASAJERO PAS
FULL JOIN PAGO PAG ON PAS.IDPASAJERO=PAG.IDPASAJERO
WHERE PAG.IDPASAJERO IS NULL
GO
```

En todos los casos el resultado es como se muestra a continuación:

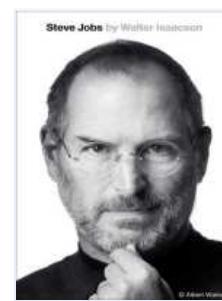
8 views

1 0

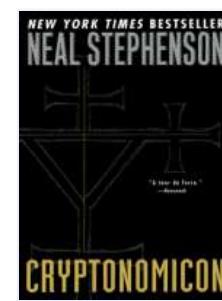
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

146 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.54:

Script que permita mostrar los registros de la tabla PASAJERO y PAGO de forma que : producto cartesiano entre sus filas. Use CROSS JOIN.

PASAJERO					
IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL	
P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	
P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	
P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	
P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	
P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	
P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	

PAGO					
IDRESERVA	IDPASAJERO	FECHA	MONTO		
1	P0005	2011-10-01			
2	P0003	2011-10-06			
3	P0008	2011-11-14			
4	P0002	2011-11-16			
5	P0001	2011-12-12			
6	P0006	2011-12-17			

```
SELECT *
  FROM PASAJERO
CROSS JOIN PAGO
GO
```

En el script se especifican las dos tablas involucradas esta vez usando **CROSS JOIN** que p combinación completa entre las tablas, si visualiza la cantidad de filas de la consulta no resultado es 130 rows ya que por cada fila de pasajeros es multiplicado por todas las fila PAGO realizando una combinación completa entre sus filas. La imagen a continuación combinación **CROSS JOIN**:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL	IDRESERVA	IDPASAJERO	FECHA
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	1	P0005	2011-10-01
2 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	2	P0003	2011-10-06
3 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	3	P0008	2011-11-14
4 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	4	P0002	2011-11-16
5 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	5	P0001	2011-12-12
6 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	6	P0006	2011-12-17

Query executed successfully.

3.8. RECUPERACIÓN DE DATOS AGRUPADOS GROUP BY, HAVING Y LAS FUNCIONES AGREGADAS SUM, COUNT, MAX, MIN Y AVG

Para recuperar información agrupada por algún criterio se debe implementar la cláusula dentro de la consulta **SELECT**.

El **GROUP BY** agrupa un conjunto de registros de acuerdo a los valores de una o más columnas de una tabla. Para una mejor implementación de las consultas agrupadas se puede usar las funciones agregadas como **Sum**, **Count**, **Max**, **Min** y **Avg**.

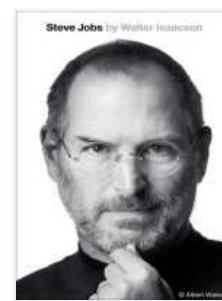
8 views

1 0

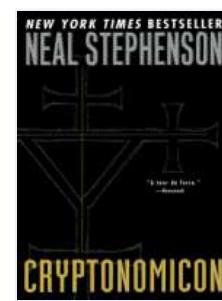
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

Donde:

- **ROLLUP:** genera filas de agregado en la cláusula Group By más filas de subtotal y también con un total general. El número de agrupaciones es igual al número de expresión de los elementos compuestos más uno, aquí justamente muestra los resultantes.
- **CUBE:** genera filas de agregado en la cláusula Group By más una fila de superagregación de tabulación cruzada. El número de agrupaciones es igual a 2^n donde n es el número de elementos compuestos.

CASO DESARROLLADO N° 3.55:

Script que permita mostrar los **IDPAIS** agrupados desde la tabla **PASAJERO**. Use la cláusula **BY**.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

```
SELECT P.IDPAIS
      FROM PASAJERO P
     GROUP BY P.IDPAIS
        GO
```

En el script se muestran los códigos de los países registrados en la tabla **PASAJERO**, haciendo en cuenta que dicho listado no repetirá los **IDPAIS** ya que se encuentran agrupados. El resultado muestra a continuación.

IDPAIS
1 0001
2 0002
3 0004
4 0005
5 0007
6 0008
7 0010

3.9. FUNCIONES AGREGADAS**A. SUM:**

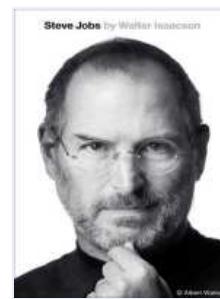
8 views

1 0

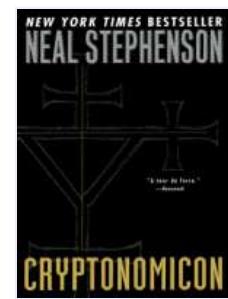
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon

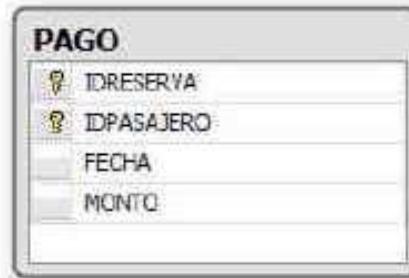


The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

148 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.56:

Script que permite mostrar el acumulado de los montos registrados en la tabla PAGO.



```
SELECT SUM(MONTO) AS [TOTAL MONTO]
FROM PAGO
GO
```

En el script se implementa la función **SUM** de la columna **MONTO** que permite acumular sin restricciones. El resultado se muestra en la siguiente imagen:

TOTAL MONTO	
1	12090.00

Si modificamos el caso y nos proponemos a determinar el acumulado de los montos, determinado año ingresado por el usuario, el script sería como sigue:

```
DECLARE @AÑO INT=2011

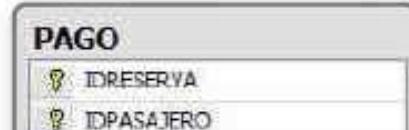
SELECT SUM(MONTO) AS [TOTAL MONTO]
FROM PAGO
WHERE YEAR(FECHA)=@AÑO
GO
```

En el script se adiciona la cláusula **WHERE** para poder condicionar el acumulado, es decir, si los montos sólo si el año es igual a la variable **@AÑO**.

TOTAL MONTO	
1	5190.00

CASO DESARROLLADO N° 3.57:

Script que permite mostrar el acumulado de los montos registrados en la tabla **PAGO** pero considere el año de la columna Fecha. Use la función agregada **SUM** y la cláusula **GROUP**



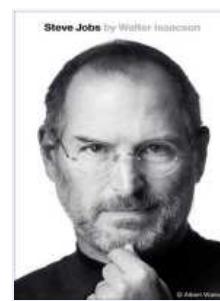
8 views

1 0

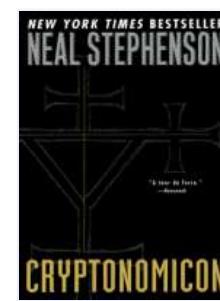
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

En el script se está agrupando los registros de la tabla **PAGO** desde la columna **FECHA**. Para solicitar los montos acumulados por años para lograr esto debemos agrupar dichas fechas por la cláusula **GROUP BY YEAR(FECHA)**. Las columnas definidas en la consulta son dos, la primera los años agrupados y el segundo mostrará el monto acumulado según el orden de agrupación, decir, por años. El resultado se muestra en la imagen siguiente:

AÑO	TOTAL MONTO
1	5190.00
2	6900.00

La imagen muestra claramente que en el año **2011** el acumulado es **5190** mientras que en **2012** es **6900** haciendo un monto total acumulado de **12090**.

Al mismo caso podríamos añadirle la columna, año, mes y el acumulado según estos dos scripts sería de la siguiente manera:

```
SELECT YEAR(FECHA) AS [AÑO],
       MONTH(FECHA) AS [MES], SUM(MONTO) AS [ACUMULADO]
  FROM PAGO
 GROUP BY YEAR(FECHA), MONTH(FECHA)
 GO
```

El resultado del script sería como sigue:

	AÑO	MES	ACUMULADO
1	2012	1	1700.00
2	2012	2	1000.00
3	2012	4	3000.00
4	2012	8	1200.00
5	2011	10	1400.00
6	2011	11	1700.00
7	2011	12	2090.00

Como notara existe un quiebre entre un año y sus meses, eso quiere decir que el monto es consecuencia del agrupamiento entre un determinado mes y un año. Por ejemplo en se muestran los acumulados de los meses de Enero, Febrero, Abril y Agosto gracias a la **GROUP BY YEAR(FECHA),MONTH(FECHA)**.

B. COUNT

Tiene por misión devolver el número de elementos de un grupo. Count siempre devolverá un número numérico.

Sintaxis:

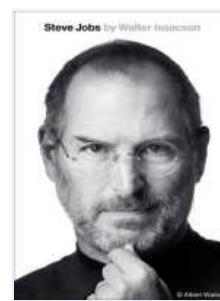
8 views

1 0

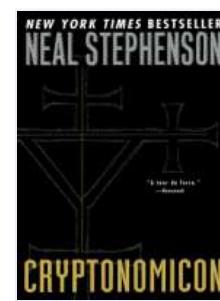
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

150 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

- **DISTINCT**: permite definir la no repetición de valores condicionados de la consulta.
- **COLUMNA**: se puede especificar el nombre de una columna de la tabla a contar.
- **Asterisco (*)**: representa a todas las filas de la tabla.
- No se puede especificar como parámetro una expresión sólo los valores expresados en la consulta, si lo hace se genera un error proveniente desde el Motor de Base de Datos.

CASO DESARROLLADO N° 3.58:

Script que permita determinar el total de pasajeros registrados. Use la función agregada COUNT

PASAJERO			
Nombre de columna	Tipo comprimido	Aceptación de valores NULL	
IDPASAJERO	char(5)	No	
NOMBRES	varchar(50)	No	
IDPAIS	char(4)	No	
TELEFONO	char(15)	No	
EMAIL	varchar(50)	No	

```
SELECT COUNT(*) AS [TOTAL PASAJEROS]
FROM PASAJERO
GO
```

En el script la función **COUNT** tiene como parámetro * que representa a cualquier columna de la tabla **PASAJERO**. El script también podría ser de la siguiente forma:

```
SELECT COUNT(IDPASAJERO) AS [TOTAL PASAJEROS]
FROM PASAJERO
GO

SELECT COUNT(NOMBRES) AS [TOTAL PASAJEROS]
FROM PASAJERO
GO
```

En ocasiones los lenguajes de programación necesitaran saber el total de registros de una tabla; si este script perteneciera a un procedimiento almacenado o una función definida podría entonces el script seria de la siguiente forma:

```
DECLARE @TOTAL INT
SELECT @TOTAL=COUNT(*)
FROM PASAJERO
```

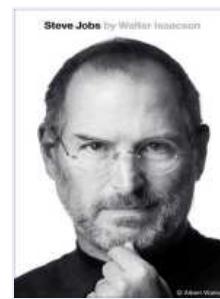
8 views

1 0

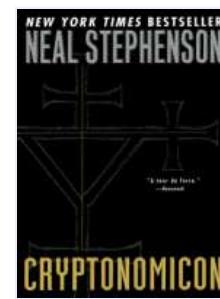
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CASO DESARROLLADO N° 3.59:

Script que permite determinar el total de pasajeros registrados agrupados por su país, teniendo en cuenta que las columnas a mostrar son NOMBRE DEL PAÍS, TOTAL DE PASAJEROS. Use la función COUNT, GROUP BY, JOIN.

PASAJERO		
Nombre de columna	Tipo comprimido	Aceptación de valores NULL
IDPASAJERO	char(5)	No
NOMBRES	varchar(50)	No
IDPAIS	char(4)	No
TELEFONO	char(15)	No
EMAIL	varchar(50)	No

```
SELECT PAI.PAIS AS [PAIS], COUNT(*) AS [TOTAL PASAJEROS]
FROM PASAJERO PAS
JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
GROUP BY PAI.PAIS
GO
```

En el script se define la primera columna con el nombre del país que se encuentra en la tabla PAÍS, tanto se tiene que añadir un **JOIN** con especificación de la tabla **PAÍS**, el conteo se realiza a través de la cláusula **COUNT**, es decir a todas las filas agrupadas en la cláusula **GROUP BY**. En la imagen siguiente se muestra el resultado de la consulta:

	PAÍS	TOTAL PASAJEROS
1	ARGENTINA	1
2	BRASIL	1
3	ECUADOR	2
4	MEXICO	1
5	PARAGUAY	1
6	PERU	3
7	URUGUAY	1

Podríamos modificar el script anterior proponiendo que sólo se muestren los países que superan la cantidad de pasajeros, el script sería como sigue:

```
SELECT PAI.PAIS AS [PAIS], COUNT(*) AS [TOTAL PASAJEROS]
FROM PASAJERO PAS
JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
GROUP BY PAI.PAIS
HAVING COUNT(*)>1
GO
```

Aquí solo se agregó la cláusula condicional **HAVING** para controlar el número de conteos resultantes.

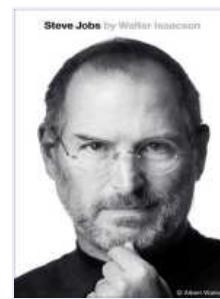
8 views

1 0

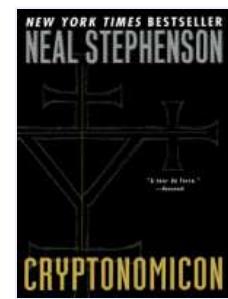
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

152 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.60:

Script que permite mostrar el total de países que aún no tienen un pasajero registrado JOIN, LEFT JOIN y la function agregada COUNT.



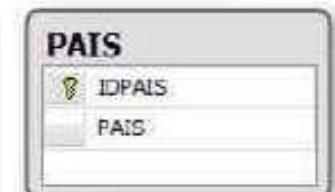
```
SELECT COUNT(*) AS [TOTAL DE PAISES]
FROM PASAJERO PAS
RIGHT JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
WHERE PAS.IDPASAJERO IS NULL
GO
```

En el script anterior se usa la cláusula **RIGHT JOIN** para filtrar los países que aún no han sido asignados a ningún pasajero. En el siguiente script se muestra la misma implementación esta vez usando **LEFT JOIN**.

```
SELECT COUNT(*) AS [TOTAL DE PAISES]
FROM PAIS PAI
LEFT JOIN PASAJERO PAS ON PAI.IDPAIS=PAS.IDPAIS
WHERE PAS.IDPASAJERO IS NULL
GO
```

CASO DESARROLLADO N° 3.61:

Script que permite mostrar el total de pasajeros y el monto acumulado de pagos por un país.



8 views

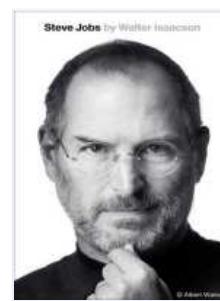
1 like

0 comments

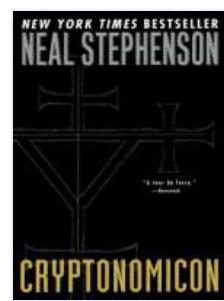
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

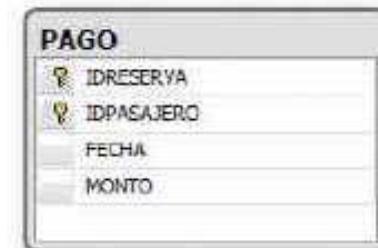
La imagen siguiente muestra el resultado de la consulta:

	PAÍS	TOTAL PASAJEROS	MONTO ACUMULADO
1	BRASIL	1	3400.00
2	ECUADOR	1	1500.00
3	PARAGUAY	1	990.00
4	PERU	3	6200.00

En el caso de BRASIL se especifica que solo hay un pasajero registrado y el monto acumula de 3400.00. Si observa PERU notara que tiene un total de 3 pasajeros y que entre ellos a valor de 6200.00.

CASO DESARROLLADO N° 3.62:

Script que permite mostrar la cantidad de pasajeros asignados por cada país y aquellos países que no registran pasajero alguno asignar el valor 0.



```

SELECT PAI.PAIS,COUNT(*) AS [TOTAL PASAJEROS]
FROM PASAJERO PAS
RIGHT JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
WHERE PAS.IDPASAJERO IS NOT NULL
GROUP BY PAI.PAIS
UNION
SELECT PAI.PAIS,0 AS [TOTAL PASAJEROS]
FROM PASAJERO PAS
RIGHT JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
WHERE PAS.IDPASAJERO IS NULL
GROUP BY PAI.PAIS
GO
    
```

En el script nuevamente se aplica el operador **UNION** que permite unir dos conjuntos de resultados con la misma cantidad de columnas. Para asignar con cero a la columna total de pasajeros a los países que no registran un pasajero es directamente sobre la consulta **SELECT PAI.PAIS, COUNT(PASAJEROS)** sólo si el IDPASAJERO de la tabla PASAJERO es nulo.

C. MAX

Esta función permite determinar el valor máximo de una expresión propuesta por el usuario.

Sintaxis:

FUNCTION AGREGADA
MAX

MAX (All o Columna o Expresion)

8 views

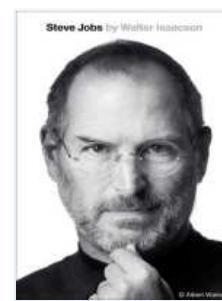
1 like

0 comments

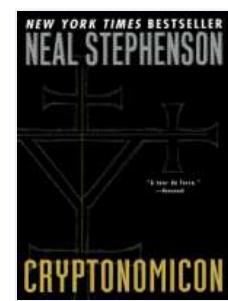
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

154 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.63:

Script que permite mostrar el monto más alto registrado en la tabla PAGO.



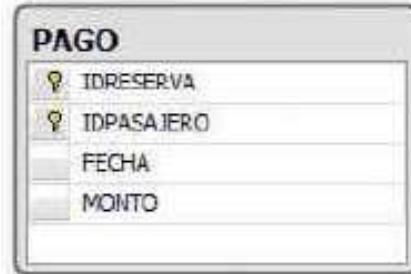
```
SELECT MAX(P.MONTO) AS [MONTO MAS ALTO]
      FROM PAGO P
      GO
```

En el script se implementa la función **MAX** que permite determinar el máximo valor ingresado en la columna **P.MONTO** que pertenece a la tabla **PAGO**.

MONTO MAS ALTO	
1	1800.00

CASO DESARROLLADO N° 3.64:

Script que permite mostrar los montos más altos por año ordenados por años en forma descendente desde la tabla PAGO.



```
SELECT YEAR(FECHA) AS [AÑO],
      MAX(P.MONTO) AS [MONTO MAS ALTO]
      FROM PAGO P
      GROUP BY YEAR(FECHA)
      ORDER BY YEAR(FECHA) DESC
      GO
```

En el script se necesita obtener el mayor monto registrado por año, la función **MAX** determina el monto más alto de acuerdo a la agrupación especificada en **GROUP BY**.

8 views

1 like

0 comments

eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)

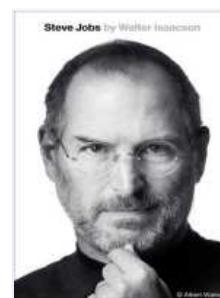
Save

Embed

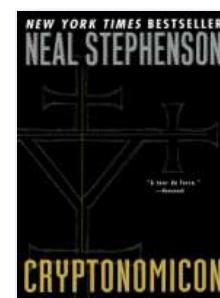
Share

Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CASO DESARROLLADO N° 3.65:

Script que permita mostrar los datos del pasajero que registra el mayor monto desde la ta

```
DECLARE @MAXIMO INT
SELECT @MAXIMO=MAX(MONTO) FROM PAGO

SELECT PAS.* 
    FROM PASAJERO PAS
    WHERE PAS.IDPASAJERO=(SELECT IDPASAJERO
                            FROM PAGO WHERE MONTO=@MAXIMO)
GO
```

En el script se declara la variable **@MAX** que permitirá obtener el mayor valor registrado en la tabla **PAGO**. Para luego ser parte de la condición de una segunda consulta. Se necesita almacenar el resultado en la variable, eso se logra con el siguiente script **SELECT @MAXIMO=MAX(MONTO)** donde el valor se asigna a la variable **@MAXIMO**.

En la segunda parte de la implementación se realiza el reporte de los pasajeros con **SELECT** si el **IDPASAJERO** es igual al **IDPASAJERO** que contiene el mayor monto desde la tabla **PAGO**, logra gracias a la subconsulta aplicada a dicha columna.

Hay que tener en cuenta que el script sólo funcionará si el mayor encontrado en la tabla es único valor, eso quiere decir que si se registran varios pagos con el más alto monto idéntico generará un error desde el motor de base de datos ya que debe recordar que toda subconsulta devolverá un valor pero en este caso ocurriría que la subconsulta devolvería más de un **1** entonces sería recomendado usar subconsultas sino **INNER JOIN**. El caso 3.66 muestra una implementación adecuada.

CASO DESARROLLADO N° 3.66:

Script que permita mostrar los datos de los pasajeros que registran el mayor y menor monto desde la tabla **PAGO**.

PASAJERO	
IDPASAJERO	
NOMBRES	
IDPAÍS	
TELEFONO	
EMAIL	

PAGO	
IDRESERVA	
IDPASAJERO	
FECHA	
MONTO	

```
DECLARE @MAXIMO INT, @MINIMO INT
SELECT @MAXIMO=MAX(MONTO),@MINIMO=MIN(MONTO) FROM PAGO
```

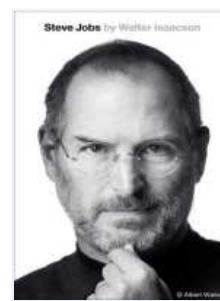
8 views

1 0

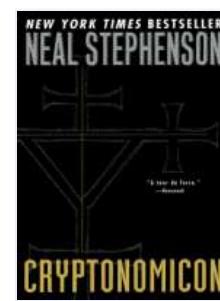
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

156 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el script se declara dos variables: una para capturar el valor máximo de los montos y uno menor. Se le asigna valores desde una misma consulta a ambas variables.

Luego se unen las dos tablas por la cláusula **JOIN** y condicionando que la columna **MONTO PAGO** sea igual al mayor asignado a la variable **@MAXIMO**.

El operador **UNION** permite unir dos consultas en una sola pero con la restricción de que los campos de ambas tablas sean de la misma cantidad.

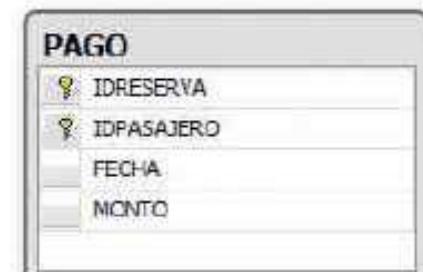
Se adicionó la columna **MONTO** proveniente de la tabla **PAGO** y la columna **CONDICIOI** exclusivamente para mostrar cuál es la condición del monto mostrado, es decir, si el resultado es el máximo o el mínimo.

La imagen siguiente muestra el resultado de la consulta:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL	MONTO	CONDICIOI
1 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM	1800.00	MONTO M
2 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM	400.00	MONTO M
3 P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM	400.00	MONTO M

CASO DESARROLLADO N° 3.67:

Script que permita mostrar los datos del país que registra el mas alto número de pasajeros. Use funciones agregadas y subconsultas.



```

DECLARE @MAXIMO INT
SELECT @MAXIMO=MAX(TOTAL)
    FROM (SELECT PAI.PAIS,COUNT(*) AS TOTAL
          FROM PASAJERO PAS
          JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
          GROUP BY PAI.PAIS
         ) X

SELECT PAI.PAIS,COUNT(*) AS TOTAL
    FROM PASAJERO PAS
    JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
  
```

8 views

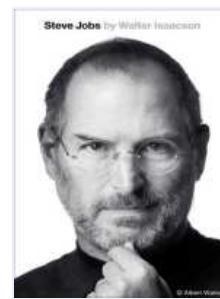
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

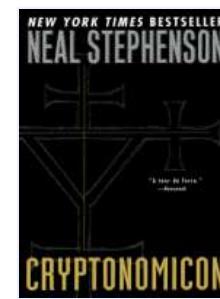
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

```
SELECT PAI.PAIS,COUNT(*) AS TOTAL
      FROM PASAJERO PAS
      JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
     GROUP BY PAI.PAIS
```

	PAIS	TOTAL
1	ARGENTINA	1
2	BRASIL	1
3	ECUADOR	2
4	MEXICO	1
5	PARAGUAY	1
6	PERU	3
7	URUGUAY	1

Como notará el listado obtenido muestra que en el país **PERÚ** está el mayor total de pasajeros, tanto ese valor deberá ser enviado a la variable **@MAXIMO**. Para poder capturar el máximo conteo como lo mostrado seguro se pensará en una expresión parecida a la siguiente **MAX(COUNT(MONTO))** pero en ninguno de los casos hay un resultado positivo ya que no se superponen funciones agregadas ya que se rompería su sintaxis. Para este caso se necesita a partir de una consulta donde la cabecera definida en la consulta será considerada como tabla trate en lo posible no colocar columnas con espacios en blanco. Para crear una tabla se especifica de la siguiente manera:

```
FROM (SELECT PAI.PAIS,COUNT(*) AS TOTAL
      FROM PASAJERO PAS
      JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
     GROUP BY PAI.PAIS
) X
```

Entonces X es el nombre que se le asigna a la consulta especificada en la cláusula **FROM**, si se tienen dos columnas en esta consulta por tanto X tiene las siguientes columnas **PAIS** y **TOTAL**. Esto nos ayudará a encontrar el mayor de la consulta gracias a la nueva columna **TOTAL** y colocando el script **SELECT @MAXIMO=MAX(TOTAL)** como notará la función **MAX** es aplicada a la columna **TOTAL** proveniente de la tabla X.

El segundo **SELECT** es para listar los países cuya condición sea que el número total de pasajeros sea igual al valor asignado a **@MAXIMO**.

Ahora presentamos un script compactado del mismo caso pero en una sola expresión, con un poco extenso pero tiene las mismas especificaciones realizadas en el script anterior:

```
SELECT PAI.PAIS,COUNT(*) AS TOTAL
      FROM PASAJERO PAS
      JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
     GROUP BY PAI.PAIS
HAVING COUNT(*)=@MAXIMO
```

8 views

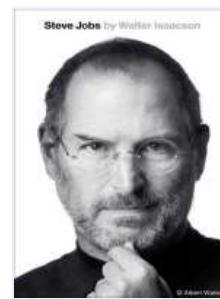
1 like

0 comments

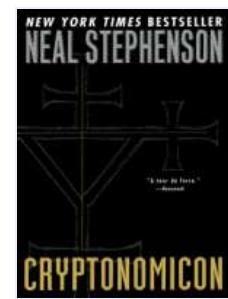
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

158 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En ambos casos la imagen de la respuesta es la misma:

PAÍS	TOTAL
PERU	3

D. MIN

Esta función permite determinar el valor mínimo de una expresión propuesta por el usuario.

Sintaxis:

FUNCIÓN AGREGADA

MIN

MIN (All o Columna o Expresión)

Donde:

- **ALL:** al especificarlo todos los valores son evaluados. Muchas veces no se coloca ALL como parámetro predeterminado.
- **COLUMNA:** se puede especificar el nombre de una columna de la tabla a buscar el menor valor.
- **Expresión:** representa a una función SQL o a un juego de operadores aritméticos.
- **Min:** sólo puede ser usado con columnas o expresiones que tengan como tipo un entero. Si se usa con otra contraria genera un error proveniente desde el Motor de Base de Datos.

CASO DESARROLLADO N° 3.68:

Un script que permita mostrar el costo mínimo registrado en la tabla RESERVA.

RESERVA		
IDRESERVA		
COSTO		
FECHA		

```
SELECT MIN(COSTO) AS [COSTO MINIMO]
FROM RESERVA
GO
```

En el script se usa la función MIN que permite obtener el menor valor ingresado a la columna COSTO de la tabla RESERVA.

CASO DESARROLLADO N° 3.69:

8 views

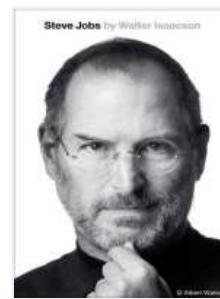
1 like

0 comments

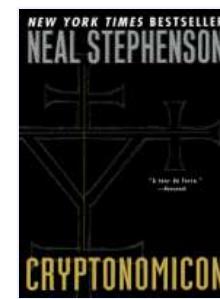
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

```

SELECT PAI.PAIS,COUNT(*) AS TOTAL
    FROM PASAJERO PAS
    JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
    GROUP BY PAI.PAIS
    HAVING COUNT(*)=(

        SELECT MIN(TOTAL)
            FROM (SELECT PAI.PAIS,COUNT(*) AS TOTAL
                FROM PASAJERO PAS
                JOIN PAIS PAI
                ON PAS.IDPAIS=PAI.IDPAIS
                GROUP BY PAI.PAIS
            ) X
    )
GO

```

En el script se busca el mínimo valor registrado en la consulta PAIS y TOTAL desde aquí se calcula el conteo de la agrupación desde la tabla PASAJERO. Para poder entender mejor el script véase caso desarrollado Nº. 3.67.

E. AVG

Devuelve el promedio de los valores de un determinado grupo considere que dicha columna es numérica. Los valores NULL no son considerados por el motor de base de datos.

Sintaxis:

**FUNCTION AGREGADA
AVG**

AVG (Columna o Expresión)

Donde:

- **COLUMNA:** se puede especificar el nombre de una columna para determinar el promedio.
- **Expresión:** representa a una función SQL o a un juego de operadores aritméticos. No se incluyen las funciones de agregado ni subconsultas.

CASO DESARROLLADO N° 3.70:

Script que permita mostrar el monto promedio agrupados por países.

PAÍS	
IDPAIS	
PAIS	

PAGO	
IDRESERVA	
IDPASAJERO	
FECHA	
MONTO	

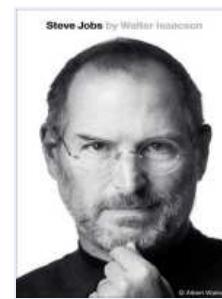
8 views

1 0

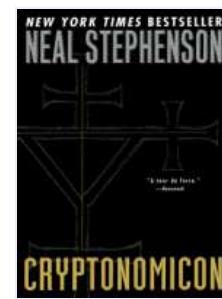
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

160 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el script se promedia los montos de la tabla PAGO agrupados por PAIS. Hay que tener en cuenta que según el diagrama de base de datos no tienen relación de asociación las tablas PAIS tiene que invocar a la tabla PASAJERO, pero sólo como un medio para determinar el promedio. Dado que desde esta tabla no necesitamos ninguna columna. La imagen a continuación muestra el resultado de la consulta:

PAÍS	MONTO PROMEDIO
1 BRASIL	850.00
2 ECUADOR	750.00
3 PARAGUAY	495.00
4 PERÚ	1240.00

Ahora debemos comprobar que el promedio en los países es el correcto, analizaremos el país PERÚ donde 1240 es el monto promedio a evaluar, para eso usamos el siguiente script:

```
SELECT AVG(MONTO) AS [MONTO PROMEDIO]
FROM PAGO PAG
JOIN PASAJERO PAS ON PAG.IDPASAJERO=PAS.IDPASAJERO
JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
GROUP BY PAI.PAIS
HAVING PAI.PAIS='PERU'
GO
```

O en su defecto usar el siguiente (ambos comprueban correctamente el valor promedio del país PERÚ)

```
SELECT AVG(MONTO) AS [MONTO PROMEDIO]
FROM PAGO
WHERE IDPASAJERO='P0001'
OR IDPASAJERO='P0002'
OR IDPASAJERO='P0005'
```

Se ha comprobado que los pasajeros con IDPASAJERO P0001, P0002 y P0005 son del país PERÚ tanto, su promedio sería como lo muestra la siguiente imagen:

MONTO PROMEDIO	
1	1240.00

3.10. AGREGAR CONJUNTO DE RESULTADOS: UNION

El operador **UNION** permite mostrar los resultados de varias consultas **SELECT** en una sola fila. Para el uso de este operador es que ambas tablas deben tener las mismas columnas o específicamente:

8 views

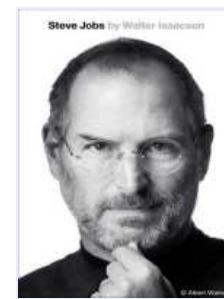
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

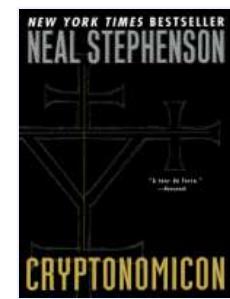
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CASO DESARROLLADO N° 3.71:

Script que permite mostrar los registros de la tabla pasajeros y país en una misma consulta.



```

SELECT PAS.IDPASAJERO, PAS.NOMBRES
  FROM PASAJERO PAS
UNION
SELECT PAI.IDPAIS, PAI.PAIS
  FROM PAIS PAI
GO
  
```

En el script se imprimen la columna IDPASAJERO y NOMBRE de los pasajeros mientras que la que uniremos en registros es IDPAIS y PAIS hasta aquí no se rompe la regla sintáctica de UNION, como notará los registros no son compatibles pero los une, como lo muestra la imagen:

	IDPASAJERO	NOMBRES
1	0001	PERU
2	0002	ARGENTINA
3	0003	CHILE
4	0004	ECUADOR
5	0005	BRASIL
6	0006	VENEZUELA
7	0007	PARAGUAY
8	0008	URUGUAY
9	0009	BOLIVIA
10	0010	MEXICO
11	0011	HONDURAS
12	0012	EEUU
13	0013	PUERTO RICO
14	P0001	ANGELA TORRES LAZARO
15	P0002	FERNANDA TORRES LAZARO
16	P0003	MARIA ZAMORA MEJIA
17	P0004	GUADALUPE ACOSTA FERRER
18	P0005	LIZ LAZARO MENOR
19	P0006	KARLA GALLEGOS SILVA
20	P0007	NERY CALLE DE LA CRUZ
21	P0008	HEIDI RENGIFO REATEGUI
22	P0009	MARISOL DIAZ ZAMBRANO
23	P0010	LINDA TUME VARAS

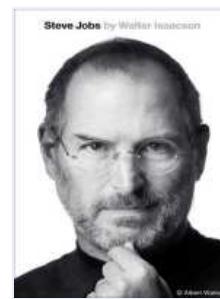
8 views

1 0

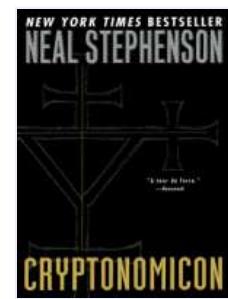
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

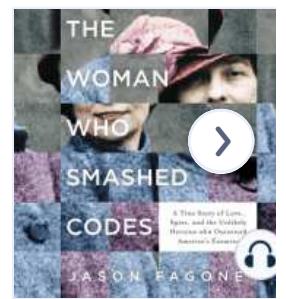
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

162 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.72:

Script que permite mostrar el total de registros de las tablas PASAJERO, PAIS y PAGO desde consulta.



```

SELECT 'PASAJERO' AS [TABLA],
       COUNT(*) AS [TOTAL]
     FROM PASAJERO
UNION
SELECT 'PAIS' AS [TABLA],
       COUNT(*) AS [TOTAL]
     FROM PAIS
UNION
SELECT 'RESERVA' AS [TABLA],
       COUNT(*) AS [TOTAL]
     FROM RESERVA
GO
  
```

En el script se implementa tres consultas consecutivas todas con dos columnas especiales, la primera muestra el nombre de la tabla mientras que la siguiente muestra el total de registros que contiene dicha tabla.

	TABLA	TOTAL
1	PAIS	13
2	PASAJERO	10
3	RESERVA	15

3.11. RESUMEN DE DATOS: OPERADOR CUBE Y ROLLUP

Genera filas de agregado mediante la cláusula GROUP BY, este genera una agrupación para permutaciones o quiebres de una determinada expresión o lista de columnas.

Sintaxis:
**OPERADOR
CUBE**

8 views

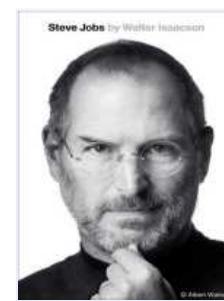
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

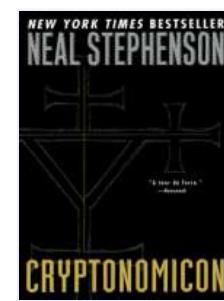
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon

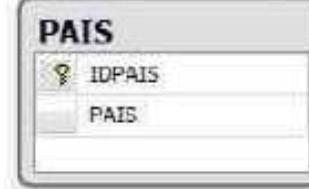


The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

CASO DESARROLLADO N° 3.73:

Script que permita mostrar el total de registros por años mostrando los países involucrados en la consulta. Use el operador CUBE.



```

SELECT PAI.PAIS, YEAR(PAG.FECHA) AS [AÑO DE REGISTRO], COUNT(*) AS TOTAL
FROM PAGO PAG
JOIN PASAJERO PAS ON PAS.IDPASAJERO=PAG.IDPASAJERO
JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
GROUP BY CUBE(PAI.PAIS,YEAR(PAG.FECHA))
GO

```

En el script se muestran las columnas PAIS, el año de las fechas de pago y el número total agrupados. El resultado se muestra en la imagen siguiente:

	PAIS	AÑO DE REGISTRO	TOTAL
1	BRASIL	2011	1
2	ECUADOR	2011	1
3	PARAGUAY	2011	1
4	PERU	2011	3
5	NULL	2011	6
6	BRASIL	2012	3
7	ECUADOR	2012	1
8	PARAGUAY	2012	1
9	PERU	2012	2
10	NULL	2012	7
11	NULL	NULL	13
12	BRASIL	NULL	4
13	ECUADOR	NULL	2
14	PARAGUAY	NULL	2
15	PERU	NULL	5

Como notará existe una fila por cada cambio de año donde muestra el total de registros de final de los registros muestra el total acumulado en este caso 13 se deriva de la fila 5 (total de la fila 10 (total del 2012)). Además de mostrarle el acumulado por país que también sumar

Si desea sólo visualizar los resúmenes, entonces deberá ejecutar el script de la siguiente forma:

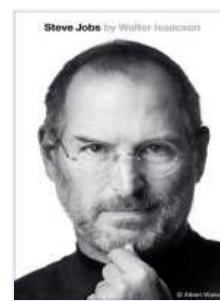
8 views

1 0

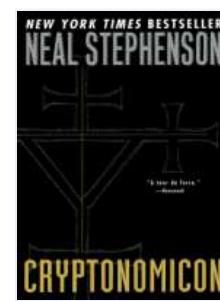
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

164 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.74:

Script que permite mostrar el total de registros por años mostrando los países involucrados en la compra. Use el operador RollUP.



```
SELECT PAI.PAIS, YEAR(PAG.FECHA) AS [AÑO DE REGISTRO], COUNT(*) AS TOTAL
FROM PAGO PAG
JOIN PASAJERO PAS ON PAS.IDPASAJERO=PAG.IDPASAJERO
JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
GROUP BY PAI.PAIS, YEAR(PAG.FECHA)
WITH ROLLUP
GO
```

En el script se implementa la agrupación desde el nombre del país y el año obtenido desde la columna **FECHA** de la tabla **PAGO**. Esta vez se agrega el operador **WITH ROLLUP** para poder obtener el total de los países y sus años respectivos. La imagen muestra el resultado de la consulta:

	PAIS	AÑO DE REGISTRO	TOTAL
1	BRASIL	2011	1
2	BRASIL	2012	3
3	BRASIL	NULL	4
4	ECUADOR	2011	1
5	ECUADOR	2012	1
6	ECUADOR	NULL	2
7	PARAGUAY	2011	1
8	PARAGUAY	2012	1
9	PARAGUAY	NULL	2
10	PERU	2011	3
11	PERU	2012	2
12	PERU	NULL	5
13	NULL	NULL	13

3.12. DECLARACIÓN MERGE

Realiza operaciones de inserción, actualización o eliminación en una tabla de destino según la combinación con una tabla de origen. Por ejemplo, puede sincronizar dos tablas actualizando o eliminando las filas de una tabla según las diferencias que se encuentren en la otra.

Sintaxis:**INTRUCCION**

8 views

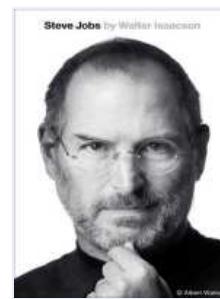
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

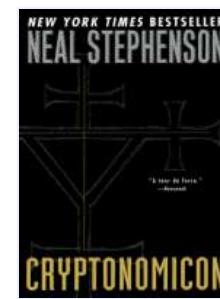
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

Donde:

- **Tabla_destino**

Se asigna la tabla destino de donde provienen los valores y se comparan con las filas de la según la condición especificada en la cláusula ON. Esta definición permite realizar operación, actualización o eliminación especificadas en las cláusulas **WHEN** de la instrucción.

- **USING <tabla_origen>**

Especifica el origen de datos que se hace coincidir con las filas de datos en target_table <merge_search condition>. El resultado de esta coincidencia dicta las acciones que emplea las cláusulas WHEN de la instrucción MERGE. <table_source> puede ser una tabla remota derivada que tengan acceso a las tablas remotas.

- **ON <condición_de_busqueda>**

Especifica las condiciones en las que <table_source> se combina con target_table para donde coinciden.

- **WHEN MATCHED THEN**

Especifica que todas las filas de target_table que coinciden con las filas que devuelve <table_source> ON <condición_de_busqueda> y que satisfacen alguna condición de búsqueda adicional se actualizan o eliminan según la cláusula <merge_matched>. La instrucción MERGE puede tener al menos una cláusula WHEN MATCHED. Si se especifican dos cláusulas, la primera debe ir acompañada de una cláusula <condición_de_busqueda>. Para una fila determinada, la segunda cláusula WHEN MATCHED solo se aplica si no se aplica la primera. Si hay dos cláusulas WHEN MATCHED, una debe especificar una acción UPDATE y la otra una acción DELETE. Si se especifica UPDATE en la cláusula <merge_matched> y más de una fila de <table_source> coincide con una fila en target_table según la <condición_de_busqueda>, SQL Server devuelve un error. La instrucción MERGE no puede actualizar la misma fila más de una vez, ni actualizar o eliminar la misma fila.

- **WHEN NOT MATCHED [BY TARGET] THEN**

Especifica que una fila se inserta en target_table para cada fila que devuelve <table_source> <merge_search_condition> que no coincide con una fila de target_table, pero satisface una condición de búsqueda adicional si está presente. La cláusula <merge_not_matched> especifica qué tipo de fila se inserta. La instrucción MERGE puede tener sólo una cláusula WHEN NOT MATCHED.

- **WHEN NOT MATCHED BY SOURCE THEN**

Especifica que todas las filas de target_table que no coinciden con las filas que devuelven <table_source> ON <merge_search_condition> y que satisfacen alguna condición de búsqueda se actualizan o eliminan según la cláusula <merge_matched>.

La instrucción MERGE puede tener a lo sumo dos cláusulas WHEN NOT MATCHED BY SOURCE. Si se especifican dos cláusulas, la primera debe ir acompañada de una cláusula AND <cláusula>.

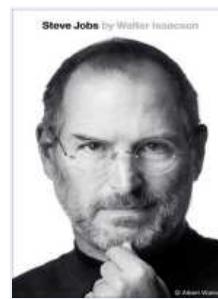
8 views

1 0

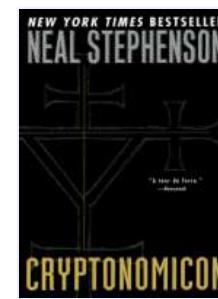
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

166 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Consideración para el MERGE:

- Toda instrucción Merge debe contar con una de las tres cláusulas **MATCHED** no importe el orden. Considere que una variable no puede actualizarse más de una vez en la misma cláusula **MA**
- Si la tabla destino tiene una inserción, actualización o eliminación esta estará limitada por restricciones definidas en la misma, incluidas las restricciones de integridad referencial
- Para finalizar una instrucción **MERGE** se requiere un punto y coma (;) como terminador, si no se coloca sin punto y coma genera un error desde el motor de base de datos.
- Despues de ejecutar una instrucción **MERGE** puede realizar una consulta a la variable **ROWCOUNT** para devolver el número total de filas insertadas, actualizadas o eliminadas descritas

CASO DESARROLLADO N° 3.75:

Script que permite insertar un registro en la tabla PASAJERO en la cual se evalúe el valor ingresado en el campo IDPAIS. En caso el pasajero se encuentre registrado debe actualizar sus datos, caso contrario registra un nuevo registro. Use MERGE.



Inicialmente los registros de la tabla **PASAJERO** se muestran como la imagen siguiente:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	001	999999999	ATORRES@HOTMAIL.COM
2 P0002	FERNANDA TORRES LAZARO	001	999999999	FTORRES@HOTMAIL.COM
3 P0003	MARIA ZAMORA MEJIA	005	957564526	MZAMORA@GMAIL.COM
4 P0004	GUADALUPE ACOSTA FERRER	002	957564526	GACOSTA@HOTMAIL.COM
5 P0005	LULU LAZARO MENOR	001	999999999	LLAZARO@GMAIL.COM
6 P0006	KARLA GALLEGOS SILVA	007	957564526	KGALLEGOS@HOTMAIL.COM
7 P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8 P0008	HEIDI RENGIFO REATEGUI	004	957564526	HRENGIFO@HOTMAIL.COM
9 P0009	MARISOL DIAZ ZAMBRANO	004	957564526	MDIAZ@GMAIL.COM
10 P0010	LINDA TUME VARAS	008	957564526	LTUME@HOTMAIL.COM

```
--1.-
DECLARE @IDPAS CHAR(5)=>'P0010',
        @NOM VARCHAR(40)=>'GUADALUPE ACOSTA FERRER',
        @IDPAI CHAR(4)=>'0003', @FONO CHAR(15)=>'938475747',
        @EMA VARCHAR(40)=>'GACOSTA@GMAIL.COM'

--2.-
MERGE PASAJERO AS TARGET
USING (SELECT @IDPAS,@NOM,@IDPAI,@FONO,@EMA) AS
      SOURCE(IDPASAJERO.NOMBRES.IDPAIS.TELEFONO.EMAIL)
```

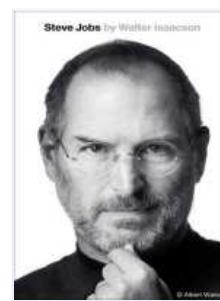
8 views

1 0

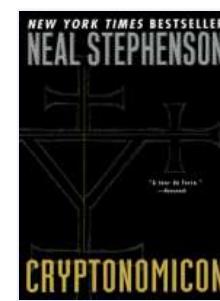
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

Definiremos punto a punto la implementación de la instrucción **MERGE**:

Punto 1: se declaran variables locales que tienen por misión asignar con valores de un registro dentro de la tabla **PASAJERO**. En este caso la cantidad de variables es la misma cantidad de columnas de la tabla **PASAJERO**.

Punto 2: En la instrucción **MERGE** se debe definir la tabla destino (**TARGET**) con la siguiente sintaxis: **MERGE PASAJERO AS TARGET**. Luego se debe asignar los valores de la tabla fuente (**SOURCE**) a las variables locales declaradas en el Punto 1, para asignar dichas variables se debe colocar **USING** (@IDPAS,@NOM,@IDPAI,@FONO,@EMA) **AS SOURCE(IDPASAJERO,NOMBRES,IDPAIS,TELEFONO,EMAIL)**. Finalmente se condiciona la unión de las tablas, es decir, la fuente y el destino por la columna **IDPASAJERO** recuerde que la tabla destino ya tiene un valor asignado por tanto se hace la siguiente condición **ON (TARGET.IDPASAJERO = SOURCE.IDPASAJERO)**.

Punto 3: para poder determinar si el registro asignado a las variables locales se encuentra dentro de la tabla se implementa con **WHEN MATCHED THEN** en el caso que **IDPASAJERO** encontrado en la tabla fuente se debe actualizar los valores con **UPDATE SET IDPASAJERO=SOURCE.IDPASAJERO, NOMBRES=SOURCE.NOMBRES, IDPAIS=SOURCE.IDPAIS, TELEFONO=SOURCE.TELEFONO, EMAIL=SOURCE.EMAIL**.

WHEN NOT MATCHED THEN en el caso que **IDPASAJERO** no se encuentre registrado en la tabla se inserta en la tabla **PASAJERO**, es decir, el origen enviará a la tabla asignada como destino en el **MERGE** los nuevos valores usando **INSERT INTO IDPASAJERO, SOURCE.NOMBRES, SOURCE.IDPAIS, SOURCE.TELEFONO, SOURCE.EMAIL**; no se olvide que por ser la línea final del **MERGE** se tiene que colocar el símbolo punto y coma para finalizar la instrucción.

En el script se asigna valores para el pasajero con código **P0010** el cual ya existe en dicha tabla, se tendrá que determinar si actualizará o insertará dicho registro.

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4 P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5 P0005	Luz Lazaro Menor	0001	999999999	LLAZARO@GMAIL.COM
6 P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7 P0007	NEY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8 P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
9 P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
10 P0010	GUADALUPE ACOSTA FERRER	0003	938475747	GACOSTA@GMAIL.COM

Para probar la implementación sólo cambie los valores asignados a las variables locales de la manera:

8 views

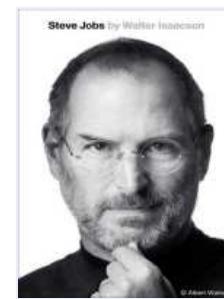
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

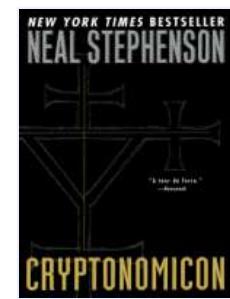
Uploaded by [anon_61286589](#)[Full description](#)

 Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

168 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 3.76:Script que permita replicar de la tabla llamada **PAISNUEVO** a la tabla **PAIS**. Use el operadoInicialmente los registros de la tabla **PAIS** se muestra como la imagen siguiente:

	COD_PAIS	PAIS
1	0001	PERU
2	0002	ARGENTINA
3	0003	CHILE
4	0004	ECUADOR
5	0005	BRASIL
6	0006	VENEZUELA
7	0007	PARAGUAY
8	0008	URUGUAY
9	0009	BOLIVIA
10	0010	MEXICO
11	0011	HONDURAS
12	0012	EEUU
13	0013	PUERTO RICO

Se tiene que crear la nueva tabla con igual estructura que la tabla PAIS. Ejecute el siguiente

```

CREATE TABLE PAISNUEVO(
    COD_PAIS           CHAR(4)          NOT NULL PRIMARY KEY,
    PAIS               VARCHAR(30)       NOT NULL
)
GO

INSERT INTO PAISNUEVO
VALUES ('A001','JAPON'),
       ('A002','INDIA'),
       ('A003','CHINA'),
       ('A004','COREA DEL SUR'),
       ('A005','SINGAPUR'),
       ('A006','ARMENIA'),
       ('A007','NEPAL')
GO
  
```

Ahora ejecutaremos el script que permitirá enviar los registros de la tabla PAISNUEVO a la

8 views

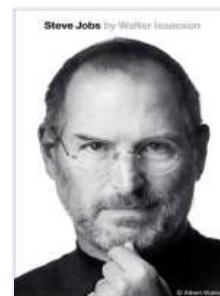
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

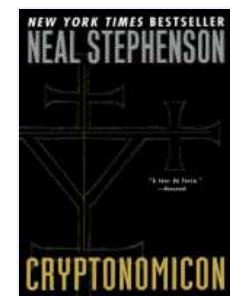
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

CAP. 3: LENGUAJE DE MANIPULACIÓN DE DATOS (DM)

En el script se define la tabla destino a **PAIS** en la cual se insertarán los países específicos de la tabla fuente con **MERGE PAIS AS TARGET**, la tabla fuente de donde provienen los valores se especifica con **USING** para esto se especifica las columnas de la tabla fuente de donde provienen los valores con **USING (SELECT COD_PAIS,PAIS FROM PAISNUEVO) AS SOURCE**, seguidamente se tiene que se especifique el campo de unión entre las dos tablas (origen-destino) con la palabra **ON (TARGET.IDPAIS = SOURCE.COD_PAIS)**, en caso las filas de la tabla **PAISNUEVO** no se encuentren en la tabla **PAIS** se insertarán como registros nuevos.

Además, se usa el operador **OUTPUT** para imprimir una consulta de los registros invocados por la implementación **MERGE**. \$action imprime que tipo de instrucción se ejecutó, insertando todos los registros insertados en el proceso. La imagen siguiente muestra la ejecución de la **MERGE**:

	Caption	IDPAIS	PAIS	IDPAIS	PAIS
1	INSERT	A001	JAPON	NULL	NULL
2	INSERT	A002	INDIA	NULL	NULL
3	INSERT	A003	CHINA	NULL	NULL
4	INSERT	A004	COREA DEL SUR	NULL	NULL
5	INSERT	A005	SINGAPUR	NULL	NULL
6	INSERT	A006	ARMENIA	NULL	NULL
7	INSERT	A007	NEPAL	NULL	NULL

Como notará en la primera columna se muestra el tipo de acción realizada por el **MERGE**. Adelante se verá con otro caso que también puede imprimir el mensaje de eliminación: así lo implementa. En el listado los valores **NULL** indican que no hubo inserción en dicha fila. Finalmente, se hace una consulta a la tabla **PAIS** para verificar que los registros de la tabla **PAISNUEVO** pasen a los registros de la tabla **PAIS**. Como se muestra en la imagen siguiente:

	COD_PAIS	PAIS
1	0001	PERU
2	0002	ARGENTINA
3	0003	CHILE
4	0004	ECUADOR
5	0005	BRASIL
6	0006	VENEZUELA
7	0007	PARAGUAY
8	0008	URUGUAY
9	0009	BOLIVIA
10	0010	MEXICO
11	0011	HONDURAS
12	0012	EEUU
13	0013	PUERTO RICO
14	A001	JAPON
15	A002	INDIA
16	A003	CHINA
17	A004	COREA DEL SUR
18	A005	SINGAPUR
19	A006	ARMENIA
20	A007	NEPAL

8 views

1 0

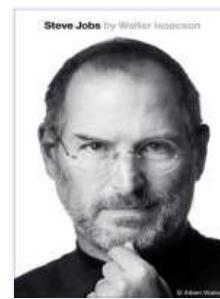
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

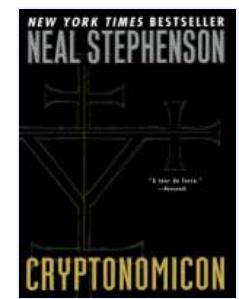
[Full description](#)

   
Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

8 views

1 like

0 comments

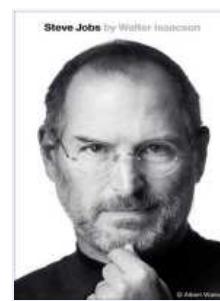
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

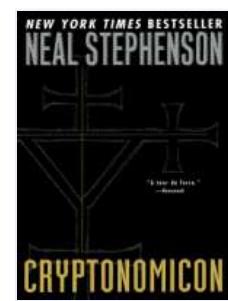
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics



Programación Transact SQL

CAPACIDAD:

El lector podrá entender e implementar script que involucren objetos de base de datos a través de la programación básica, intermedia y avanzada.

Primero, se reconocerá como se trabaja con el lenguaje Transact-SQL realizando casos descriptivos sencillos, luego pasaremos por las estructuras de control como las selectivas y las repetitivas acompañando de casos de media complejidad.

En la mitad del capítulo implementaremos las funciones y procedimientos como base fundamental para la programación en SQL para esto controlaremos los errores dentro de las instrucciones ejecutadas, implementaremos reportes simples con el uso de los cursor y finalizaremos implementando triggers con casos relativamente complejos.

CONTENIDO:

- *Introducción*
- *Fundamentos de Programación TRANSACT SQL*
- *VARIABLES, IDENTIFICADORES*
- *Funciones CAST y CONVERT*
- *Estructuras de Control*
- *Estructura selectiva IF*
- *Estructura condicional múltiple CASE*
- *Estructura de control WHILE*
- *Ejecución de Scripts en TRANSACT SQL*
- *Procedimientos almacenados definidos por el usuario*
- *Procedimientos almacenados con parámetros de entrada*
- *Procedimientos almacenados con parámetros de entrada y salida*
- *Modificar la implementación de un procedimiento almacenado*
- *Eliminar procedimientos almacenados*

8 views

1 0

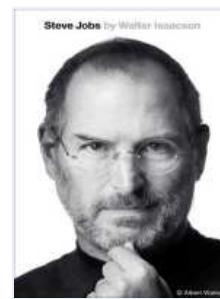
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

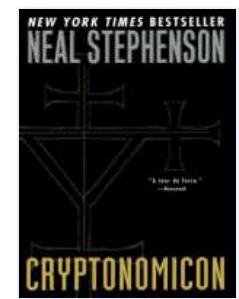
[Full description](#)

   
Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who
Smashed
Codes: A Tru

8 views

1 like

0 comments

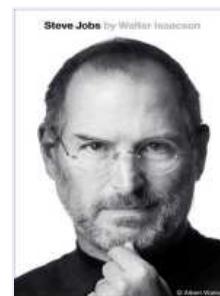
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

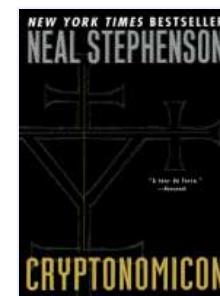
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

4.1. INTRODUCCIÓN

Transact SQL es el lenguaje de programación que proporciona Microsoft SQL Server para SQL estándar con otro tipo de instrucciones y elementos propios de los lenguajes de programación que esta parte está limitada en SQL.

Características:

- No tiene sensibilidad entre mayúsculas y minúsculas.
- Se pueden incluir comentarios de una línea con -- como la de varias líneas con /* */
- El usuario no puede definir variables globales ya que estas son propias de SQL Server identificadas con @@variableGlobal.
- En el caso de variables locales se usan el operador DECLARE y siempre se empieza con @.
- Al referirnos a un script en este libro hacemos referencia a un conjunto de sentencias T-SQL en texto plano que se ejecutan dentro del servidor de SQL Server. Algunas veces es llan

Con Transact SQL se puede programar usando las siguientes unidades de SQL Server:

- Procedimientos almacenados
- Funciones
- Triggers
- Scripts

4.2. FUNDAMENTOS DE PROGRAMACIÓN TRANSACT SQL

Transact SQL amplia las sentencias SQL con una serie de extensiones que resultan de gran utilidad en la implementación de sentencias. Si no toma en consideración la programación Transact SQL, que seguir usando instrucciones simples y la lógica condicional o repetitiva la tendrá que realizar en un lenguaje de programación asociada a SQL Server.

4.3. VARIABLES, IDENTIFICADORES

En Transact SQL se pueden declarar variables de tipo local ya que las globales que están impuestas tienen como propietario a SQL Server. La visibilidad de dichas variables es limitada por las unidades de programación, es decir, sólo será accesible dentro de un proceso de lotes, un procedimiento almacenado, una función, etc.

Características de las variables:

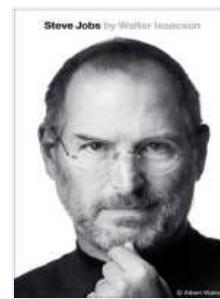
8 views

1 0

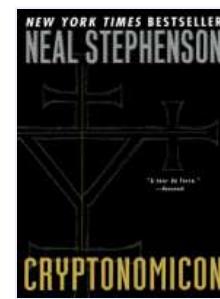
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

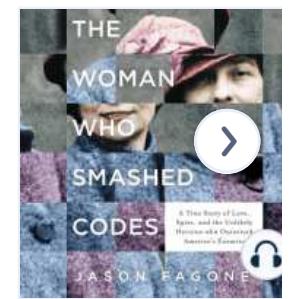
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

174 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Sintaxis para DECLARE:

**TRANSACT SQL
DECLARE**
DECLARE @nombreVariable TipoDatos

Donde:

- **@nombreVariable:** es el nombre de la variable a emplear; debe empezar con @ obligatoriamente ya que al no colocarlo SQL lo interpretará como una columna de una tabla.
- **tipoDatos:** es el ámbito de la variable, es decir, la capacidad que tendrá dicha variable cc números, etc.

Sintaxis para SET:

**TRANSACT SQL
SET**
SET @nombreVariable = Valor

Donde:

- **@nombreVariable:** es el nombre de la variable previamente declarada con el operador **DECLARE**.
- **Valor:** es el dato asignado a la variable considere que dicho valor debe ser del mismo tipo que se declaró. Tenga en cuenta que de acuerdo al tipo de datos se debe asignar por ejemplo, para caracteres o fechas colocarlo entre dos comillas simples y en los números entre dos comillas dobles.

CASO DESARROLLADO N° 4.1

Script que permita calcular el promedio de 4 notas de un determinado alumno, dichas no estar inicializadas. Use variables locales.

```

'1.
DECLARE @N1 INT,@N2 INT,@N3 INT,@N4 INT,@PROMEDIO DECIMAL(5,2)
SET @N1=12
SET @N2=20
SET @N3=15
SET @N4=18

'2.
SET @PROMEDIO=(@N1+@N2+@N3+@N4)/4.0

'3.
PRINT '*** RESUMEN DE NOTAS ***'
PRINT ''
PRINT 'LA NOTA 1: '+CAST(@N1 AS CHAR(2))

```

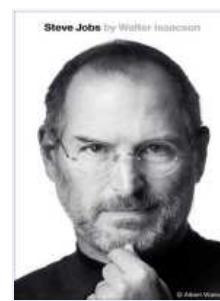
8 views

1 0

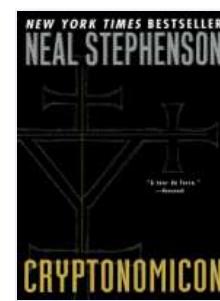
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

En el punto uno se declaran las variables n1,n2,n3,n4 y el promedio en este último se tipo DECIMAL(5,2) para que el promedio de como resultado un valor decimal de 2 cifras c enteros posibles, esto es especificado con el numero 5; por ejemplo, 99.99 el punto considerado dentro de la declaración de capacidad de la variable en total son 4 nueves y eso se asignó 5; de esos cinco se toma dos para los decimales por eso la asignación 2.

En el punto dos se calcula el promedio de notas aplicando una expresión simple de com operadores aritméticos como + , / y () como notará el número 4 que se aplica en el pro colocó 4.0 para poder determinar los decimales correctos a la expresión.

Probamos con dos modelos de la misma expresión y las respuestas son diferentes para veamos el primer caso SET @PROMEDIO=(@N1+@N2+@N3+@N4)/4 el resultado es 1 expresión sería SET @PROMEDIO=(@N1+@N2+@N3+@N4)/4.0 el resultado sería 16.25.

En el punto tres se está imprimiendo el reporte a nivel de consola, no olvide que las variable como numéricos tienen que ser convertidos con CAST para poder ser impresos en la funci

Mostraremos una segunda versión del mismo script, pero con menos líneas de implement

```
DECLARE @N1 INT=12,@N2 INT=20,@N3 INT=15,@N4 INT=18,@PROMEDIO DECIMAL(5,2)

SET @PROMEDIO=(@N1+@N2+@N3+@N4)/4

PRINT '*** RESUMEN DE NOTAS ***'
PRINT ''
PRINT 'LA NOTA 1: '+CAST(@N1 AS CHAR(2))
PRINT 'LA NOTA 2: '+CAST(@N2 AS CHAR(2))
PRINT 'LA NOTA 3: '+CAST(@N3 AS CHAR(2))
PRINT 'LA NOTA 4: '+CAST(@N4 AS CHAR(2))
PRINT ''
PRINT 'EL PROMEDIO ES: '+CAST(@PROMEDIO AS CHAR(5))
GO
```

En este caso la declaración de las variables locales y la asignación se realiza en una sola acción @N1 INT=12 donde **@N1** es de tipo **INT** y asignado con el valor **12**.

En ambos casos el resultado es el mismo y se muestra en la siguiente imagen:

```
*** RESUMEN DE NOTAS ***

LA NOTA 1: 12
LA NOTA 2: 20
LA NOTA 3: 15
LA NOTA 4: 18
```

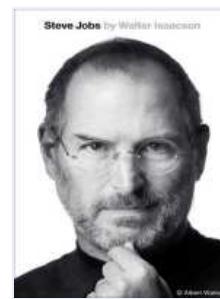
8 views

1 0

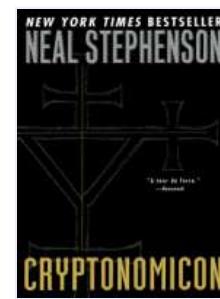
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

176 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 4.2

Script que permita mostrar los pasajeros de un determinado país. Use una variable local como valor el nombre del país a buscar.

PASAJERO	
IDPASAJERO	
NOMBRES	
IDPAIS	
TELEFONO	
EMAIL	

PAIS	
Column Name	Condensed Type
IDPAIS	char(4)
PAIS	varchar(30)

```

DECLARE @PAIS VARCHAR(40)='PERU'

SELECT PAS.* 
    FROM PASAJERO PAS
    WHERE PAS.IDPAIS=(SELECT IDPAIS
        FROM PAIS
        WHERE PAIS.PAIS=@PAIS
    )
GO

```

En el script se declara la variable @PAIS asignándole el valor PERU para que sea parte de dentro de una subconsulta y así mostrar los pasajeros que cumplan con dicha condición.

CASO DESARROLLADO N° 4.3

Script que permita mostrar el país de donde proviene un determinado cliente para este caso ingresar el nombre del pasajero. Use variables locales para el nombre del pasajero y el país.

PASAJERO	
IDPASAJERO	
NOMBRES	
IDPAIS	
TELEFONO	
EMAIL	

PAIS	
Column Name	Condensed Type
IDPAIS	char(4)
PAIS	varchar(30)

```

--1.
DECLARE @PASAJE VARCHAR(40)='MARISOL DIAZ ZAMBRANO'
DECLARE @PAIS VARCHAR(40)

--2.
SELECT @PAIS=PAI.PAIS
    FROM PASAJERO PAS
    JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS

```

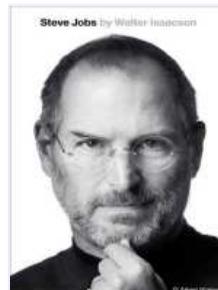
8 views

 1 0

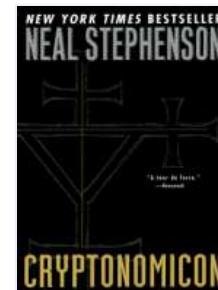
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by **anon_61286589**

Full description



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A Trt



CAPÍTULO 4. PROGRAMACIÓN TRANSACT-SQL

la variable @PASAJE es decir el nombre de un pasajero.

siguiente muestra el resultado si el pasajero fuera Marisol Díaz Zambrano.

Digitized by srujanika@gmail.com

4.4. FUNCIONES CAST Y CONVERT

El uso de CAST o CONVERT es para pasar de un tipo de datos a otro, normalmente se conversion cuando una función requiere un tipo especial de datos como parámetro.

Sintaxis:

TRANSACT SQL CAST Y CONVERT

```
CAST ( EXPRESIÓN AS TIPO_DATOS [ (LONGITUD) ])
CONVERT ( TIPO_DATOS [ (LOGITUD) ] , EXPRESIÓN)
```

Veamos la tabla donde se muestran todas las conversiones de tipos de datos explícitas permitidas para los tipos de datos proporcionados por el sistema de SQL Server.

8 views

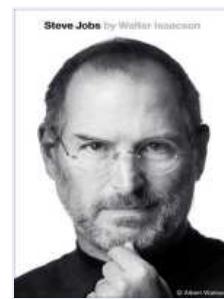
1 like

0 comments

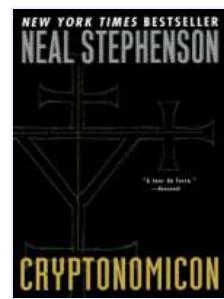
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

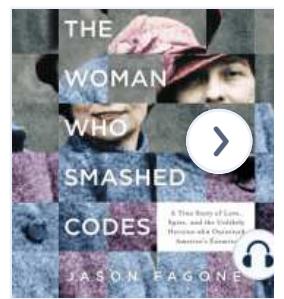
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

178 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Ejemplo: Si tenemos un monto registrado en la variable local @MONTO de 1250.75 y lo queremos mostrarle al usuario por medio de la función PRINT. Veamos tres implementaciones el primero sin conversión, el segundo con CAST y el tercero con CONVERT.

PRIMERO: SIN CONVERSION

```
DECLARE @MONTO MONEY
SET @MONTO = 1250.75
PRINT 'EL MONTO INGRESADO ES: '+@MONTO
GO
```

Mens. 235, Nivel 16, Estado 0, Línea 3
 No se puede convertir un valor char a money.
 La sintaxis del valor de tipo char es incorrecta.

Claro está que si usa la sentencia **SELECT** el resultado sería otro. Veamos de todo implementación con la sentencia SELECT:

```
DECLARE @MONTO MONEY
SET @MONTO = 1250.75
SELECT @MONTO AS [VALOR DE MONTO]
GO
```

VALOR DE MONTO	
1	1250.75

SEGUNDO: CAST

```
DECLARE @MONTO MONEY
SET @MONTO = 1250.75
PRINT 'EL MONTO INGRESADO ES: '+CAST(@MONTO AS CHAR(10))
GO
```

EL MONTO INGRESADO ES: 1250.75

TERCERO: CONVERT

```
DECLARE @MONTO MONEY
SET @MONTO = 1250.75
PRINT 'EL MONTO INGRESADO ES: '+CONVERT(CHAR(10),@MONTO)
GO
```

EL MONTO INGRESADO ES: 1250.75

Existe una función más que no es recomendada pero podría salvar en muchas ocasiones cuando no recuerde la sintaxis de las funciones CAST y CONVERT.

EXCEPCIONAL: STR

```
DECLARE @MONTO MONEY
```

8 views

1 like

0 comments

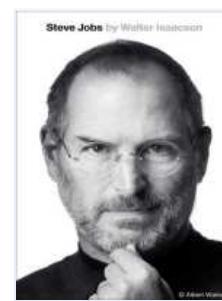
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

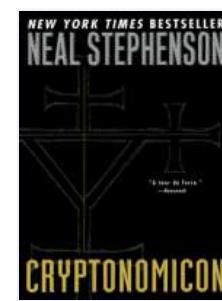
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

4.5. ESTRUCTURAS DE CONTROL

Existe un teorema de la estructura o también llamado teorema de Böhm-Jacopini en honor a Böhm y Giuseppe Jacopini donde se establece que toda función computable puede ser implementada en un lenguaje de programación en este caso Transact SQL cuenta solo con tres estructuras de control. Estas tres formas también son llamadas estructuras de control y son:

- **Estructuras secuenciales:** Donde se ejecutan las instrucciones una a continuación de la otra.
- **Estructuras selectivas:** Donde se ejecutan las instrucciones según el valor lógico de una condición.
- **Estructuras repetitivas:** Donde se ejecutan las instrucciones en forma repetida según una condición lógica, llamado ciclo o bucle.

En Transact SQL las estructuras de control permiten modificar el flujo de ejecución de las expresiones dentro de un script es decir ahora podrá insertar registros, modificarlos o eliminarlos de acuerdo a una condición.

Entonces las estructuras de control en Transact SQL pueden realizar:

- De acuerdo a una condición lógica, ejecutar un grupo u otro de sentencias (**IF...ELSE**)
- De acuerdo al valor de una variable poder determinar uno u otro valor dentro de las estructuras de control (**CASE**)
- Ejecutar un grupo de sentencias mientras una condición sea verdadera o falsa (**WHILE**)

Si tiene experiencia con alguna estructura de control en algún lenguaje de programación, recordar que en Transact SQL no será de mucho problema resolver un script que involucre lógica de control basada en dichas estructuras la única variación de estas estructuras es en la sintaxis y que en Transact SQL no existe la estructura repetitiva **FOR**.

Para un mejor control de las expresiones dentro de uno u otra estructura de control se pueden utilizar operadores de inicio y fin con los siguientes operadores:

- **BEGIN:** Indica el inicio de un proceso.
- **END:** Por cada Begin iniciado se debe culminar con el operador END para indicar que finalizó dicho proceso.

Sintaxis:

TRANSACT SQL

BEGIN END

```
<Estructura_Control>
BEGIN
    <expresiones>
END
```

8 views

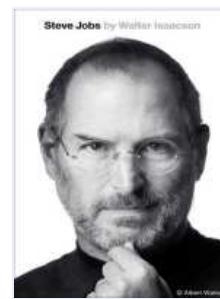
1 like

0 comments

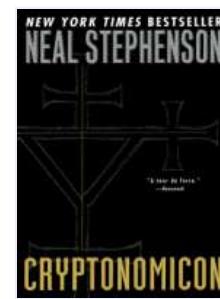
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True

180

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

4.6. ESTRUCTURA SELECTIVA IF

La estructura **IF** evalúa una condición lógica y en función del resultado se realiza una u otra acción. Dicha condición se especifica mediante una expresión lógica donde el resultado pueda ser booleano es decir True o False.

Sintaxis genérica:

TRANSACT SQL IF

```
IF <CONDICION_LOGICA>
<BEGIN>
    <expresiones_verdaderas>
<END>
```

Donde:

- **CONDICION_LOGICA:** Es la expresión que el usuario debe determinar para ejecutar las expresiones. Hay que tener en cuenta que aquí es el momento de usar los operadores lógicos (>=, <>) y relacionales (AND OR NOT EXISTS).
- **BEGIN:** Marca el inicio de las expresiones si la condición resultara True en su evaluación.
- **END:** Pone fin a las expresiones iniciada desde el operador BEGIN.

La estructura **IF** presenta dos alternativas para Transact SQL y tienen el siguiente formato:

IF DOBLE:

```
IF <CONDICION_LOGICA>
    <BEGIN>
        <expresiones_verdaderas>
    <END>
    ELSE
        <BEGIN>
            <expresiones_falsas>
        <END>
```

La estructura IF doble permite controlar una expresión lógica; determinando así qué expresiones se ejecutarán cuando la salida de dicha expresión sea TRUE o qué acciones se realizarán al resultar FALSE.

IF DOBLEMENTE ANIDADO:

```
IF <CONDICION_LOGICA>
    <BEGIN>
        <expresiones_verdaderas>
    <END>
```

8 views

1 like

0 comments

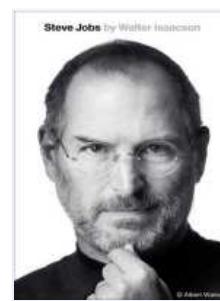
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

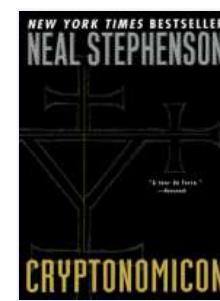
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

La estructura **IF** doblemente anidada normalmente se usa cuando expresión lógica devuelva resultado lógico es decir trabaja de acuerdo a las alternativas implementadas dentro de la estructura similar a trabajar con **CASE**.

CASO DESARROLLADO N° 4.4

Script que permite insertar un nuevo registro a la tabla país, en caso se registre duplicidad el script de un país mostrar un mensaje de **PAÍS YA REGISTRADO** caso contrario insertar dicho registro y mostrar un mensaje de **PAÍS REGISTRADO CORRECTAMENTE**.

```
--1.
DECLARE @IDPAIS CHAR(4)='0014',@PAIS VARCHAR(40)='HONDURAS'

--2.
IF EXISTS(SELECT * FROM PAIS WHERE PAIS=@PAIS)
BEGIN
    PRINT 'PAÍS YA REGISTRADO'
END
ELSE
BEGIN
    INSERT INTO PAIS
    VALUES(@IDPAIS,@PAIS)
    PRINT 'PAÍS REGISTRADO CORRECTAMENTE'
END
GO
```

En el punto uno se declara dos variables la primera almacenará el código del nuevo país a insertar y la segunda variable almacenara el nombre del país.

En el punto dos se verifica la existencia del valor asignado a la variable @PAIS dentro de la estructura IF. En caso la condición sea **TRUE** se muestra el mensaje **PAÍS YA REGISTRADO**, pero como colocó la impresión entre Begin End tenga en cuenta que no era necesario colocarlo para que no se rompa la regla sintáctica no habrá problemas. Cuando la condición es **FALSE** se ejecuta la instrucción **INSERT** y el mensaje de registrado correctamente esta vez sí tiene que estar entre Begin End ya que son dos instrucciones.

CASO DESARROLLADO N° 4.5

Script que permite mostrar el mensaje de **NO HAY PASAJEROS EN ESTE PAÍS** solo cuando no existan pasajeros asignados a un determinado PAÍS no tenga registros en la tabla PASAJERO. Caso contrario se debe determinar cuántos pasajeros tiene dicho país y mostrarlo con el mensaje **EL PAÍS X TIENE Y PASAJEROS**.

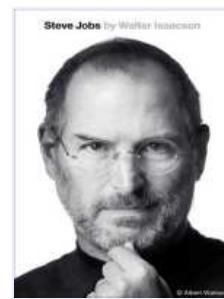
8 views

1 0

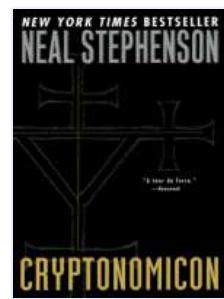
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

182 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

```
--1.
DECLARE @PAIS VARCHAR(40)='CHINA'

--2.
IF (SELECT COUNT(*)
     FROM PASAJERO PAS
     LEFT JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
     GROUP BY PAI.PAIS
     HAVING PAI.PAIS=@PAIS) IS NULL
    PRINT 'NO HAY PASAJEROS EN ESTE PAÍS'
ELSE
--3.
BEGIN
    DECLARE @TOTAL INT
    SELECT @TOTAL=COUNT(*)
        FROM PASAJERO PAS
        LEFT JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
        GROUP BY PAI.PAIS
        HAVING PAI.PAIS=@PAIS
    PRINT 'EL PAÍS '+@PAIS+' TIENE '+
    CAST(@TOTAL AS CHAR(2))+ ' PASAJEROS'
END
GO
```

En el punto uno se declara la variable local **@PAIS** y se le asigna el valor **CHINA** para realizar la consulta.

En el punto dos se evalúa el total de pasajeros del país en este caso **CHINA** se usa la cláusula **IS NULL** para poder contar también a los países que no registran pasajero alguno, la condición es con el operador **IS NULL** justamente para determinar si dicho país no tiene pasajeros así caso sea **TRUE** entonces se muestra el mensaje **NO HAY PASAJEROS EN ESTE PAÍS**.

En el punto tres las instrucciones implementadas solo ocurren cuando la condición del punto falso eso quiere decir que el país si tiene pasajeros asignados. Primero hay que indicar que las sentencias deben estar envueltas con **Begin End** ya que son más de dos instrucciones la que componen este proceso; luego se declara la variable **@TOTAL** para almacenar el total de pasajeros ya que estamos en la condición donde no es nulo la condición. Finalmente se imprime mostrando el nombre del país y el total de pasajeros que tiene asignado.

En caso la asignación del país sea **CHINA** entonces el mensaje se muestra en la imagen siguiente:

NO HAY PASAJEROS EN ESTE PAÍS

Si asignamos el país **PERU** entonces el mensaje de muestra en la imagen siguiente:

EL PAÍS PERU TIENE 3 PASAJEROS

8 views

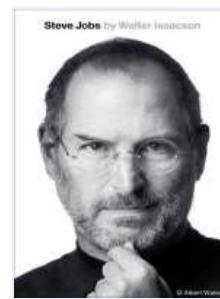
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

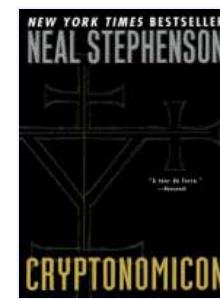
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

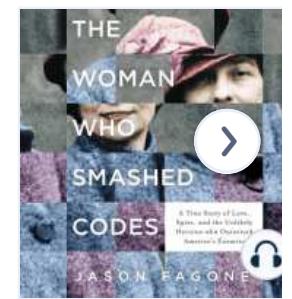
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

RESERVA

Column Name	Condensed Type	Nullable	Identity	Default Value
IDRESERVA	int	No	<input checked="" type="checkbox"/>	
COSTO	money	Yes	<input type="checkbox"/>	((0))
FECHA	date	Yes	<input type="checkbox"/>	(getdate())

```
--1.
DECLARE @AÑO INT=2012,@TOTAL INT

--2.
SELECT @TOTAL=COUNT(*)
      FROM RESERVA RES
      GROUP BY YEAR(RES.FECHA)
      HAVING YEAR(RES.FECHA)=@AÑO

--3.
IF @TOTAL>0 AND @TOTAL<=10
      PRINT 'EL TOTAL DE RESERVAS SE ENCUENTRA ENTRE 1 Y 10'
ELSE IF @TOTAL>10 AND @TOTAL<=15
      PRINT 'EL TOTAL DE RESERVAS SE ENCUENTRA ENTRE 11 Y 15'
ELSE
      PRINT 'EL TOTAL DE RESERVAS SUPERA A 15'

GO
```

En el punto uno se declara la variable @AÑO con la asignación del valor 2012 y también la TOTAL se declara para obtener el total de reservas por año.

En el punto dos se acumulan el total de reservas obtenidas desde la tabla RESERVA se agrupado es el mismo que está asignado en la variable.

En el punto tres se hace una comparación de rango de valores de ese total acumulado en @TOTAL.

4.7. ESTRUCTURA CONDICIONAL MÚLTIPLE CASE

Con frecuencia es necesario que existan más de dos posibles acciones en una determinación para esto se usa las estructuras de condicional múltiple. La estructura CASE evalúa una expresión y podrá tomar N valores distintos según se elija uno de estos valores se tomara N posibles acciones.

Sintaxis:

ESTRUCTURA DE CONTROL SELECTIVA MULTIPLE
CASE...WHEN

CASE <expresión>

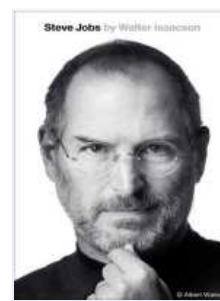
8 views

1 0

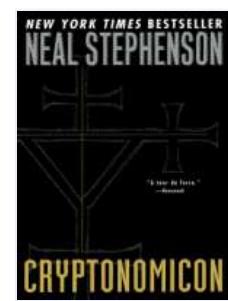
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

184 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Donde:

- **columna**: Es el nombre de la columna que se va a comparar, también se pueden usar f
- **WHEN** especifica las expresiones que se van a buscar según la condición.
- **THEN** especifican las expresiones resultantes de una opción.
- **END** especifica la finalización de la estructura.
- Una cláusula AS opcional que define un alias de la función CASE.

La estructura **CASE** presenta dos casos particulares de opción al momento de condicionar múltiple, veamos:

Primero: Múltiples condiciones para múltiples opciones**CASE**

```
WHEN expresion_condicional1 THEN resultado_expcion
ELSE resultado_expcion_falso
END
```

En este caso particular las condiciones podrían tratarse de un rango de valores que tiene que evaluados directamente en el WHEN.

Segundo: Múltiples opciones de una sola condición

```
CASE <expresion_condicional_unica>
WHEN valor_posible1 THEN resultado_posible1
WHEN valor_posible2 THEN resultado_posible2
...
ELSE resultado_expcion_falso
END
```

En este caso hay una sola condición que puede tener muchas opciones como resultado, tanto el WHEN no presenta una expresión sino más bien un valor de acuerdo al tipo de salida la condición.

CASO DESARROLLADO N° 4.7

Script que permita mostrar el tipo de servidor según el correo electrónico registrado por cada pasajero

PASAJERO	
IDPASAJERO	
NOMBRES	
IDPAIS	
TELEFONO	
EMAIL	

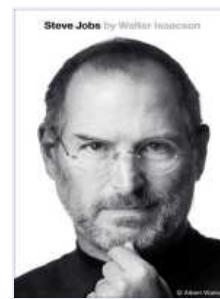
8 views

1 0

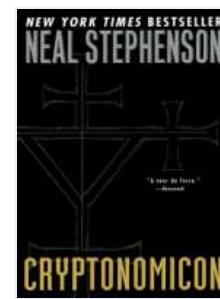
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

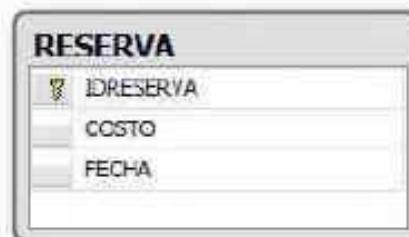
En el script se muestran todos las columnas de la tabla PASAJERO y además se evalúa el tipo del email del pasajero para el caso de Hotmail se condiciona con **WHEN EMAIL LIKE '%HOTMAIL%** algún valor coincide con el caso entonces mostrara el mensaje HOTMAIL, de igual manera para el servidor de gmail con **WHEN EMAIL LIKE '%GMAIL%' THEN 'GMAIL'**, después de la palabra **CASE** se puede asignar el nombre de la columna.

La siguiente imagen muestra el resultado de la consulta:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL	SERVIDOR DE CORREO
P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM	HOTMAIL
P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM	HOTMAIL
P0003	MARIA ZAMORA MEJIA	0005	957564525	MZAMORA@GMAIL.COM	GMAIL
P0004	GUADALUPE ACOSTA FERRER	0002	957564525	GACOSTA@HOTMAIL.COM	HOTMAIL
P0005	LULU LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM	GMAIL
P0006	KARLA GALLEGOS SILVA	0007	957564525	KGALLEGOS@HOTMAIL.COM	HOTMAIL

CASO DESARROLLADO N° 4.8

Script que permita mostrar la fecha en texto registrada en la tabla RESERVA de la siguiente forma: fecha fuera 2011-10-01 entonces deberá adicionar una columna a la consulta para mostrar el año 2011.



```

SELECT *,
CAST(DAY(FECHA) AS CHAR(2))+
CASE MONTH(FECHA)
    WHEN 1 THEN ' ENERO '
    WHEN 2 THEN ' FEBRERO '
    WHEN 3 THEN ' MARZO '
    WHEN 4 THEN ' ABRIL '
    WHEN 5 THEN ' MAYO '
    WHEN 6 THEN ' JUNIO '
    WHEN 7 THEN ' JULIO '
    WHEN 8 THEN ' AGOSTO '
    WHEN 9 THEN ' SETIEMBRE '
    WHEN 10 THEN ' OCTUBRE '
    WHEN 11 THEN ' NOVIEMBRE '
    WHEN 12 THEN ' DICIEMBRE '
END

```

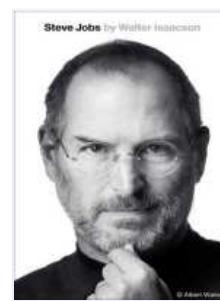
8 views

1 0

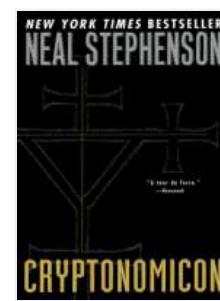
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

186 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el script se muestran todas las columnas de la tabla **RESERVA** además de componer u letras a partir de la fecha con formato corto. Para dicha composición primero debemos ca de la semana con el siguiente script **DAY(FECHA)**, luego se necesita obtener el número c convertirlo en texto, para obtener usamos la función **MONTH(FECHA)** en este caso la estr toma un segundo formato donde la evaluación del caso se realiza desde la columna **FECHA** de allí el número de mes, por tanto si el numero evaluado por **CASE MONTH(FECHA)** es 1 texto a mostrar será ENERO con la siguiente instrucción **WHEN 1 THEN 'ENERO'** en este es una condición sino más bien una opción de lo evaluado en el **CASE**, también tenga ei espacios entre el texto **ENERO** ya que de otra manera saldría de la siguiente forma **1ENERO** final de la composición se encuentra el año obtenido desde el script **YEAR(FECHA)**.

La siguiente imagen muestra el resultado de la consulta:

IDRESERVA	COSTO	FECHA	FECHA
1	450.00	2011-10-01	1 OCTUBRE 2011
2	850.00	2011-10-06	6 OCTUBRE 2011
3	450.00	2011-11-14	14 NOVIEMBRE 2011
4	1150.00	2011-11-16	16 NOVIEMBRE 2011
5	1450.00	2011-12-12	12 DICIEMBRE 2011
6	540.00	2011-12-17	17 DICIEMBRE 2011
7	400.00	2012-01-14	14 ENERO 2012
8	1300.00	2012-01-15	15 ENERO 2012

CASO DESARROLLADO N° 4.9

Script que permita mostrar el número total de pasajeros y el mensaje NO CUENTA solo cuyo número de pasajeros sea cero.

PASAJERO	
IDPASAJERO	
NOMBRES	
IDPAIS	
TELEFONO	
EMAIL	

Column Name	Condensed Type
IDPAIS	char(4)
PAIS	varchar(30)

```

SELECT PAI.PAIS,
       COUNT(PAS.IDPASAJERO) AS [TOTAL PASAJEROS],
       CASE
           WHEN COUNT(PAS.IDPASAJERO)=0 THEN 'NO CUENTA'
           ELSE ''
       END AS [MENSAJE]
FROM PAIS PAI
LEFT JOIN PASAJERO PAS ON PAI.IDPAIS=PAS.IDPAIS
GROUP BY PAI.PAIS
  
```

8 views

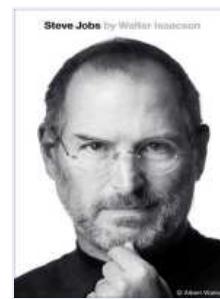
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

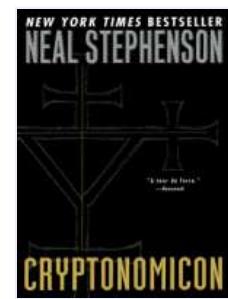
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

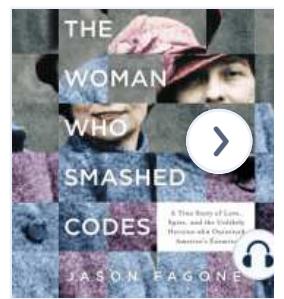
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

CAP. 4: PROGRAMACIÓN TRANSACT S

Si es así entonces se pasa a una evaluación **CASE** para determinar qué cantidad de pasajeros la implementación dentro del caso **WHEN COUNT(IDPASAJERO)=0 THEN** aquí se muestra **NO CUENTA** para determinar que dicho país no cuenta con pasajeros registrados, caso contrario; ahora se preguntará porque se implementa el operador **ELSE** si al final no habrá necesidad de colocar las comillas vacías ya que de otra manera la consulta mostraría el operador para aquellos países que si registran por lo menos un pasajero y el problema planteado no detallaría el mensaje mostrar en este caso.

La imagen siguiente muestra el resultado de la consulta:

	PAÍS	TOTAL PASAJEROS	MENSAJE
1	ARGENTINA	1	
2	BOLIVIA	0	NO CUENTA
3	BRASIL	1	
4	CHILE	1	
5	ECUADOR	2	
6	EEUU	0	NO CUENTA
7	HONDURAS	0	NO CUENTA
8	MEXICO	1	
9	PARAGUAY	1	
10	PERU	3	
11	PUERTO R.	0	NO CUENTA
12	URUGUAY	0	NO CUENTA
13	VENEZUELA	0	NO CUENTA

Como vera el mensaje **NO CUENTA** solo aparece en los países que tienen como total de valor cero, caso contrario no se muestra nada.

4.8. ESTRUCTURA DE CONTROL WHILE

Desde que la computadora fue diseñada siempre tuvo como visión hacer que las aplicaciones ejecutaran notadamente las líneas de código haciendo inclusive que aquellas instrucciones que se repiten se tratase dentro de un control de flujo repetitivo. Debido a esto se implementó la estructura de control repetitiva como **WHILE** en Transact SQL.

WHILE hace repetir un conjunto de instrucciones en secuencia un número determinado de veces. Esto se le denomina bucle o ciclo. Veamos si se desea acumular sumas o determinar el mayor elemento de un conjunto de valores entonces estamos frente a un caso de repeticiones que se controla y administrada por la estructura repetitiva **WHILE**.

En **WHILE** se establece una condición para la ejecución repetida de una instrucción o más instrucciones SQL. Las instrucciones se ejecutan repetidamente siempre que la condición sea verdadera. Se puede controlar la ejecución de instrucciones en el bucle **WHILE** con la palabras **BREAK** y **CONTINUE**.

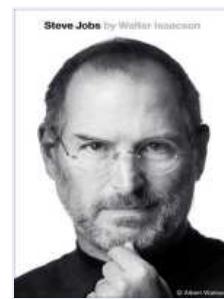
8 views

1 0

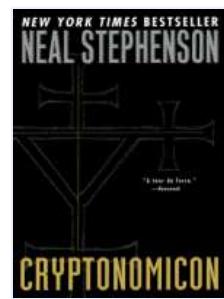
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

188 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Donde:

- **Condición:** Se aplicara una evaluación lógica sobre dicha condición para poder determinar los ciclos darán las expresiones colocadas dentro del While. Considere que puede usar los operadores lógicos y relaciones de SQL server.
- **BREAK:** Permite cortar el ciclo de repeticiones y según la secuencia de instrucciones se aplica en la aplicación.
- **CONTINUE:** Permite regenerar el ciclo de repeticiones no tomando en cuenta las asignadas después del operador **CONTINUE**.
- Es mejor plantear la estructura While con los operadores de inicio y fin (BEGIN END) para tener un mejor control de las expresiones internas del ciclo, además de poder usar las estructuras anteriores como IF o CASE.

CASO DESARROLLADO N° 4.10

Script que permite mostrar los números consecutivos del 1 al 10.

```
--1.
DECLARE @N INT=0

--2.
WHILE (@N<10)
BEGIN
    SET @N+=1
    PRINT 'VALOR I: '+CAST(@N AS CHAR(2))
END
GO
```

En el punto uno se declara una variable **@N** que permitirá dar inicio al ciclo de repeticiones.

En el punto dos se aplica la estructura **WHILE** donde la condición para que el ciclo funcione es que el valor de la variable **@N** sea menor a 10, luego se encierra todas las instrucciones del ciclo entre **END** y **GO** y así indicar que se repetirán solo si la condición del **WHILE** es **TRUE**.

Dentro del ciclo de repeticiones se asigna el aumento en uno a la variable **@N** para que desde la función **PRINT**, no se olvide que se tiene que aplicar un **CAST** a la variable **@N**.

La imagen siguiente muestra el resultado del ciclo:

VALOR I:	1
VALOR I:	2
VALOR I:	3
VALOR I:	4
VALOR I:	5
VALOR I:	6
VALOR I:	7
VALOR I:	8
VALOR I:	9
VALOR I:	10

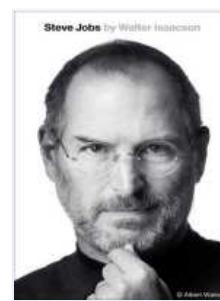
8 views

1 0

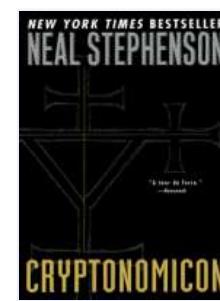
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

Tenga en cuenta que el script pudo tener una forma distinta de implementar los ciclos como resultado, podría ser:

```

DECLARE @N INT=1
WHILE (@N<=10)
BEGIN
    PRINT 'VALOR I: '+CAST(@N AS CHAR(2))
    SET @N+=1
END
GO

```

CASO DESARROLLADO N° 4.11

Script que permita aumentar en 10% según el valor del costo de las reservas solo si el promedio de estas no supera a los 10000, cuando se termine de actualizar dichos valores mostrar el mensaje HAY MAS QUE ACTUALIZAR.

RESERVA			
IDRESERVA	COSTO	FECHA	
1	450.00	2011-10-01	
2	850.00	2011-10-06	
3	450.00	2011-11-14	
4	1150.00	2011-11-16	
5	1450.00	2011-12-12	
6	540.00	2011-12-17	
7	400.00	2012-01-14	
8	1300.00	2012-01-15	
9	1000.00	2012-02-01	
10	1800.00	2012-04-02	
11	1200.00	2012-04-09	
12	400.00	2012-08-01	
13	800.00	2012-08-15	

Lo primero que tenemos que hacer es verificar cuales son los registros actuales de la tabla.

IDRESERVA	COSTO	FECHA
1	450.00	2011-10-01
2	850.00	2011-10-06
3	450.00	2011-11-14
4	1150.00	2011-11-16
5	1450.00	2011-12-12
6	540.00	2011-12-17
7	400.00	2012-01-14
8	1300.00	2012-01-15
9	1000.00	2012-02-01
10	1800.00	2012-04-02
11	1200.00	2012-04-09
12	400.00	2012-08-01
13	800.00	2012-08-15

```

--1.
WHILE (SELECT AVG(COSTO) FROM RESERVA) < 10000
BEGIN
    --2.
    UPDATE RESERVA
        SET COSTO=COSTO*1.1

```

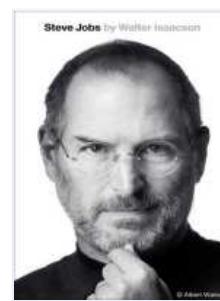
8 views

1 0

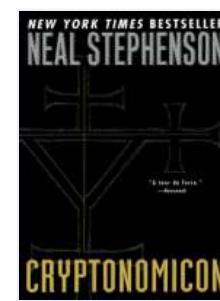
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

190

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el punto uno se inicia el ciclo de repeticiones con WHILE la condición para se inicie el promedio es menor a 10000; para devolver el promedio se usó la función AVG aplicada a COSTO.

En el punto dos se actualiza el costo de todas las reservas de la tabla RESERVA solo si la planteada en WHILE sea TRUE. En este caso la actualización se realiza a la misma columna adicionando el 10% a su valor actual **SET COSTO=COSTO*1.1**.

En el punto tres se evaluará si ya se llegó al tope exigido por la condición ya que si no el ciclo no tendría como finalizar y se quedaría actualizando todo el tiempo a los valores de COSTO. Consideraremos necesario usar la función MAX para poder determinar cuál es el más actualizado en la tabla RESERVA de acuerdo a esto si ya se llegó al tope entonces usamos BREAK para salir del ciclo de repeticiones caso contrario tiene que continuar con la actualización que el máximo supere el valor tope.

Finalmente se muestra un mensaje de **YA NO HAY MAS QUE ACTUALIZAR** que inmediatamente después de ejecutar el **BREAK** dentro del ciclo de repeticiones.

La imagen siguiente muestra que la actualización se ha realizado a todos los costos sin excepción que su promedio sea superior a 10000.

DRESERVA	COSTO	FECHA
1	2501.9632	2011-10-01
2	4725.93	2011-10-06
3	2501.9632	2011-11-14
4	6393.9051	2011-11-16
5	8051.8803	2011-12-12
6	3002.3552	2011-12-17
7	2223.9569	2012-01-14
8	7227.8925	2012-01-15
9	5559.9176	2012-02-01
10	10007.8...	2012-04-02
11	6671.901	2012-04-09
12	2223.9669	2012-08-01
13	4447.934	2012-08-15

El mensaje al final de las actualizaciones sucesivas lo muestra la siguiente imagen:

(26 filas afectadas)
YA NO HAY MAS QUE ACTUALIZAR

CASO DESARROLLADO N° 4.12

Script que permita disminuir en un 50% el valor del costo de las reservas solo si el promedio supera a los 3000 para los registros del año ingresado por el usuario, cuando se termine de

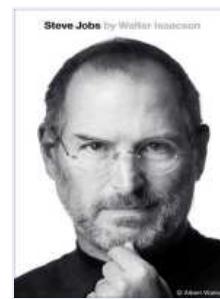
8 views

1 0

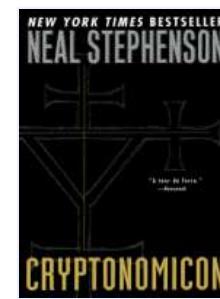
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

```
--1.
DECLARE @AÑO INT=2011
WHILE (SELECT AVG(COSTO)
       FROM RESERVA
       WHERE YEAR(FECHA)=@AÑO) > 3000
BEGIN
    --2.
    UPDATE RESERVA
        SET COSTO=COSTO/2
        WHERE YEAR(FECHA)=@AÑO

    --3.
    IF (SELECT MAX(COSTO)
        FROM RESERVA
        WHERE YEAR(FECHA)=@AÑO)<=3000
        BREAK
    ELSE
        CONTINUE
END
PRINT 'YA NO HAY MAS QUE ACTUALIZAR EN ESE AÑO'
```

En el punto uno se declara la variable **@AÑO** que permitirá asignar el valor de un año respetando la condición de que el promedio de la columna COSTO sea menor a 3000. Luego se implementa el ciclo WHILE para actualizar los registros de acuerdo a ese valor. Allí mismo se implementa la condición del promedio de la columna COSTO solo del año solicitado en la variable **@AÑO** esto se realizó con la sentencia **YEAR(FECHA)=@AÑO** a todo esto se le evalúa si dicho promedio no supera el valor 3000 en ese caso se ejecuta la sentencia **BREAK** caso contrario se indica entonces que aún hay costos de ese año que aún no han sido actualizados eso se logrará con el operador **CONTINUE**.

En el punto dos se implementa la sentencia **UPDATE** para poder actualizar la columna COSTO reduciendo a la mitad dicho costo solo para el año especificado en la condición.

En el punto tres se verifica si ese promedio es menor o igual a 3000 entonces deberá cortar el bucle con la sentencia **BREAK** caso contrario se indica entonces que aún hay costos de ese año que aún no han sido actualizados eso se logrará con el operador **CONTINUE**.

Y para mostrar un mensaje de finalización del ciclo se imprime el texto **YA NO HAY MAS QUE ACTUALIZAR EN ESE AÑO**.

Si verificamos los valores iniciales de la tabla **RESERVA**, notará que el registro 1 tiene como valor **2752.1595** al aplicar el script su nuevo valor será **1376.0797** que representa a la mitad del valor inicial.

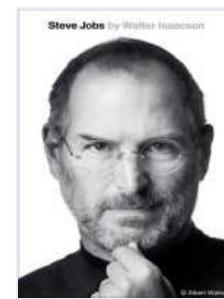
8 views

1 0

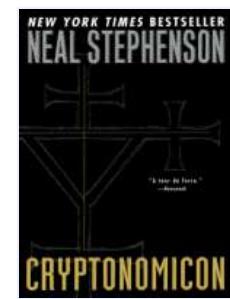
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

192 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

IDRESERVA	COSTO	FECHA
1	2752.1595	2011-10-01
2	5198.523	2011-10-06
3	2752.1595	2011-11-14
4	7033.2956	2011-11-16
5	8068.0683	2011-12-12
6	3102.5907	2011-12-17
7	2446.3636	2012-01-14
8	7950.6818	2012-01-15
9	6115.9094	2012-02-01
10	11008.6...	2012-04-02
11	7339.0911	2012-04-09
12	2446.3636	2012-08-01
13	4092.7274	2012-08-15

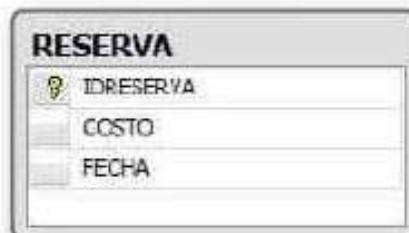
IDRESERVA	COSTO	FECHA
1	1376.0797	2011-10-01
2	2599.2615	2011-10-06
3	1376.0797	2011-11-14
4	3516.6478	2011-11-16
5	4434.0341	2011-12-12
6	1651.2953	2011-12-17
7	2446.3636	2012-01-14
8	7950.6818	2012-01-15
9	6115.9094	2012-02-01
10	11008.6365	2012-04-02
11	7339.0911	2012-04-09
12	2446.3636	2012-08-01
13	4092.7274	2012-08-15

El mensaje final mostrado al usuario solo en todos los casos es:

YA NO HAY MAS QUE ACTUALIZAR EN ESE AÑO

CASO DESARROLLADO N° 4.13

Script que permita mostrar los N últimos registros de la tabla RESERVA condicionando que el registro de reserva sean las más actuales posibles.



```
--1.
DECLARE @N INT=1,@TOPE INT=3

--2.
WHILE @N<=100
BEGIN
    --3.
    IF @N=@TOPE
    BEGIN
        SELECT TOP(@N) * FROM RESERVA ORDER BY RESERVA.FECHA DESC
        BREAK
    END
    --4.
    ELSE
    BEGIN
        SET @N+=1
        CONTINUE
    END
END
```

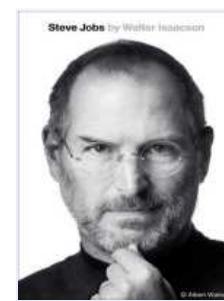
8 views

1 0

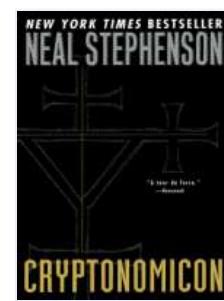
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

En el punto dos se implementa el ciclo WHILE con la condición que @N sea menor que 10 representa un límite superior a lo que el usuario pueda ingresar en la variable @TOPE. Si ciclo de repeticiones nos encontramos con la condicional IF que permitirá determinar si @ límite solicitado por el usuario si esto es así entonces se imprimen los registros hasta el tope que también lo tiene asignado en @N ordenándolo por la columna FECHA en forma de salir del ciclo de repeticiones con BREAK ya que se llegó al valor @TOPE.

En el punto cuatro solo será accesible si aún no se llega al tope por tanto la variable @N aumentar su valor en uno ya que el ciclo lo volverá a evaluar hasta que cumpla con el tope y no se olvide que aquí se invoca al operador CONTINUE para que el proceso repetitivo siga.

El resultado se muestra en la siguiente imagen:

IDRESERVA	COSTO	FECHA
1	13	4892.7274 2012-08-15
2	26	4892.7274 2012-08-15
3	25	2445.3636 2012-08-01

CASO DESARROLLADO N° 4.14

Script que permita mostrar los registros de la tabla PASAJERO teniendo el control de los decir especificar desde que registró y hasta donde se deberán mostrar.



```
--1.
DECLARE @VI INT=3,@VF INT=5

--2.
SELECT *
FROM (SELECT ROW_NUMBER()
OVER (ORDER BY P.IDPASAJERO) AS [NUMERO],*
      FROM PASAJERO P) X
WHERE NUMERO BETWEEN @VI AND @VF
GO
```

En el punto uno se declara la variable @VI que tiene asignado el valor inicial de los registros y el @VF define el valor final de los mismos.

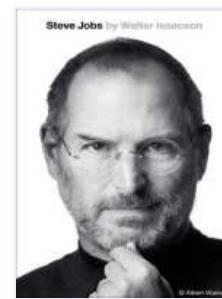
8 views

1 0

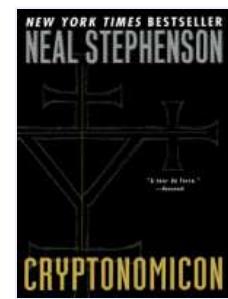
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

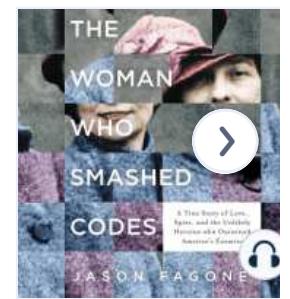
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

194 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

La imagen siguiente muestra el resultado de la consulta:

	NUMERO	IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1	3	P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
2	4	P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
3	5	P0005	LUZ LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM

CASO DESARROLLADO N° 4.15

Script que permite mostrar los registros de la tabla PASAJERO paginados de 5 en 5.

RowNum	IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1	P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2	P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3	P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4	P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5	P0005	LUZ LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM

RowNum	IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
6	P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7	P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8	P0008	HEIDI RENGIFO REATE...	0004	957564526	HRENGIFO@HOTMAIL.COM
9	P0009	MARISOL DIAZ ZAMBRA...	0004	957564526	MDIAZ@GMAIL.COM
10	P0010	GUADALUPE ACOSTA F...	0003	938475747	GACOSTA@GMAIL.COM



--1. La página que deseamos mostrar.

DECLARE @PAGINA INT=1

WHILE(@PAGINA<=5)

BEGIN

--2. Número de filas por página.

DECLARE @CANT_DEFILAS INT = 5

SELECT * FROM (SELECT ROWNUM = ROW_NUMBER()
OVER (ORDER BY P.IDPASAJERO), *
FROM PASAJERO P) X

WHERE

ROWNUM > (@CANT_DEFILAS * (@PAGINA - 1)) AND

ROWNUM <= (@CANT_DEFILAS * (@PAGINA - 1)) + @CANT_DEFILAS

SET @PAGINA+=1

END

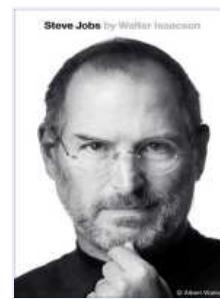
8 views

1 0

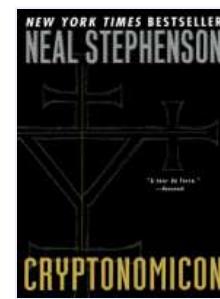
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

En el punto dos se declara una variable **@CANT_DEFILAS** que representa el número de fi página a mostrar. Se inicializa con 5 a la variable y así determinar la cantidad de pasajeros por

Luego se implementa una consulta que permitirá mostrar los registros asignados por cada se adiciona una subconsulta llamada **X** para controlar los registros de la tabla y poder eval de valores de la columna **ROWNUM** que representa al número secuencial de los registro **PASAJERO**.

La imagen siguiente muestra el resultado de la paginación:

RowNum	IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1	P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2	P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3	P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4	P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5	P0005	LUZ LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM
RowNum	IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
6	P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7	P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8	P0008	HEIDI RENGIFO REATE	0004	957564526	HRENGIFO@HOTMAIL.COM
9	P0009	MARISOL DIAZ ZAMBRA	0004	957564526	MDIAZ@GMAIL.COM
10	P0010	GUADALUPE ACOSTA F...	0003	939475747	GACOSTA@GMAIL.COM

4.9. CONTROL DE ERRORES EN TRANSACT SQL

Los lenguaje de programación tienen un especial control de los errores posibles que pued a nivel de programación, SQL Server incorpora diferentes alternativas para controlar dic no encargando esta tarea al lenguaje de programación sino al Transact SQL desligándose las validaciones propias del SQL, a partir de la versión 2005 del SQL Server se incorpora el CATCH que permite gestionar los errores de manera adecuada.

Sintaxis:

CONTROL DE ERRORES TRY-CATCH

```
BEGIN TRY
    Expresión_Sql
END TRY
BEGIN CATCH
    Expresión_Sql
END CATCH
[ ; ]
```

Dónde:

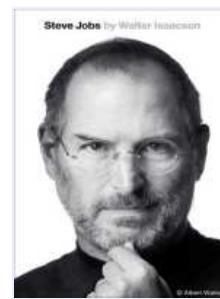
8 views

1 0

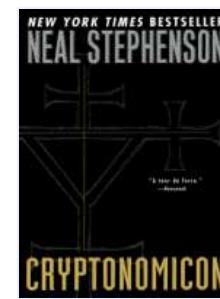
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

196 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Consideraciones:

- El bloque **TRY-CATCH** detecta todos los errores de ejecución que tienen una gravedad i y que no cierran la conexión con la base de datos en SQL Server.
- La implementación de un bloque TRY-CATCH no puede abarcar varios bloques del mismo se podrá incluir dentro de un Try-Catch otro Try-Catch ni mucho menos dentro de un blo ya que generaría un error desde el Motor de Base de Datos de SQL Server. Pero dentro BEGIN CATCH si se puede implementar un BEGIN TRY.
- La forma de trabajo del bloque TRY-CATCH es que si no hay errores en el script incluido d bloque BEGIN TRY, se ejecutarán las sentencias inmediatamente posterior a la instrucc hubiera un error en el script incluido en un bloque BEGIN TRY, el control se transfiere instrucción del bloque BEGIN CATCH.
- Toda vez que la instrucción END CATCH sea la última de un store procedure o trigger, e devuelve a la instrucción que llamó al store procedure o activó el trigger.
- Los errores capturados por un bloque BEGIN CATCH no devuelven valor alguno a la que realiza la llamada. En caso necesite devolver alguna información tendrá que imp funciones como RAISERROR o PRINT.
- La implementación de un bloque TRY...CATCH no se puede utilizar en una función de usuario.

4.10. FUNCIONES ESPECIALES DE ERROR

- **ERROR_NUMBER()**: Devuelve el número de error detectado.
- **ERROR_MESSAGE()**: Devuelve el texto completo del mensaje de error. El texto incluye suministrados para los parámetros sustituibles, como longitudes, nombres de objeto u
- **ERROR_SEVERITY()**: Devuelve la gravedad del error.
- **ERROR_STATE()**: Devuelve el número de estado del error.
- **ERROR_LINE()**: Devuelve el número de línea dentro de la rutina en que se produjo el er
- **ERROR_PROCEDURE()**: Devuelve el nombre del procedimiento almacenado o desenca que se produjo el error.

CASO DESARROLLADO N° 4.16

Script que permita mostrar el valor devuelto por las funciones de control de errores a p división sin resultado como 1/0.

```
BEGIN TRY
    SELECT 1/0
END TRY
BEGIN CATCH
```

8 views

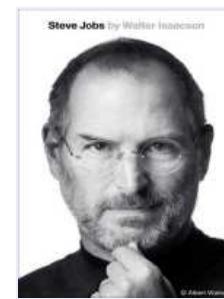
1 like

0 comments

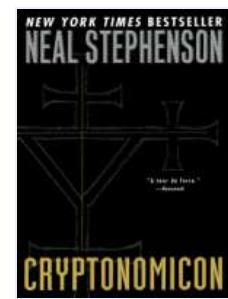
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

En el script el **BEGIN TRY** se muestra la consulta 1/0 normalmente debería emitir un resultado de división no fuera indeterminada, eso quiere decir que en este caso no emitirá nada y quedará vacío.

En el **BEGIN CATCH** se implementa una consulta donde se mostrarán todos los valores emitidos por las funciones de error.

La imagen siguiente muestra el resultado del script:

(Sin nombre de columna)					
ErrorNumber	ErrorSeverity	ErrorState	ErrorProcedure	ErrorLine	ErrorMessage
1	8134	16	1	NULL	2

CASO DESARROLLADO N° 4.17

Script que permite insertar un registro a la tabla PASAJERO controlando los posibles errores con bloque TRY y CATCH.



BEGIN TRY

```
DECLARE @IDPAS CHAR(5)='P0010',
@NOM VARCHAR(40)='MANUEL TORRES REMON',
@IDPAI CHAR(4)='0004',@TEL CHAR(15)='9999999999',
@EMA VARCHAR(40)='PMTORRES@CIBERTEC.EDU.PE'
```

INSERT INTO PASAJERO

```
VALUES(@IDPAS,@NOM,@IDPAI,@TEL,@EMA)
```

END TRY

BEGIN CATCH

```
PRINT 'ERROR EN LA TABLA PASAJERO'
```

END CATCH

En el script se declaran variables que representan a la misma cantidad de columnas de la tabla PASAJERO y también se inicializa con los valores mostrados, hemos dejado un error en el script para que el IDPASAJERO P0010 ya fue ingresado antes, eso quiere decir que el error es de tipo 1/0. Un mensaje sin control de errores se mostraría como la siguiente imagen:

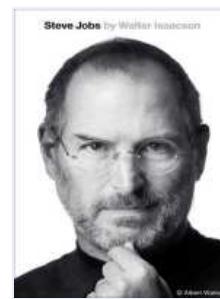
8 views

1 0

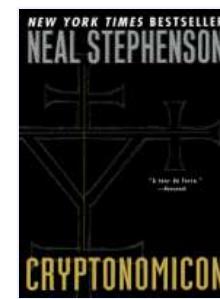
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

198

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Si empleamos las funciones de error podríamos implementar el script de la siguiente manera:

```

BEGIN TRY
    DECLARE @IDPAS CHAR(5)='P0010',
            @NOM VARCHAR(40)='MANUEL TORRES REMON',
            @IDPAI CHAR(4)='0004',@TEL CHAR(15)='9999999999',
            @EMA VARCHAR(40)='PMTORRES@CIBERTEC.EDU.PE'

    INSERT INTO PASAJERO
        VALUES(@IDPAS,@NOM,@IDPAI,@TEL,@EMA)
END TRY
BEGIN CATCH
    PRINT 'ERROR EN LA TABLA PASAJERO'
    PRINT 'Numero de Error           :'+CAST(ERROR_NUMBER() AS VARCHAR(10))
    PRINT 'Numero de Severidad       :'+CAST(ERROR_SEVERITY() AS VARCHAR(10))
    PRINT 'Numero de Estado          :'+CAST(ERROR_STATE() AS VARCHAR(10))
    PRINT 'Linea de Error N° :'+CAST(ERROR_LINE() AS VARCHAR(10))
    PRINT 'Mensaje de Error          :'+ERROR_MESSAGE()
END CATCH

```

La siguiente imagen muestra el resultado de la consulta:

```

|0 filas afectadas|
ERROR EN LA TABLA PASAJERO
Numero de Error      :2627
Numero de Severidad :14
Numero de Estado     :1
Linea de Error N°   :5
Mensaje de Error     :Infracción de la restricción PRIMARY KEY 'PK_PASAJERO_166890CB20C1E1'.
                           No se puede insertar una clave duplicada en el objeto 'dbo.PASAJERO'.
|

```

CASO DESARROLLADO N° 4.18

Script que permite mostrar la tabla PASAJERO donde comprueba si la tabla existe, en caso de no existir mostrar un mensaje de TABLA NO REGISTRADA EN LA BASE. Use el bloque TRY y CATCH.



```
BEGIN TRY
```

```
SELECT * FROM PASAJEROS
```

8 views

1 like

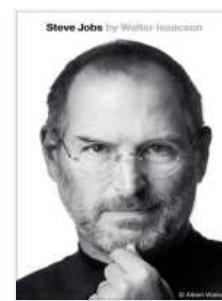
0 comments

eMacro - Programación Transact SQL Server 2012-4.pdf

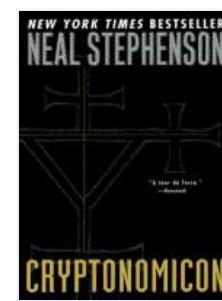
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

En primera instancia estaríamos pensando que el script anterior soluciona al error que genera al no proporcionar un nombre correcto a la tabla. Pero lo único que logramos es un error desde el Motor de Base de Datos. La imagen siguiente muestra el error conseguido:

```
Mens. 208, Nivel 16, Estado 1, Línea 2
El nombre de objeto 'PASAJEROS' no es válido.
```

Para dar solución al caso debemos implementar un procedimiento almacenado que realiza una validación errada y este al devolver el control al bloque si podrá detectar un error, aquí se muestra el código:

Primero: Se implementa el procedimiento almacenado **SP_TABLA** que permite mostrar los datos de la tabla PASAJEROS si la tabla en mención no existe dentro de la base de datos.

```
CREATE PROC SP_TABLA
AS
    SELECT * FROM PASAJEROS
GO
```

Segundo: Ahora modificamos el bloque BEGIN TRY para hacer la invocación de la ejecución del procedimiento almacenado con EXEC SP_TABLA.

```
BEGIN TRY
    EXEC SP_TABLA
END TRY
BEGIN CATCH
    PRINT 'TABLA NO REGISTRADA EN LA BASE DE DATOS'
    PRINT 'Número de Error' :'+CAST(ERROR_NUMBER() AS VARCHAR(10))
    PRINT 'Número de Severidad' :'+CAST(ERROR_SEVERITY() AS VARCHAR(10))
    PRINT 'Número de Estado' :'+CAST(ERROR_STATE() AS VARCHAR(10))
    PRINT 'Línea de Error Nº' :'+CAST(ERROR_LINE() AS VARCHAR(10))
    PRINT 'Mensaje de Error' :'+ERROR_MESSAGE()
END CATCH
GO
```

La imagen siguiente muestra el resultado de la consulta con los cambios establecidos:

```
TABLA NO REGISTRADA EN LA BASE DE DATOS
Número de Error :208
Número de Severidad :16
Número de Estado :1
Línea de Error Nº :4
Mensaje de Error :El nombre de objeto 'PASAJEROS' no es válido.
```

8 views

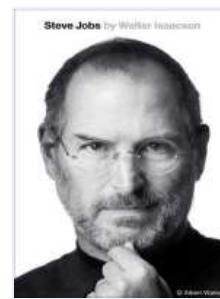
1 like

0 comments

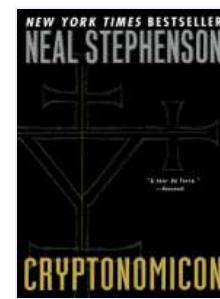
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

200 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

4.11. FUNCIÓN @@ERROR

La función del sistema **@@ERROR** devuelve 0 si la última sentencia Transact-SQL se ejecutó sin error, **@@ERROR** devuelve el número de error. Hay que tener en cuenta que el valor de **@@ERROR** cambia al finalizar cada sentencia Transact-SQL.

En vista que **@@ERROR** obtiene un nuevo valor cuando se completa cada sentencia Transact-SQL entonces deberá considerar las siguientes opciones:

- Use la función **@@ERROR** después de una instrucción Transact-SQL.
- Guarde **@@ERROR** en una variable local de tipo entero después de que se complete una instrucción Transact-SQL. El valor de la variable se puede usar inmediatamente dentro de la asignación.

Consideraciones:

- **@@ERROR** debe comprobarse o guardarse después de cada instrucción Transact-SQL. Un programador no puede predecir qué instrucción puede generar un error. Es necesario codificar una lista de instrucciones Transact-SQL que deben implementarse para manejar la lógica deseada.
- Las construcciones TRY...CATCH son mucho más simples. Un bloque de instrucciones TRY se limita por instrucciones BEGIN TRY y END TRY y después se escribe un bloque CATCH para gestionar errores que pueden generarse mediante el bloque TRY.
- Fuera de un bloque CATCH, **@@ERROR** es la única parte de un error de Database Engine (base de datos) disponible en el lote, procedimiento almacenado o desencadenador que genera el error. El resto de las partes del error, como su gravedad, estado y texto del mensaje que aparece en la pantalla, así como las cadenas de sustitución (nombres de objeto, por ejemplo) sólo se devuelven a la aplicación mediante los mecanismos de control de errores de las API. Si el error ocurre dentro de un bloque CATCH, se pueden usar las funciones del sistema **ERROR_LINE**, **ERROR_MESSAGE**, **ERROR_PROCEDURE**, **ERROR_NUMBER**, **ERROR_SEVERITY** y **ERROR_STATE**.

CASO DESARROLLADO N° 4.19

Script que permite actualizar los datos de un determinado pasajero. Use la función **@@ERROR**

```
--1.
UPDATE PASAJERO
SET IDPAIS='9999'
WHERE PASAJERO.IDPASAJERO='P0010'

--2.
IF @@ERROR<>0
    PRINT 'OCURRIO UN ERROR AL ACTUALIZAR LA TABLA PASAJERO'
GO
```

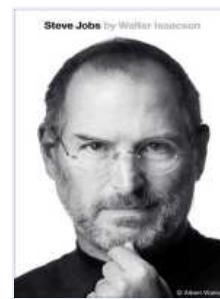
8 views

1 0

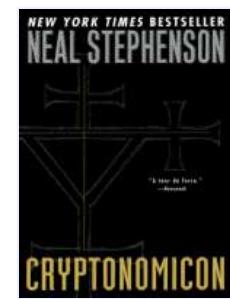
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

En el punto dos se verifica que la función @@ERROR sea diferente de cero ya que seguirá emitirán cero cuando no tenga errores en la sentencia SQL. Entonces el resultado se muestra en la siguiente imagen:

```
Mens. 547, Nivel 16, Estado 0, Línea 2
Instrucción UPDATE en conflicto con la restricción FOREIGN KEY "FK_PASAJERO_IDPAIS_ZE1BDC42".
El conflicto ha aparecido en la base de datos "AGENCIA", tabla "dbo.PAIS", columna 'IDPAIS'.
Se terminó la instrucción.
OCURRIO UN ERROR AL ACTUALIZAR LA TABLA PASAJERO
```

Para mejorar el script anterior emplearemos el bloque TRY CATCH de la siguiente forma:

```
--1.
BEGIN TRY
    UPDATE PASAJERO
        SET IDPAIS='9999'
        WHERE PASAJERO.IDPASAJERO='P0010'
END TRY
--2.
BEGIN CATCH
    IF @@ERROR<>0
        PRINT 'OCURRIO UN ERROR AL ACTUALIZAR LA TABLA PASAJERO'
END CATCH
GO
```

En el punto uno se implementa el bloque BEGIN TRY, en donde se coloca la sentencia UPDATE. Sabemos que si ocurriera algún error dentro de este bloque se enviará al bloque BEGIN CATCH.

En el punto dos se implementa el bloque BEGIN CATCH que permite verificar que error se produjo. En este caso solo se condiciona a que @@ERROR sea diferente de cero y se imprimirá un mensaje genérico de error. La diferencia con el script anterior es que no mostrará las líneas de error generadas por el Motor de Base de Datos.

La imagen siguiente muestra el resultado del script modificado:

```
OCURRIO UN ERROR AL ACTUALIZAR LA TABLA PASAJERO
```

Podríamos capturar el error en una variable de tipo entera, como se muestra en el siguiente script:

```
BEGIN TRY
    UPDATE PASAJERO
        SET IDPAIS='9999'
        WHERE PASAJERO.IDPASAJERO='P0010'
END TRY
```

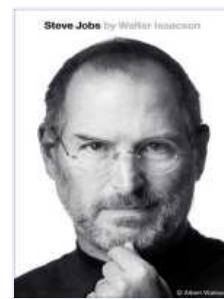
8 views

1 0

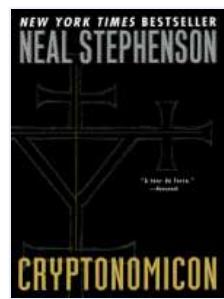
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

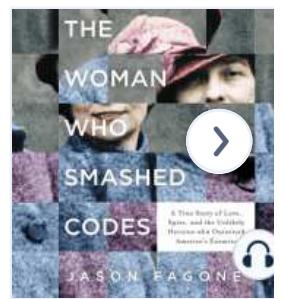
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

202

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Como verá en el script anterior el resultado es el mismo, ya que esta vez solo se captura el variable y esta variable es condicionada.

Para tener una lista de los errores controlados por @@ERROR podemos ejecutar el siguiente permitirá mostrar los 97526 errores que se podría ocasionar en una instrucción SQL.

```
SELECT *
FROM SYS.SYSPRECEDENCES
GO
```

La imagen a continuación muestra el catalogo sys.messages:

1	21	20	0	Warning: Fatal error %d occurred at %S_DATE. Note the error and time, and contact your system administrator.
2	101	15	0	Query not allowed in Waitfor.
3	102	15	0	Incorrect syntax near '%.1s'.
4	103	15	0	The %S_MSG that starts with "%.%e" is too long. Maximum length is %d.
5	104	15	0	ORDER BY items must appear in the select list if the statement contains a UNION, INTERSECT or EXCEPT operat
6	105	15	0	Unclosed quotation mark after the character string '%.1s'.
7	106	16	0	Too many table names in the query. The maximum allowable is %d.
8	107	15	0	The column prefix '%.1s' does not match with a table name or alias name used in the query.
9	108	15	0	The ORDER BY position number %d is out of range of the number of items in the select list.

4.12. FUNCIÓN RAISERROR

Se usa para devolver mensajes definidos por el usuario a las aplicaciones con el mismo formato de error del sistema o un mensaje de advertencia generado por el Motor de base de datos de Microsoft.

Sintaxis:

CONTROL DE ERRORES RAISERROR

```
RAISERROR ( msg_id | msg_str | @local_variable
,severity ,state
[ ,argument [ ,...n ] ] )
[ WITH option [ ,...n ] ]
```

Donde:

- **msg_id:** Es un número de mensaje de error definido por el usuario almacenado en **SYSMESSAGES** mediante el procedimiento almacenado **SP_ADDMESSAGE**. Los números de mensaje definidos por el usuario deben ser mayores que **50000** ya que este es el tope de los números asignados en forma nativa en Motor de Base de Datos de Microsoft. Si especifica msg_id, RAISERROR genera un mensaje de error con el número 50000.

8 views

1 like

0 comments

eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)

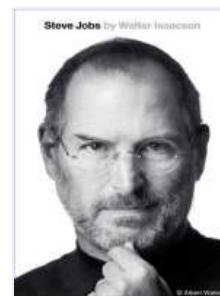
Save

Embed

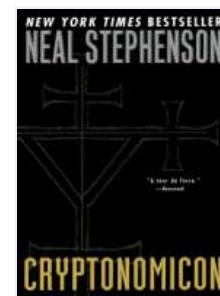
Share

Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

- severity: Es el nivel de gravedad definido por el usuario asociado a este mensaje. Cuan msg_id para generar un mensaje definido por el usuario creado mediante sp_add gravedad especificada en RAISERROR reemplaza la gravedad especificada en sp_addme

Todos los usuarios pueden especificar los niveles de gravedad del 0 al 18. Sólo los miembros de la función fija de servidor sysadmin o los usuarios con permisos ALTER TRACE pueden especificar los niveles de gravedad del 19 al 25. Para los niveles de gravedad del 19 al 25, se necesita WITH LOG.

- state: Es un número entero entre 0 y 255. Los valores negativos o mayores que 255 generan un error.

Consideraciones:

- RAISERROR permite asignar un número de error, gravedad y un estado específico.
- Puede solicitar que el error se guarde en el registro de errores del Motor de base de datos o en el registro de aplicación de Microsoft Windows.
- Puede sustituir valores de argumento en el texto del mensaje, de forma parecida a la función PRINT. Tanto RAISERROR como PRINT se pueden usar para devolver mensajes de información o a una aplicación.
- Cuando RAISERROR se usa con el msg_id de un mensaje definido por el usuario en sys.messages, msg_id se devuelve como el número de error de SQL Server o el código nativo del error. Si RAISERROR se usa con una msg_str en lugar de un msg_id, el número de error y el número de error de SQL Server devuelto es 50000.
- Cuando se usa RAISERROR para devolver un mensaje de error definido por el usuario, use un estado distinto en cada RAISERROR que haga referencia al error. Esto puede ayudar a los errores cuando se generan.

CASO DESARROLLADO N° 4.20

Script que permite mostrar el valor devuelto a partir de una división sin resultado como instrucción RAISERROR.

```

BEGIN TRY
    PRINT 1/0
END TRY
BEGIN CATCH
    DECLARE @MENSAJEERROR VARCHAR(4000)
    DECLARE @SEVERIDAD INT
    DECLARE @ESTADO INT

    SELECT
        @MENSAJEERROR = ERROR_MESSAGE()

```

8 views

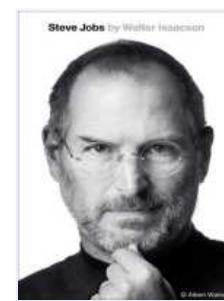
1 like

0 comments

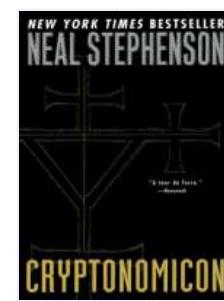
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

204

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el script se pretende dividir 1/0 el cual genera un error y entra a ejecución el bloque BEG donde se declaran variables locales que tendrán por misión obtener los parámetros del error de las funciones ERROR_MESSAGE(), ERROR_SEVERITY() y ERROR_STATE() que luego serán dentro de la instrucción RAISERROR .

Finalmente en la instrucción RAISERROR se imprimen los valores de las variables asignadas de la consulta:

```
SELECT
    @MENSAJEERROR = ERROR_MESSAGE(),
    @SEVERIDAD = ERROR_SEVERITY(),
    @ESTADO = ERROR_STATE()
```

La siguiente imagen muestra la ejecución del script:

Mens. 50000, Nivel 16, Estado 1, Línea 13
Error de división entre cero.

CASO DESARROLLADO N° 4.21

Script que permita eliminar un registro de la tabla RESERVA y que muestre un mensaje de error así ocurriera caso contrario también mostrar un mensaje de ELIMINACION CORRECTA. Usando los casos la instrucción RAISERROR.



```
BEGIN TRY
    DECLARE @IDRES INT=1
    DELETE RESERVA WHERE IDRESERVA=@IDRES
    RAISERROR('ELIMINACION CORRECTA',10,1)
END TRY
BEGIN CATCH
    RAISERROR('NO SE PUEDE ELIMINAR LA RESERVA',16,0)
END CATCH
GO
```

En el script se implementa la eliminación de la reserva cuyo código es 1 desde el bloque BEGIN TRY. Aquí mismo se adiciona la instrucción RAISERROR que permitirá devolver el mensaje de ELIMINACION CORRECTA como lo solicita el caso. La imagen siguiente muestra cual sería el resultado si el comando DELETE se ejecuta correctamente:

8 views

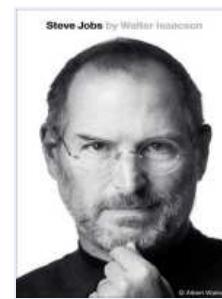
1 like

0 comments

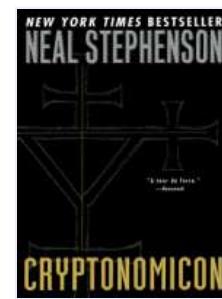
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

Veamos el siguiente script con el cambio en la severidad inicial de 16 por 10 para mostrar de error:

```
BEGIN TRY
    DECLARE @IDRES INT=1
    DELETE RESERVA WHERE IDRESERVA=@IDRES
    RAISERROR('ELIMINACION CORRECTA',10,1)
END TRY
BEGIN CATCH
    RAISERROR('NO SE PUEDE ELIMINAR LA RESERVA',10,0)
END CATCH
```

La siguiente imagen muestra el resultado del script donde la respuesta del sistema presenta definido por el usuario:

(0 filas afectadas)
NO SE PUEDE ELIMINAR LA RESERVA

4.13. IMPLEMENTACIÓN DE CURSORES

Hasta el momento hemos visto que la sentencia SELECT ha trabajado para mostrar filas de una determinada tabla o mostrando algún valor agrupado o asignando valores a las variables pero usted ya debe tener en cuenta que esta instrucción controla a los registros de una tabla matricial es por eso que en algún momento que usted desea almacenar información descubrió que declarar una variable de tipo TABLE. Pero a pesar de eso hasta ahora no se ha visto de los registros fila por fila ya que para esto tendríamos que recorrer todo el conjunto enviado desde SELECT y tratar de bajarlos a variables locales pero resultaría bastante trabajar de esa manera, para esto contamos con CURSORES dentro de Transact SQL que nos permiten tener el control fila por fila de un conjunto de registros.

Esto se debe a que hay algunas aplicaciones en línea que no pueden controlar información fila por fila. Estas aplicaciones necesitan un mecanismo que trabaje con una fila o un pequeño bloque de vez. Los cursores son una extensión de los conjuntos de resultados que proporcionan dicho control.

Y en qué casos usaremos cursores?, puesto que si ya tenemos acceso a las filas de una tabla entonces podremos obtener dichos valores y controlarlos desde variables locales y así podemos obtener los valores, contarlos, aplicar quiebres, etc.

Consideraciones:

- Los cursores permiten situarse en filas específicas del conjunto de resultados.
- Recuperan una fila o un bloque de filas de la posición actual en el conjunto de resultados.

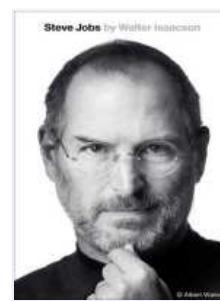
8 views

1 0

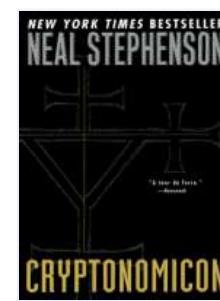
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

206 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Proceso de Implementación de un Cursor

Para implementar un cursor se necesitan pasos específicos siempre será recomendado im dentro de un procedimiento almacenado pero eso lo veremos cuando desarrollemos el te procedure de este mismo capítulo:

1. Declarar el cursor y asignar un conjunto de resultados provenientes de la instrucción **SELECT**.
2. Aperturar el cursor para que se active; use el operador **OPEN**.
3. Recuperar las filas del cursor que desea visualizar y usar la estructura repetitiva **WHILE** trabajo, aquí es el momento de usar la variable global **@@FETCH_STATUS**.
4. Tenga en cuenta que se puede realizar operaciones de actualización o eliminación en posición actual del cursor.
5. Cerrar el cursor usando el operador **CLOSE**.
6. Liberar el cursor usando el operador **DEALLOCATE**.

Sintaxis de la declaración de un CURSOR:

FORMATO DE DECLARACION DE UN CURSOR

```
DECLARE NOMBRE_CURSOR [ INSENSITIVE ] [ SCROLL ] CURSOR
FOR EXPRESION_SELECT
[ FOR READ ONLY | UPDATE [ OF column_name [ ,...n ] ] ]
```

Donde:

- **NOMBRE_CURSOR:** Es el nombre definido al cursor.
- **INSENSITIVE:** Define un cursor que hace una copia temporal de los datos que utiliz solicitudes que se realizan al cursor se responden desde esta tabla temporal de tanto, las modificaciones realizadas en las tablas base no se reflejan en los datos devu operaciones de recuperación realizadas en el cursor y además este cursor no admite mod
- **SCROLL:** Especifica que están disponibles todas las opciones de recuperación (FIRST, LAST, NEXT, RELATIVE, ABSOLUTE). Si no se especifica **SCROLL** en una instrucción **DECLARE** única opción de recuperación que se admite es **NEXT**.
- **READ ONLY:** Evita que se efectúen actualizaciones a través de este cursor. No es p referencia al cursor en una cláusula WHERE CURRENT OF de una instrucción UPDATE o I opción reemplaza la capacidad predeterminada de actualizar el cursor.
- **UPDATE [OF column_name [,...n]]:** Define las columnas actualizables en el cursor. Si s OF column_name [,...n], sólo las columnas enumeradas admiten modificaciones. Si s UPDATE sin indicar una lista de columnas, se pueden actualizar todas las columnas.

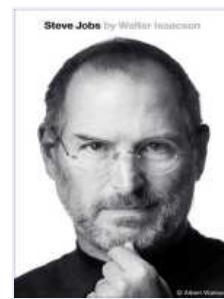
8 views

1 0

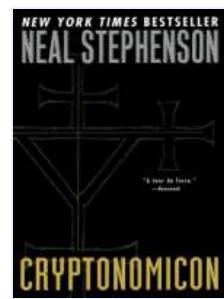
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

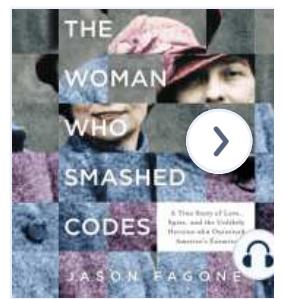
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

Sintaxis de la recuperacion de registros desde el CURSOR:

**RECUPERACION DE FILAS
FETCH**

```

FETCH
  [ [ NEXT | PRIOR | FIRST | LAST
      | ABSOLUTE N | @NVAR
      | RELATIVE N | @NVAR
    ]
  FROM
  ]
  [ GLOBAL ] NOMBRE_CURSOR | @VARIABLECURSOR
  [ INTO @NOMBRE_VARIABLE[ ,...N ] ]

```

Donde:

- **NEXT:** Devuelve la fila de resultados inmediatamente posterior a la fila actual, y actual a la fila devuelta. Si FETCH NEXT es la primera operación de recuperación en un cursor, devuelve la primera fila del conjunto de resultados. NEXT es la opción predeterminada para la recuperación de cursos.
- **PRIOR:** Devuelve la fila de resultados inmediatamente anterior a la fila actual, y reduce la fila devuelta. Si FETCH PRIOR es la primera operación de recuperación en un cursor, devuelve ninguna fila y el cursor queda posicionado delante de la primera fila.
- **FIRST:** Devuelve la primera fila del cursor y la convierte en la fila actual.
- **LAST:** Devuelve la última fila del cursor y la convierte en la fila actual.
- **ABSOLUTE N | @NVAR :** Si n o @nvar es positivo, se devuelve la fila n desde el principio y la fila devuelta se convierte en la nueva fila actual. Si N o @NVAR es negativo, se devuelven filas anteriores al final del cursor y la fila devuelta se convierte en la nueva fila actual. Si N o @nvar es 0, no se devuelven filas.
- **RELATIVE N | @NVAR:** Si n o @nvar es positivo, se devuelve la fila n posterior a la fila actual y la fila devuelta se convierte en la nueva fila actual. Si n o @nvar es negativo, se devuelven filas anteriores a la fila actual y la fila devuelta se convierte en la nueva fila actual. Si n o @nvar es 0, se devuelve la fila actual. Si se especifica FETCH RELATIVE con n o @nvar establecido en un número negativo o en 0 en la primera operación de recuperación que se realiza en un cursor, no se devuelven filas.

CASO DESARROLLADO N° 4.22

Script que permita implementar un cursor básico donde se imprima el primer registro PASAJERO.

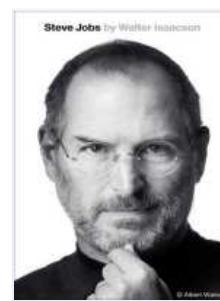
8 views

1 0

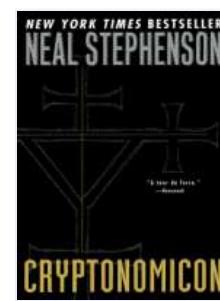
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

208

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

```
--1.
DECLARE MICURSORBAS CURSOR
      FOR SELECT * FROM PASAJERO
--2.
OPEN MICURSORBAS
--3.
FETCH NEXT FROM MICURSORBAS
```

En el punto uno se declara la variable **MICURSORBAS** de tipo **CURSOR** donde almacena de la consulta a los pasajeros.

En el punto dos se apertura el cursor **MICURSORBAS** no hay mucho que especificar aquí es la apertura.

En el punto tres se accede al primer registro del cursor con el operador **NEXT**, el operador **F** desde que cursor se obtienen las filas. El resultado de la ejecución de este script simple es

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM

Si desea visualizar la información de la siguiente fila tendrá que ejecutar solo la siguiente línea

FETCH NEXT FROM MICURSORBAS

El resultado se muestra en la siguiente imagen, recuerde que esto se debe a que el cursor abierto:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM

Si intenta volver a ejecutar el script presentado al inicio se generaría el siguiente error:

```
Mens. 16915, Nivel 16, Estado 1, Línea 2
  Ya existe un cursor con el nombre 'MICURSORBAS'.
Mens. 16906, Nivel 16, Estado 1, Línea 4
  El cursor ya está abierto.

(1 filas afectadas)
```

Sintaxis de cierre de CURSOR: Cierra un cursor abierto mediante la liberación del conjunto de actual y todos los bloqueos de cursor mantenidos en las filas en las que está colocado. las estructuras de datos accesibles para que se puedan volver a abrir, pero las recuperaciones actualizaciones posicionadas no se permiten hasta que se vuelva a abrir el cursor. CLOSE debe ejecutarse en un cursor abierto, por lo que no se permite en cursos que sólo están declarados o que ya estén cerrados.

CERRANDO UN

8 views

1 like

0 comments

eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)

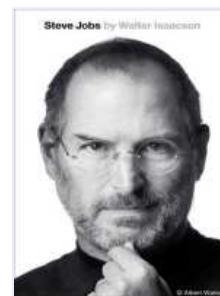
Save

Embed

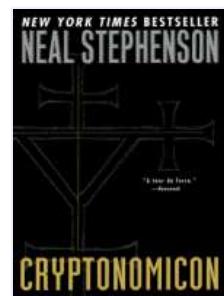
Share

Print

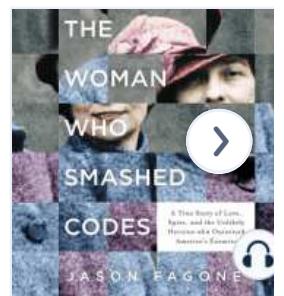
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True

CAP. 4: PROGRAMACIÓN TRANSACT S

CASO DESARROLLADO N° 4.23

Script que permita implementar un cursor básico donde se imprima el primer registro PASAJERO.



```
DECLARE MICURSORBAS CURSOR
    FOR SELECT * FROM PASAJERO
OPEN MICURSORBAS
FETCH NEXT FROM MICURSORBAS
CLOSE MICURSORBAS
DEALLOCATE MICURSORBAS
GO
```

En el script se declara la variable **MICURSORBAS** que tiene como filas los registros de la tabla **PASAJERO**. La declaración se realiza con la instrucción **DECLARE**. Luego se tiene que abrir el cursor con el operador **OPEN**, seguidamente se obtiene el primer registro desde el cursor usando **FETCH**, finalmente se cierra el cursor con **CLOSE** y se libera con el operador **DEALLOCATE**. Se podrá ejecutarlo las veces que quiera pero todo en bloque para que no ocasione algún error. También es ejecutable por bloques por la siguiente forma:

```
DECLARE MICURSORBAS CURSOR
    FOR SELECT * FROM PASAJERO
GO

OPEN MICURSORBAS
GO

FETCH NEXT FROM MICURSORBAS
GO

CLOSE MICURSORBAS
GO

DEALLOCATE MICURSORBAS
GO
```

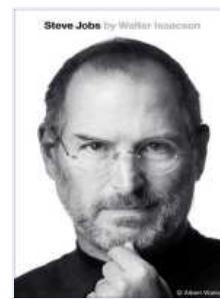
8 views

1 0

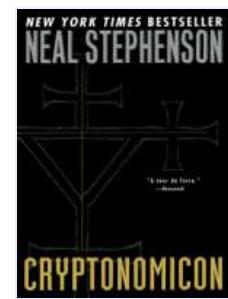
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon

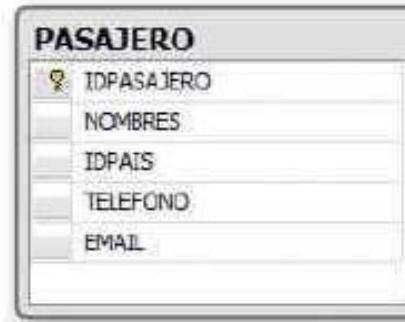


The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

210 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 4.24

Script que permita implementar un cursor donde imprima en forma de reporte a los re tabla PASAJERO. Debe adicionar el nombre del país a la consulta.



```
--1.
DECLARE @IDPA CHAR(5),@NOM CHAR(30),@PAI CHAR(30),
        @TEL CHAR(15),@EMA VARCHAR(40)
--2.
DECLARE MICURSOR CURSOR
        FOR SELECT PAS.IDPASAJERO,PAS.NOMBRES,
PAI.PAIS,PAS.TELEFONO,PAS.EMAIL
                FROM PASAJERO PAS
                JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
--3.
OPEN MICURSOR
--4.
FETCH MICURSOR INTO @IDPA,@NOM,@PAI,@TEL,@EMA
PRINT 'CODIGO      PASAJERO          PAIS          TELEFONO          EMAIL'
PRINT '-----'
--5.
WHILE @@FETCH_STATUS=0
BEGIN
        PRINT @IDPA+SPACE(5)+@NOM+SPACE(5)+@PAI+SPACE(5)+@TEL+SPACE(5)+@EMA
        FETCH MICURSOR INTO @IDPA,@NOM,@PAI,@TEL,@EMA
END
--6.
CLOSE MICURSOR
DEALLOCATE MICURSOR
GO
```

En el punto uno se declara las variables locales que representaran el valor de las col consulta, como notara se ha declarado como CHAR a columnas que en la tabla son VAF es para que se mantenga una estética al mostrar el informe ya que al ser dinámico el VARCHAR descuadra el informe cuando el valor de la columna contiene un texto largo.

En el punto dos se declara el cursor con la consulta a la tabla PASAJERO y PAÍS ya que obtener el nombre del país, para imprimirlo en la consulta.

En el punto tres se apertura el cursor.

8 views

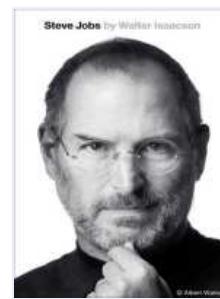
1 like

0 comments

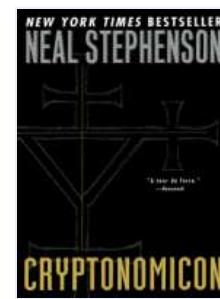
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Hidden Heroine Behind a Conspiracy

CAP. 4: PROGRAMACIÓN TRANSACT S

La imagen siguiente muestra la cabecera implementada:

CODIGO	PASAJERO	PAÍS	TELÉFONO	EMAIL
--------	----------	------	----------	-------

En el punto cinco se implementa la estructura repetitiva WHILE para poder imprimir todos asignados al cursor para poder determinar los ciclos de repeticiones se tiene que evaluar @@FETCH_STATUS de acuerdo al valor emitido el ciclo ejecutara, veamos los valores posibles de FETCH_STATUS:

0	Al evaluar el FETCH no se encontró error alguno.
-1	La instrucción FETCH no se ejecutó correctamente o la fila estaba más allá del conjunto de resultados.
-2	Falta la fila recuperada.

Entonces el WHILE recorrerá mientras que la función @@FETCH_STATUS mantenga el dentro de este ciclo se imprime el reporte con la función PRINT aquí se especificaran las variables que participaran en el reporte, observe que se usa la función SPACE(5) para poder espaciar una linea de otra en el reporte final.

En el punto seis se cierra el cursor con el operador CLOSE y se libera el recurso con DEALLOCATE, la imagen siguiente muestra el resultado del cursor:

CODIGO	PASAJERO	PAÍS	TELÉFONO	EMAIL
P0001	ANGELA TORRES LAZARO	PERU	999999999	ATORRES@HOTMAIL.COM
P0002	FERNANDA TORRES LAZARO	PERU	999999999	FTORRES@HOTMAIL.COM
P0003	MARIA ZAMORA MEJIA	BRAZIL	967664526	MZAMORA@GMAIL.COM
P0004	GUADALUPE ACOSTA FERRER	ARGENTINA	967564526	GACOSTA@HOTMAIL.COM
P0005	LIZ LAZARO MENOS	PERU	999999999	LLAZARO@GMAIL.COM
P0006	KARLA GALLEGOS SILVA	PAPAGUAY	967564526	KGALLEGOS@HOTMAIL.COM
P0007	HERY CALLE DE LA CRUZ	MEXICO	967564526	HSCALLE@GMAIL.COM
P0008	HEIDI MENCINO MEATEGUI	ECUADOR	967564526	HMENCIPO@GMAIL.COM
P0009	MARISOL DIAZ ZAMBRANO	ECUADOR	967564526	MDIAZ@GMAIL.COM
P0010	GUADALUPE ACOSTA FERRER	CHILE	966475747	GACOSTA@GMAIL.COM

Si además del reporte mostrado quisieramos implementar al final del reporte el total de registros, se tendrá que colocar el siguiente script:

```
--1.
DECLARE @IDPA CHAR(5),@NOM CHAR(30),@PAI CHAR(30),
        @TEL CHAR(15),@EMA VARCHAR(40),@TOTAL INT=0
--2.
DECLARE MICURSOR CURSOR
        FOR SELECT PAS.IDPASAJERO,PAS.NOMBRES,PAI.PAIS,PAS.TELEFONO,PAS.EMAIL
        FROM PASAJERO PAS
        JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
--3.
OPEN MICURSOR
--4.
FETCH MICURSOR INTO @IDPA,@NOM,@PAI,@TEL,@EMA
PRINT 'CODIGO      PASAJERO          PAIS          TELEFONO          EMAIL'
PRINT '-----'
--5.
WHILE @@FETCH_STATUS=0
```

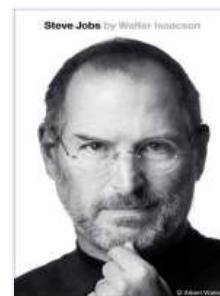
8 views

 1 0

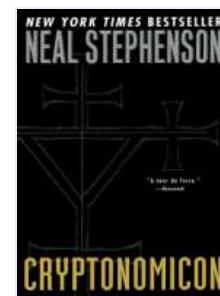
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by **anon_61286589**

Full description



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A Tri

 Save  Embed  Share  Print

212 | PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el punto uno se adiciona la declaración de la variable **@TOTAL** inicializada en cero para el conteo de los pasajeros de la siguiente forma **@TOTAL INT=0**.

En el punto cinco se adiciona el contador de la variable @TOTAL aumentando en uno por cada estructura WHILE con **SET @TOTAL+=1**.

En el punto seis se adiciona la función **PRINT** que tiene por misión mostrar el total obtenido del conteo.

La imagen muestra el reporte de los pasajeros además del total al final de la línea del reporte.

CODIGO	PASAJERO	PAIS	TELEFONO	EMAIL
P0001	ANGELA TORRES LAZARO	PERU	999999999	ATOPRES@HOTMAIL.COM
P0002	FERNANDA TORRES LAZARO	PERU	999999999	FTOPRES@HOTMAIL.COM
P0003	MARIA ZAMORA MEJIA	BRAZIL	967564526	MZAMORA@GMAIL.COM
P0004	GUADALUPE ACOSTA FERRER	ARGENTINA	967564526	GACOSTA@HOTMAIL.COM
P0005	LIZZ LASARO MENOR	PERU	999999999	LLAZARO@GMAIL.COM
P0006	KARLA CALLEGOS SILVA	PARAGUAY	967564526	KCALLEGOS@HOTMAIL.COM
P0007	MERY CALLE DE LA CRUZ	MEXICO	967564526	MCALLE@GMAIL.COM
P0008	HEIDI RENCIPIO REATEGUI	ECUADOR	967564526	HRENCIPIO@HOTMAIL.COM
P0009	MARISOL DIAZ LAMBANANO	ECUADOR	967564526	MDIAZ@GMAIL.COM
P0010	GUADALUPE ACOSTA FERRER	CHILE	990478787	GACOSTA@GMAIL.COM

CASO DESARROLLADO N° 4.25

Script que permita implementar un cursor que reporte el código, nombre del pasajero y solo aquellos servidores ingresados por el usuario en una variable local.

PASAJERO
IDPASAJERO
NOMBRES
IDPAIS
TELEFONO
EMAIL

```
--1.  
DECLARE @IDPA CHAR(5),@NOM CHAR(30),  
        @EMA CHAR(40),@TOTAL INT=0,@SERVIDOR VARCHAR(30)='HOTMAIL'  
--2.  
DECLARE MICURSOR CURSOR  
    FOR SELECT PAS.IDPASAJERO,PAS.NOMBRES,PAS.EMAIL  
          FROM PASAJERO PAS  
         WHERE PAS.EMAIL LIKE '%' +@SERVIDOR+'%'  
--3.  
OPEN MICURSOR  
--4.  
FETCH MICURSOR INTO @IDPA,@NOM,@EMA  
PRINT 'CODIGO      PASAJERO                      EMAIL'  
PRINT '-----'  
--5.  
WHILE @@FETCH STATUS=0
```

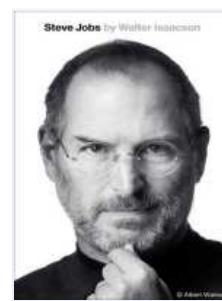
8 views

1 0

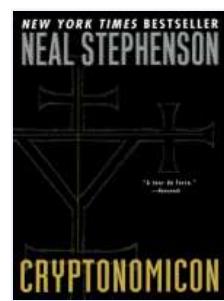
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Hidden Heroine Behind a Conspiracy

En el script se adiciono la variable **@SERVIDOR** inicializada con **HOTMAIL** para almacenar el de dicho correo.

En el punto dos se especifican solo tres columnas de la tabla **PASAJERO** condicionado por **EMAIL** para solo almacenar dentro del cursor a los pasajeros cuyo email sea el valor de la **SERVIDOR** en este caso mostrara los pasajeros que tienen como servidor de correo a Hotmail.

En la siguiente imagen se muestra el reporte ejecutado por el cursor:

CODIGO	PASAJERO	EMAIL
P0001	ANGELA TORRES LAZARO	ATORRES@HOTMAIL.COM
P0002	FERNANDA TORRES LAZARO	FTORRES@HOTMAIL.COM
P0004	GUADALUPE ACOSTA FERRER	GACOSTA@HOTMAIL.COM
P0006	KARLA GALLEGOS SILVA	KGALLEGOS@HOTMAIL.COM
P0008	HEIDI RENGIFO REATEGUI	HRENGIFO@HOTMAIL.COM
EL TOTAL DE PASAJEROS CON SERVIDOR HOTMAIL ES:5		

Si se cambia el valor de la variable **@SERVIDOR** por **GMAIL** el resultado sería:

CODIGO	PASAJERO	EMAIL
P0003	MARIA ZAMORA MEJIA	MZAMORA@GMAIL.COM
P0005	LUZ LAZARO MENOR	LLAZARO@GMAIL.COM
P0007	NERY CALLE DE LA CRUZ	NCALLE@GMAIL.COM
P0009	MARISOL DIAZ ZAMBRANO	MDIAZ@GMAIL.COM
P0010	GUADALUPE ACOSTA FERRER	GACOSTA@GMAIL.COM
EL TOTAL DE PASAJEROS CON SERVIDOR GMAIL ES:5		

CASO DESARROLLADO N° 4.26

Script que permita implementar un cursor donde imprima el siguiente reporte:

PASAJERO	AÑO	TOTAL
AÑO: 2011		
ANGELA TORRES LAZARO	2011	1
FERNANDA TORRES LAZARO	2011	1
HEIDI RENGIFO REATEGUI	2011	1
KARLA GALLEGOS SILVA	2011	1
LUZ LAZARO MENOR	2011	1
MARIA ZAMORA MEJIA	2011	1
EL TOTAL DE PASAJEROS DEL AÑO 2011 ES: 6		
AÑO: 2012		
FERNANDA TORRES LAZARO	2012	1
HEIDI RENGIFO REATEGUI	2012	1
KARLA GALLEGOS SILVA	2012	1
MARIA ZAMORA MEJIA	2012	3
EL TOTAL DE PASAJEROS DEL AÑO 2012 ES: 7		

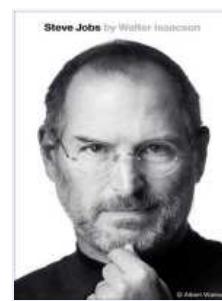
8 views

1 0

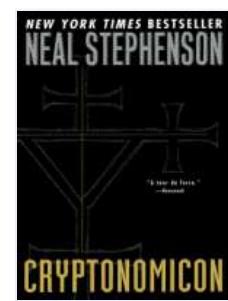
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

214

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Observe que se debe quebrar los registros por año de reserva y por cada bloque emitir pasajeros por ese año. Al final imprimir el total acumulado. Use las siguientes tablas:



```

--1.
DECLARE @NOM CHAR(30),@AÑO INT,@TOTAL INT,
        @CAÑO INT=0,@AAÑO INT=0,@ACTUAL INT
--2.
DECLARE MICURSOR CURSOR
FOR SELECT PAS.NOMBRES,YEAR(RES.FECHA),COUNT(*)
FROM RESERVA RES
JOIN PAGO PAG ON RES.IDRESERVA=PAG.IDRESERVA
JOIN PASAJERO PAS ON PAS.IDPASAJERO=PAG.IDPASAJERO
GROUP BY PAS.NOMBRES,YEAR(RES.FECHA)

--3.
OPEN MICURSOR
--4.
FETCH MICURSOR INTO @NOM,@AÑO,@TOTAL
SET @ACTUAL=@AÑO
PRINT 'PASAJERO' AÑO TOTAL'
PRINT '-----'
PRINT 'AÑO:' +CAST(@AÑO AS CHAR(4))

--5.
WHILE @@FETCH_STATUS=0
BEGIN
    SET @AAÑO+=@TOTAL
    IF @AÑO=@ACTUAL
    BEGIN
    
```

8 views

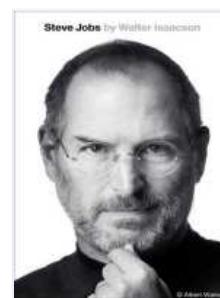
1 like

0 comments

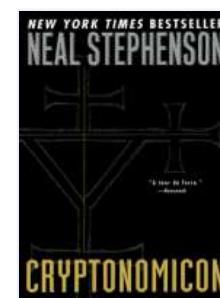
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

CAP. 4: PROGRAMACIÓN TRANSACT S

```

PRINT ''
PRINT 'AÑO:'+CAST(@AÑO AS CHAR(4))
SET @ACTUAL=@AÑO
SET @CAÑO=@TOTAL
END
FETCH MICURSOR INTO @NOM,@AÑO,@TOTAL
END
--6.
PRINT 'EL TOTAL DE PASAJEROS DEL AÑO '+
      CAST(@AÑO AS CHAR(4))+'' ES '+
      CAST(@CAÑO AS CHAR(6))
PRINT 'EL TOTAL ACUMULADO ES '+CAST(@AAÑO AS CHAR(6))
CLOSE MICURSOR
DEALLOCATE MICURSOR
GO

```

En el punto uno se declaran las variables locales **@NOM**, **@AÑO**, **@TOTAL** que permitirán el nombre del pasajero, el año de su reserva y el total de reservas de dicho cliente respeto también se declaran **@CAÑO INT=0** para contabilizar las reservas por año, **@AAÑO** acumular el total en todos los años y **@ACTUAL** que permitirá permutar entre los años e en la tabla **RESERVA**. Recuerde que todo contador o acumulador siempre se deben inicializar.

En el punto dos se declara el cursor **MICURSOR** y se almacena la consulta **SELECT PAS.NOMBRE, FECHA, COUNT(*)** proveniente de las tablas **PASAJERO, PAGO Y RESERVA**.

En el punto cuatro se captura el primer registro del cursor con **FETCH MICURSOR INTO @NOM, @AÑO, @TOTAL** hasta aquí el cursor ya tiene valores registrados en las variables **@NOM, @AÑO** por lo tanto ahora se puede capturar el año y enviarlo a la variable **@ACTUAL** con **SET @ACTUAL=@AÑO**, seguidamente se imprime las cabeceras del informe y a la vez se imprime el año no sin aplicarle la función **CAST** a dicha variable.

En el punto cinco se implementa la estructura repetitiva **WHILE** para poder controlar el acuerdo a la cantidad de filas que tiene el cursor. Luego se acumula el valor de la columna siguiente forma **SET @AAÑO+=@TOTAL** esta variable tendrá el acumulado de los totales para ser 13 que representa la suma de todos los totales del cursor este valor no debe estar controlado. Luego se condiciona la variable **@AÑO** comparando si se parece al año establecido en **@AAÑO** que si se cumple con la condición tendrá que imprimir a las filas coincidentes y así genera un bloque de filas correspondientes al primer año encontrado e imprimirlas además de acumular los totales de ese año con **SET @CAÑO+=@TOTAL**.

En caso la comparación del **@AÑO** con lo asignado en **@ACTUAL** no sean iguales querrá decir que las filas están mostrando un año distinto y antes de actualizar las variables para el siguiente año se debe imprimir el resumen de este primer año con el siguiente script:

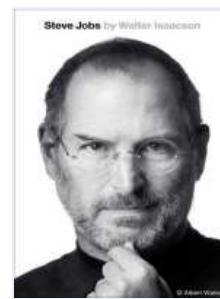
8 views

1 0

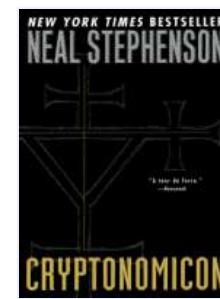
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

216 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Luego se actualizan los valores de las variables involucradas en @ACTUAL y @CAÑO ya que valores frente a las filas anteriores mostradas, todo esto se realiza con SET @ACTUAL=@CAÑO=@TOTAL el primero cambia el año actual por el nuevo año encontrado desde la fila el segundo actualiza el @CAÑO actualiza con el primer valor encontrado en la fila nueva primer @TOTAL.

Finalmente saliendo de la condicional se debe obtener el siguiente registro, observe que no tiene condición ya que cumpla o no siempre se debe avanzar a la siguiente fila del cursor con el comando MICURSOR INTO @NOM,@AÑO,@TOTAL.

En el punto seis se imprime el reporte del último año encontrado además del total acumulado de repeticiones además de cerrar el cursor y liberarlo. Todo eso se implementa con el siguiente script:

```

PRINT 'EL TOTAL DE PASAJEROS DEL AÑO '+
      CAST(@ACTUAL AS CHAR(4))+'' ES '+
      CAST(@CAÑO AS CHAR(6))
PRINT 'EL TOTAL ACUMULADO ES '+CAST(@AAÑO AS CHAR(6))
CLOSE MICURSOR
DEALLOCATE MICURSOR
GO

```

A continuación se muestra el reporte que imprime el cursor al ejecutarse.

PASAJERO	AÑO	TOTAL
AÑO: 2011		
ANGELA TORRES LAZARO	2011	1
FERNANDA TORRES LAZARO	2011	1
HEIDI RENGIFO REATEGUI	2011	1
MARIA GALLEGOS SILVA	2011	1
LUZ LAZARO MENOR	2011	1
MARIA ZAMORA MEJIA	2011	1
EL TOTAL DE PASAJEROS DEL AÑO 2011 ES 6		
AÑO: 2012		
FERNANDA TORRES LAZARO	2012	1
HEIDI RENGIFO REATEGUI	2012	1
MARIA GALLEGOS SILVA	2012	1
MARIA ZAMORA MEJIA	2012	3
EL TOTAL DE PASAJEROS DEL AÑO 2012 ES 7		
EL TOTAL ACUMULADO ES 13		

CASO DESARROLLADO N° 4.27

Script que permita implementar un cursor donde imprima el siguiente reporte:

CODIGO	PASAJERO	PAÍS	EMAIL
SERVIDOR: GMAIL			
P0003	MARIA ZAMORA MEJIA	BRASIL	MZAMORA@GMAIL.COM
P0005	LUZ LAZARO MENOR	PERU	LLAZARO@GMAIL.COM

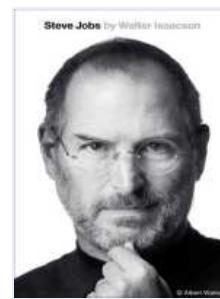
8 views

1 0

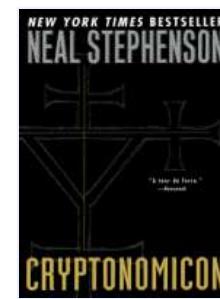
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

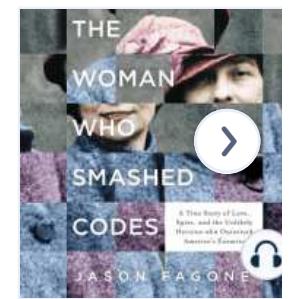
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

Considere que el quiebre se encuentra por cada servidor de correo. Las tablas involucradas son:

PASAJERO	
IDPASAJERO	
NOMBRES	
IDPAIS	
TELEFONO	
EMAIL	

PAIS	
Column Name	Condensed Type
IDPAIS	char(4)
PAIS	varchar(30)

```
--1.
DECLARE @IDPAS CHAR(5),@NOM CHAR(30),@PAIS CHAR(20),
        @EMAI CHAR(30),@SERVIDOR CHAR(15),
        @SERV CHAR(15),@CSERV INT=0,@CTOTAL INT=0
--2.
DECLARE MICURSOR CURSOR
      FOR SELECT      PAS.IDPASAJERO,PAS.NOMBRES,PAI.PAIS,PAS.EMAIL,
                      CASE      WHEN EMAIL LIKE '%HOTMAIL%' THEN 'HOTMAIL'
                                WHEN EMAIL LIKE '%GMAIL%' THEN 'GMAIL'
                                ELSE 'OTRO'
                      END AS [SERVIDOR DE CORREO]
      FROM PASAJERO PAS
      JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
      ORDER BY [SERVIDOR DE CORREO]

--3.
OPEN MICURSOR
--4.
FETCH MICURSOR INTO @IDPAS,@NOM,@PAIS,@EMAI,@SERVIDOR
SET @SERV=@SERVIDOR
PRINT 'CODIGO      PASAJERO          PAIS          EMAIL'
PRINT '-----'
PRINT 'SERVIDOR:' +@SERV

--5.
WHILE @@FETCH_STATUS=0
BEGIN
      SET @CTOTAL+=1
      IF @SERVIDOR=@SERV
      BEGIN
            PRINT @IDPAS+SPACE(5)+@NOM+SPACE(5)+@PAIS+@EMAI
            SET @CSERV+=1
      END
      ELSE
      BEGIN
            PRINT 'EL TOTAL DE PASAJEROS CON SERVIDOR '+ @SERV+' ES '+
                  CAST(@CSERV AS CHAR(4))
            PRINT ''
            PRINT 'SERVIDOR:' +@SERVIDOR
            SET @SERV=@SERVIDOR
            SET @CTOTAL=@CSERV
            SET @CSERV=0
      END
END
```

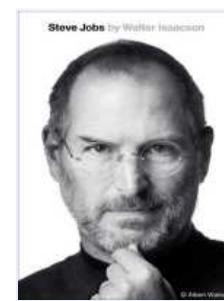
8 views

1 0

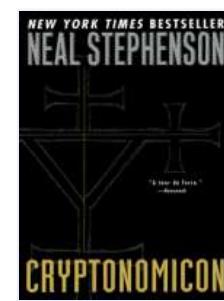
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

218 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el punto uno se declaran las variables que representaran a las columnas de la consulta las variables que permitirán almacenar el servidor actual (@SERV), el contador de dicho CSERV) inicializado en cero por ser un contador y el contador total (@CTOTAL) inicializada cero.

```
DECLARE @IDPAS CHAR(5),@NOM CHAR(30),@PAIS CHAR(20),
        @EMAI CHAR(30),@SERVIDOR CHAR(15),
        @SERV CHAR(15),@CSERV INT=0,@CTOTAL INT=0
```

En el punto dos se declara el cursor MICURSOR como notará en el caso anterior tambi igual pero como cada cursor cierre y libera entonces ese nombre no traerá problema alg del cursor se almacena la consulta en la cual se debe adicionar la columna servidor a la cc se realiza con la estructura CASE donde separa a HOTMAIL, GMAIL y OTROS para que lc se filtren por dicha columna recuerde que el cursor captura las nuevas columnas de la exclusivamente columnas de la tabla. Hay que tener en cuenta que dicha columna debe o que de otra forma emitiría un reporte con duplicidades. Observe el ORDER BY que hace r nombre de la cabecera definida en el CASE.

```
DECLARE MICURSOR CURSOR
      FOR SELECT      PAS.IDPASAJERO,PAS.NOMBRES,PAI.PAIS,PAS.EMAIL,
                      CASE      WHEN EMAIL LIKE '%HOTMAIL%' THEN 'HOTMAIL'
                                WHEN EMAIL LIKE '%GMAIL%' THEN 'GMAIL'
                                ELSE 'OTRO'
                      END AS [SERVIDOR DE CORREO]
      FROM PASAJERO PAS
      JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
      ORDER BY [SERVIDOR DE CORREO]
```

En el punto cuatro se obtiene el primer registro del cursor almacenándolo en las variables p declaradas. Se actualiza el servidor actual con el servidor obtenido del cursor. Y luego se i cabecera del reporte.

```
FETCH MICURSOR INTO @IDPAS,@NOM,@PAIS,@EMAI,@SERVIDOR
SET @SERV=@SERVIDOR
PRINT 'CODIGO      PASAJERO          PAIS          EMAIL'
PRINT '-----'
PRINT 'SERVIDOR:' +@SERV
```

En el punto cinco se entra el ciclo repetitivo para obtener los demás registros del curso lugar deberá comenzar a contar los registros:

8 views

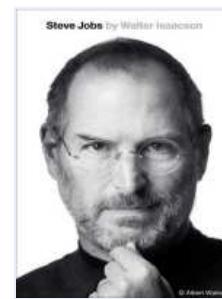
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

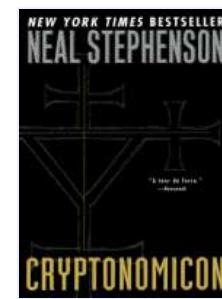
Uploaded by [anon_61286589](#)[Full description](#)

 Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

Luego se condicionara el tipo de servidor encontrado por cada registro del cursor si estos se imprimirán en el informe del primer bloque de coincidencias además de contabilizar registros.

```
IF @SERVIDOR=@SERV
BEGIN
    PRINT @IDPAS+SPACE(5)+@NOM+SPACE(5)+@PAIS+@EMAI
    SET @CSERV+=1
END
```

En caso contrario el servidor no sea el mismo encontrado en el cursor, entonces imprimir encontrados en dicho servidor además de actualizar las variables para el siguiente bloque a imprimir, no se olvide que el FETCH debe ubicarse fuera de la condicional ya que la o registros del cursor no está condicionado a nada.

```
ELSE
BEGIN
    PRINT 'EL TOTAL DE PASAJEROS CON SERVIDOR '+@SERV+' ES '+
          CAST(@CSERV AS CHAR(4))
    PRINT ''
    PRINT 'SERVIDOR:' +@SERVIDOR
    SET @SERV=@SERVIDOR
    SET @CTOTAL=@CSERV
    SET @CSERV=0
END
FETCH MICURSOR INTO @IDPAS,@NOM,@PAIS,@EMAI,@SERVIDOR
END
```

En el punto seis se imprimen los valores finales del reporte en la cual esta involucradc pasajeros con el ultimo servidor. No olvidarse que tiene que cerrar y liberar al cursor.

```
PRINT 'EL TOTAL DE PASAJEROS CON SERVIDOR '+@SERV+' ES '+
      CAST(@CSERV AS CHAR(4))
PRINT 'EL TOTAL DE PASAJEROS ES '+CAST(@CTOTAL AS CHAR(4))
CLOSE MICURSOR
DEALLOCATE MICURSOR
GO
```

El resultado de la ejecución del cursor se muestra en la siguiente imagen:

CÓDIGO	PASAJERO	PAÍS	EMAIL
SERVIDOR: GMAIL			
P0003	MARIA ZAMORA MEJIA	BRASIL	MZAMORA@GMAIL.COM
P0005	LUIS LAZARO MENOR	PERU	LLAZARO@GMAIL.COM

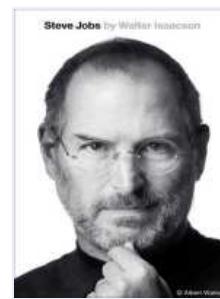
8 views

1 0

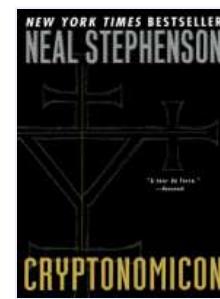
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

220 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 4.28

Script que permita implementar un cursor donde imprima el primer, quinto y último registro Pasajero. Use First, Absolute y Last para el informe.



```
--1.
DECLARE @N INT=5
DECLARE MICURSOR CURSOR SCROLL
    FOR SELECT * FROM PASAJERO
--2.
OPEN MICURSOR
--3.
FETCH FIRST FROM MICURSOR
FETCH ABSOLUTE @N FROM MICURSOR
FETCH LAST FROM MICURSOR
--4.
CLOSE MICURSOR
DEALLOCATE MICURSOR
GO
```

En el punto uno se declara la variable @N que tiene por misión ingresar un número de mostrar desde el cursor y además de la declaración del cursor usando el operador SCROLL que desplazarse por los registros almacenados en el cursor.

```
DECLARE @N INT=5
DECLARE MICURSOR CURSOR SCROLL
    FOR SELECT * FROM PASAJERO
```

En el punto dos solo se apertura el cursor con OPEN MICURSOR, en el punto tres se invoca registro con el operador FIRST desde el cursor así mismo sobre el valor de la variable ABSOLUTE y el final de los registros con el operador LAST.

```
FETCH FIRST FROM MICURSOR
FETCH ABSOLUTE @N FROM MICURSOR
FETCH LAST FROM MICURSOR
```

8 views

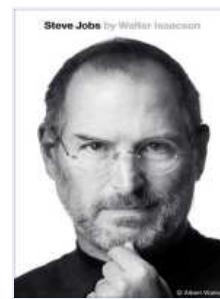
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

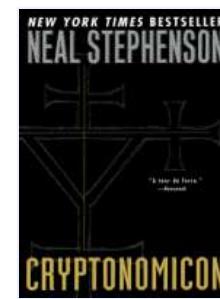
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

Finalmente la siguiente imagen muestra el resultado de la ejecución del cursor como nota mucho a tres consultas **SELECT** ejecutadas en simultáneo:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
1 PD005	Luz Lazaro Menor	0001	999999999	LLAZARO@GMAIL.COM
1 PD010	GUADALUPE ACOSTA FERRER	0003	938475747	GACOSTA@GMAIL.COM

4.14. FUNCIONES

La solución a problemas complejos se facilita si se dividen en porciones de código llamado permiten al programador en Transact SQL dividir en unidades de programas o módulo diseñadas para una tarea específica, la ventaja de esto es que solo se escribirá una vez p ser invocadas desde cualquier módulo de consulta ya que estas funciones se crean dentro de datos y se quedan registradas en él.

Hasta el momento ya hemos usado varias funciones estas tenían por misión devolver al ejecutar alguna acción como por ejemplo GETDATE() en las fechas o @@FETCH_STATUS implementación de cursores y así nos encontraremos con muchas funciones pre-establecid podemos llegar a la siguiente clasificación:

- Funciones del Sistema
- Función definidas por el usuario
 - Funciones Escalares
 - Funciones de Tabla
 - Funciones Tabla Online

4.15. FUNCIONES DEL SISTEMA

SQL Server cuenta con una infinita variedad de funciones dependiendo de los valores configuraciones que se desea realizar. Estas funciones normalmente se dividen en 4 grupos

- **Funciones de conjuntos de filas:** Devuelven un objeto que se puede utilizar como la tabla en una instrucción SQL.
- OPENDATASOURCE
- OPENROWSET

8 views

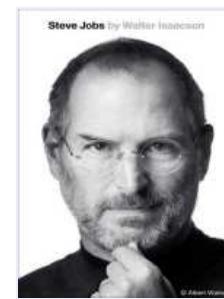
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

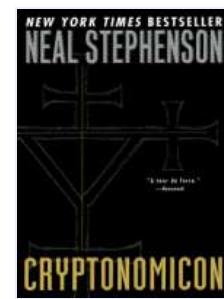
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

222 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

- SUM
 - COUNT
 - STDEV
 - COUNT_BIG
 - STDEVP
 - GROUPING
 - VAR
 - GROUPING_ID
 - VARP
 - MAX
- Funciones de categoría:** Devuelven un valor de categoría para cada fila de una partición
- RANK
 - NTILE
 - DENSE_RANK
 - ROW_NUMBER
- Funciones escalares:** Operan sobre un valor y después devuelven otro valor. Las funciones se pueden utilizar donde la expresión sea válida.
- Funciones de configuración:** Devuelven información acerca de la configuración actual
- @@DATEFIRST
 - @@OPTIONS
 - @@DBTS
 - @@REMSERVER
 - @@LANGID
 - @@SERVERNAME
 - @@LANGUAGE
 - @@SERVICENAME
 - @@LOCK_TIMEOUT
 - @@SPID
 - @@MAX_CONNECTIONS
 - @@TEXTSIZE
 - @@MAX_PRECISION
 - @@VERSION
 - @@NESTLEVEL

8 views

1 like

0 comments

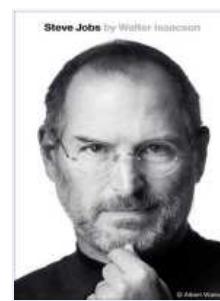
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

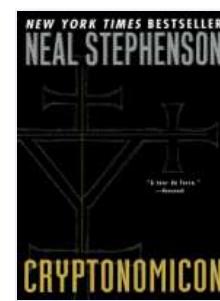
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

CAP. 4: PROGRAMACIÓN TRANSACT S

- **Funciones del cursor:** Devuelven información acerca de los cursosres.
 - @@CURSOR_ROWS
 - CURSOR_STATUS
 - @@FETCH_STATUS
- **Tipos de datos y funciones de fecha y hora:** Llevan a cabo operaciones sobre un valor de fecha y hora, y devuelven un valor numérico, de cadena o de fecha y hora.
 - SYSDATETIME
 - SYSDATETIMEOFFSET
 - SYSUTCDATETIME
 - CURRENT_TIMESTAMP
 - GETDATE
 - GETUTCDATE
 - DATENAME
 - DATEPART
 - DAY
 - MONTH
 - YEAR
 - DATEDIFF
 - DATEADD
 - EOMONTH
 - SWITCHOFFSET
 - TODATETIMEOFFSET
 - @@DATEFIRST
 - SET DATEFIRST
 - SET DATEFORMAT
 - @@LANGUAGE
 - SET LANGUAGE
 - SP_HELPLANGUAGE
 - ISDATE
- **Funciones lógicas:** Realizan operaciones lógicas.

8 views

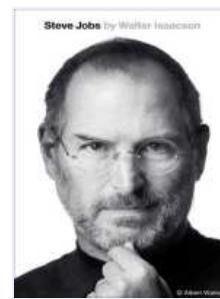
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

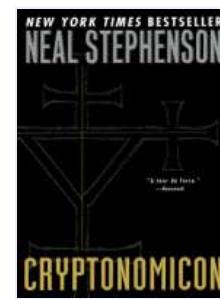
Uploaded by [anon_61286589](#)[Full description](#)

 Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

224

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

- RAND
- ACOS
- EXP
- ROUND
- ASIN
- FLOOR
- SIGN
- ATAN
- LOG
- SIN
- ATN2
- LOG10
- SQRT
- CEILING
- PI
- SQUARE
- COS
- POWER
- TAN
- COT
- RADIANS

- **Funciones de metadatos:** Devuelven información acerca de la base de datos y los objetos de base de datos.

- @@PROCID
- INDEX_COL
- APP_NAME
- INDEXKEY_PROPERTY
- APPLOCK_MODE
- INDEXPROPERTY
- APPLOCK_TEST
- NEXT VALUE FOR
- ASSEMBLYPROPERTY
- OBJECT_DEFINITION
- COL_LENGTH

8 views

1 like

0 comments

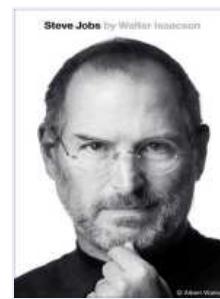
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

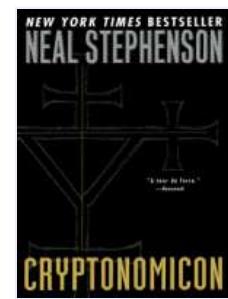
[Full description](#)

Save Embed Share Print

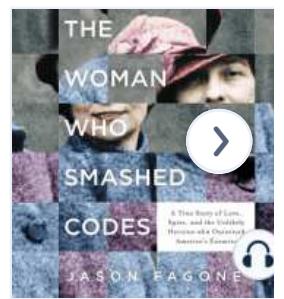
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

- DATABASEPROPERTYEX
- OBJECTPROPERTYEX
- DB_ID
- ORIGINAL_DB_NAME
- DB_NAME
- PARSENAME
- FILE_ID
- SCHEMA_ID
- FILE_INDEX
- SCHEMA_NAME
- FILE_NAME
- SCOPE_IDENTITY
- FILEGROUP_ID
- SERVERPROPERTY
- FILEGROUP_NAME
- STATS_DATE
- FILEGROUOPROPERTY
- TYPE_ID
- FILEPROPERTY
- TYPE_NAME
- FULLTEXTCATALOGPROPERTY
- TYPEPROPERTY
- FULLTEXTSERVICEPROPERTY

● Funciones de seguridad: Devuelven información acerca de usuarios y roles.

- CERTENCODED
- PWDCOMPARE
- CERTPRIVATEKEY
- PWDENCRYPT
- CURRENT_USER
- SCHEMA_ID
- DATABASE_PRINCIPAL_ID
- SCHEMA_NAME
- sys.fn_builtin_permissions
- SESSION_USER

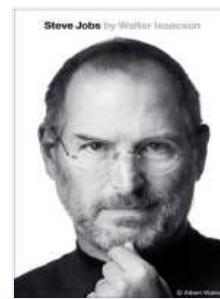
8 views

1 0

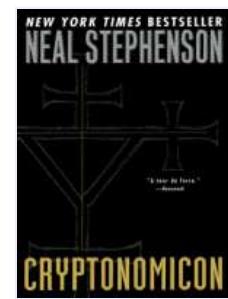
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of Codebreakers, Cryptographers, and Espionage

226 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

- IS_MEMBER
- SYSTEM_USER
- IS_ROLEMEMBER
- SUSER_NAME
- IS_SRVROLEMEMBER
- USER_ID
- ORIGINAL_LOGIN
- USER_NAME
- PERMISSIONS
- **Funciones de cadena:** Realizan operaciones en el valor de entrada de una cadena (char) y devuelven una cadena o un valor numérico.
 - ASCII
 - LTRIM
 - SOUNDEX
 - CHAR
 - NCHAR
 - SPACE
 - CHARINDEX
 - PATINDEX
 - STR
 - CONCAT
 - QUOTENAME
 - STUFF
 - DIFFERENCE
 - REPLACE
 - SUBSTRING
 - FORMAT
 - REPLICATE
 - UNICODE
 - LEFT
 - REVERSE
 - UPPER
 - LEN
 - RIGHT

8 views

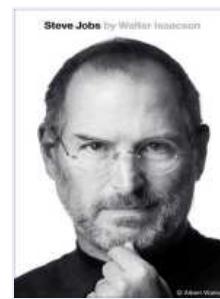
1 like

0 comments

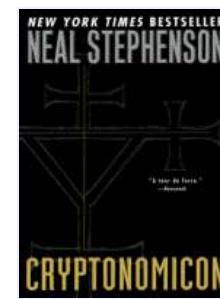
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

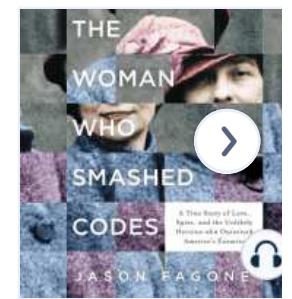
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

- **Funciones estadísticas** del sistema: Devuelven información estadística acerca del sistema.

 - @@CONNECTIONS
 - @@PACK_RECEIVED
 - @@CPU_BUSY
 - @@PACK_SENT
 - fn_virtualfilestats
 - @@TIMETICKS
 - @@IDLE
 - @@TOTAL_ERRORS
 - @@IO_BUSY
 - @@TOTAL_READ
 - @@PACKET_ERRORS
 - @@TOTAL_WRITE

- **Funciones de texto e imagen:** Realizan operaciones sobre los valores de entrada o como resultado de la ejecución de una consulta SELECT o PRINT; devuelven información acerca del valor.

 - PATINDEX
 - TEXTVALID
 - TEXTPTR

4.16. FUNCIONES DEFINIDAS POR EL USUARIO

Es un tipo de implementación registrada por Transact-SQL que tiene por finalidad devolver el resultado de la ejecución de una función definida por el usuario. Los lenguajes de programación que se utilizan para crear funciones definidas por el usuario son T-SQL y el lenguaje C. El resultado de la ejecución de una función definida por el usuario debe ser obtenido por algún medio mientras que el resultado de la ejecución de una función definida por el usuario se muestra a través de una consulta SELECT o la función PRINT; o asignarlo a una variable mediante la sentencia SET.

Las funciones definidas por el usuario no se pueden utilizar para realizar acciones que cambien el estado de la base de datos. Las funciones definidas por el usuario, como las funciones escalares, se pueden llamar desde una consulta. Las funciones escalares se pueden ejecutar con la sentencia EXECUTE, igual que los procedimientos almacenados.

Sintaxis de la función ESCALAR:

FUNCION DEFINIDA POR EL USUARIO ESCALAR

```
CREATE FUNCTION [ PROPIETARIO. ] NOMBRE_FUNCION (
    @PARAMETRO1 AS 1 TIPODATO1 = VALORXDEFECTO1 1 1
```

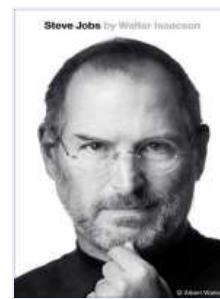
8 views

1 0

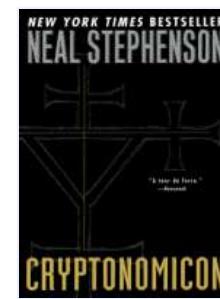
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

228 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Donde:

- **PROPIETARIO:** SQL Server administra sus objetos por medio de su creador es decir un para un grupo de objetos, aquellos objetos creados por cuenta del rol SYSADMIN automaticamente pertenecen a la cuenta DBO.

DBO es una cuenta de usuario muy particular ya que no puede ser utilizada para conexiones a la base de datos pero cuenta con privilegios especiales. La cuenta dbo también porque es el dueño de la base de datos. Por tanto los objetos pertenecientes a DBO tienen todos los permisos con dichos objetos y también se puede derivar a otras cuentas de usuario. Cuando sucede esto se tiene que especificar el nombre del propietario y luego como: PROPIETARIO.OBJETO.

En las funciones definidas por el usuario se tiene que especificar el propietario al ejecutar la función, se necesita comprobar que tipo de permiso tiene dicho usuario, para esto se puede especificar la siguiente manera: DBO.FN_CALCULA().

- **@PARAMETRO:** Se define las variables de entrada a la función es decir quien lo invoca tiene que enviar un valor por cada parámetro especificado en la función, hay que tener mucho cuidado con el tipo de datos de dichos valores.
- **RETURNS:** Permite especificar el tipo de datos de salida de la expresión en una función, teniendo en cuenta que una función de tipo escalar devuelve siempre un valor por lo tanto se debe especificar de qué tipo de datos es el valor de la salida.
- **RETURN:** Aquí se especifica el valor de salida de la función, se puede enviar directamente o por medio de una expresión.

CASO DESARROLLADO N° 4.29

Script que permite implementar una función que devuelva el promedio de dos números ingresados por el usuario.

```

IF OBJECT_ID(' MIPROMEDIO ') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.MIPROMEDIO
END
GO

CREATE FUNCTION MIPROMEDIO(@VALOR1 INT,@VALOR2 INT)
RETURNS DECIMAL(10,2)
AS
BEGIN
    DECLARE @RESULTADO DECIMAL(10,2)
    SET @RESULTADO=(@VALOR1+@VALOR2)/2.0
    RETURN @RESULTADO
END
GO

```

8 views

1 like

0 comments

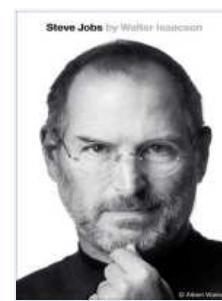
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

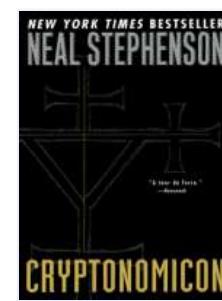
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

Luego se define el tipo de salida en esta función, la salida es DECIMAL ya que todo promete resultará una expresión decimal.

RETURNS DECIMAL(10,2)

Seguidamente se declara la variable @RESULTADO que tendrá el valor resultante de la expresión implementa la expresión del promedio, hay que tener en cuenta que se debe dividir entre los resultados que de eso dependerán los decimales.

```
DECLARE @RESULTADO DECIMAL(10,2)
SET @RESULTADO=(@VALOR1+@VALOR2)/2.0
```

Finalmente, se envía el resultado de la función por medio del operador RETURN. Así se obtiene el valor resultante de la función desde la invocación con Transact SQL o un lenguaje de programación.

Ahora tomaremos especial interés en el uso de la función de dos formas:

Primera forma: SELECT

```
SELECT dbo.MIPROMEDIO(12,13) AS [PROMEDIO]
GO
```

En el script se usa la sentencia **SELECT** como medio de invocación a la función **MIPROMEDIO**. Para llamar a la función se tiene que especificar el nombre del propietario en este caso es **dbo**. Tenga en cuenta que los parámetros 12 y 13 son parametrizados por la función con el nombre **VALOR1** y **VALOR2** respectivamente. La imagen siguiente muestra el resultado de la prueba con **SELECT**.

PROMEDIO	
1	12.50

Segunda forma: PRINT

```
DECLARE @N1 INT=12,@N2 INT=13
PRINT 'EL PROMEDIO DE '+
      'CAST(@N1 AS CHAR(2))+'+CAST(@N2 AS CHAR(2))+'' ES: '+
      CAST(DBO.MIPROMEDIO(@N1,@N2) AS CHAR(10))
GO
```

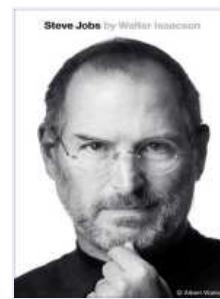
8 views

1 0

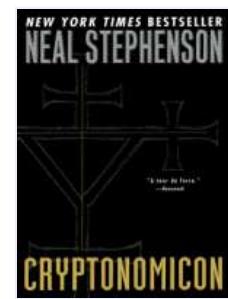
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

230 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 4.30

Script que permite implementar una función que devuelva el nombre de un mes en letra fecha ingresada por el usuario.

```

IF OBJECT_ID(' MESLETROS ') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.MESLETROS
END
GO

CREATE FUNCTION MESLETROS(@FECHA DATE)
RETURNS VARCHAR(20)
AS
BEGIN
    DECLARE @NOMBRE VARCHAR(20)
    SET @NOMBRE=
        CASE DATENAME(MONTH,@FECHA)
        WHEN 'JANUARY' THEN 'ENERO'
        WHEN 'FEBRUARY' THEN 'FEBRERO'
        WHEN 'MARCH' THEN 'MARZO'
        WHEN 'APRIL' THEN 'ABRIL'
        WHEN 'MAY' THEN 'MAYO'
        WHEN 'JUNE' THEN 'JUNIO'
        WHEN 'JULY' THEN 'JULIO'
        WHEN 'AUGUST' THEN 'AGOSTO'
        WHEN 'SEPTEMBER' THEN 'SETIEMBRE'
        WHEN 'OCTOBER' THEN 'OCTUBRE'
        WHEN 'NOVEMBER' THEN 'NOVIEMBRE'
        WHEN 'DECEMBER' THEN 'DICIEMBRE'
        END
    RETURN @NOMBRE
END
GO

```

En el script la función **MESLETROS** tiene como parámetro una fecha que será enviada por la función obtendrá el mes en número de esa fecha para remitirla por el nombre de dicho

En esta función se hizo necesario el uso de la estructura múltiple **CASE** ya que son 12 compa un mismo valor que es el número de mes obtenido de la fecha ingresada como parámetrc

La función **DATENAME** permite devolver el nombre del mes desde una fecha determin devuelve en inglés y no nosotros necesitamos componer una fecha en castellano.

Mostraremos TRES formas de ejecutar la función:

Primera forma: SELECT

8 views

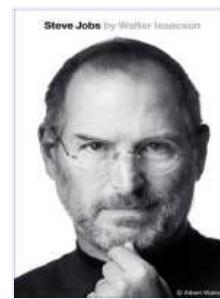
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

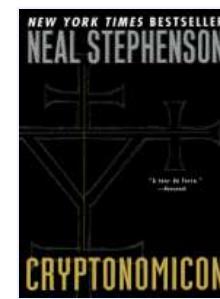
Uploaded by [anon_61286589](#)[Full description](#)

 Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Hidden Heroine Behind a Conspiracy

Segunda forma: PRINT

```
DECLARE @FEC DATE=GETDATE()
PRINT      'LA FECHA '+CAST(@FEC AS VARCHAR(15))+'
              ' TIENE COMO MES A: '+dbo.MESLETRAS(GETDATE())
GO
```

LA FECHA 2012-08-26 TIENE COMO MES A: AGOSTO

Tercera forma: Usando la columna de una tabla

```
SELECT PAG.IDRESERVA AS [Nº RESERVA],
       PAS.NOMBRES AS [PASAJERO],
       CAST(DAY(FECHA) AS CHAR(2))+'' DE ''
          +dbo.MESLETRAS(FECHA)
          +' DEL '+' 
          CAST(YEAR(FECHA) AS CHAR(4)) AS [FECHA],
       PAG.MONTO AS [MONTO DE PAGO]
     FROM PAGO PAG
   JOIN PASAJERO PAS ON PAG.IDPASAJERO=PAS.IDPASAJERO
GO
```

En el script se implementa la columna **FECHA** la cual se compone del día capturada con la **DAY** y convertida a caracteres con **CAST**, el mes en letras obtenida desde la función **MESLETRAS** obtenido desde la función **YEAR**.

La imagen siguiente muestra que la columna fecha expresa una fecha en formato de texto.

Nº RESERVA	PASAJERO	FECHA	MONTO DE PAGO
1	LUZ LAZARO MENOR	1 DE OCTUBRE DEL 2011	500.00
2	MARIA ZAMORA MEJIA	6 DE OCTUBRE DEL 2011	900.00
3	HEIDI RENGIRO REATEGUI	14 DE NOVIEMBRE DEL 2011	500.00
4	FERNANDA TORRES LAZARO	16 DE NOVIEMBRE DEL 2011	1200.00
5	ANGELA TORRES LAZARO	12 DE DICIEMBRE DEL 2011	1500.00
6	KARLA GALLEGOS SILVA	17 DE DICIEMBRE DEL 2011	590.00
7	MARIA ZAMORA MEJIA	14 DE ENERO DEL 2012	400.00
8	MARIA ZAMORA MEJIA	15 DE ENERO DEL 2012	1300.00
9	HEIDI RENGIRO REATEGUI	1 DE FEBRERO DEL 2012	1000.00
10	FERNANDA TORRES LAZARO	2 DE ABRIL DEL 2012	1800.00
11	ANGELA TORRES LAZARO	9 DE ABRIL DEL 2012	1200.00
12	KARLA GALLEGOS SILVA	1 DE AGOSTO DEL 2012	400.00
13	MARIA ZAMORA MEJIA	20 DE AGOSTO DEL 2012	900.00

8 views

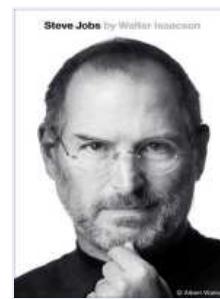
1 like

0 comments

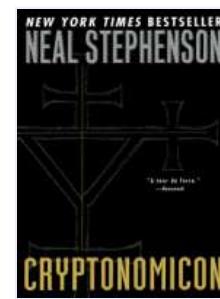
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

232

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

```

IF OBJECT_ID(' PROMEDIOCOSTO ') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.PROMEDIOCOSTO
END
GO

CREATE FUNCTION PROMEDIOCOSTO(@AÑO INT)
RETURNS DECIMAL(10,2)
BEGIN
    DECLARE @PROM DECIMAL(10,2)
    SELECT @PROM=AVG(COSTO)
    FROM RESERVA
    WHERE YEAR(FECHA)=@AÑO
    RETURN @PROM
END
GO

```

En el script se implementa la función PROMEDIOCOSTO que tiene como parámetro un año que de la condicional de la consulta. Como se define el tipo de salida DECIMAL(10,2) la variable @PROM que será definida de la misma capacidad ya que al final ese valor será el retornado por la función.

La imagen siguiente muestra el resultado de la función:

EL PROMEDIO DE COSTOS DEL AÑO 2012 ES: 811.11

CASO DESARROLLADO N° 4.32

Script que permita implementar una función que devuelva el total de pasajeros de un país ingresado por el usuario.

PASAJERO	
IDPAJERO	
NOMBRES	
IDPAIS	
TELEFONO	
EMAIL	

Column Name	Condensed Type
IDPAIS	char(4)
PAIS	varchar(30)

```

IF OBJECT_ID(' PASAJEROSXPAIS ') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.PASAJEROSXPAIS
END
GO

CREATE FUNCTION PASAJEROSXPAIS(@PAIS VARCHAR(40))

```

8 views

1 like

0 comments

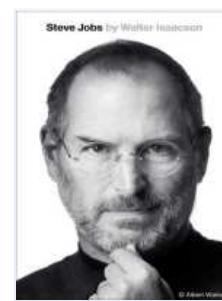
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

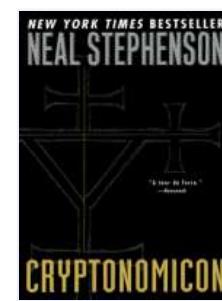
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

En el script se implementa la función **PASAJEROSXPAIS** con el parámetro **@PAIS** que permitirá el nombre del país que será parte de la condicional en el conteo de pasajeros por país.

Se declara la variable **@TOTAL** que almacenara el total devuelto por la función **COUNT** cuando sea **TRUE** la condición **WHERE PAI.PAIS=@PAIS**, se tuvo que implementar un **JOIN** ya que el nombre del país se encuentra en la tabla **PAÍS**.

Para la ejecución de la función se puede usar el siguiente script:

```
DECLARE @PAIS VARCHAR(40)='PERU'
PRINT 'EL TOTAL DE PASAJEROS DE ' + @PAIS + ' ES :'+ 
      CAST(DBO.PASAJEROSXPAIS(@PAIS) AS CHAR(4))
GO
```

La imagen siguiente muestra el resultado de la ejecución del script anterior:

EL TOTAL DE PASAJEROS DE PERU ES : 3

CASO DESARROLLADO N° 4.33

Script que permita implementar una función que devuelva el acumulado de los montos determinado mes y año ingresado por el usuario.



```
IF OBJECT_ID('ACUMULADOMESYANO') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.ACUMULADOMESYANO
END
GO

CREATE FUNCTION ACUMULADOMESYANO(@MES INT,@AÑO INT)
RETURNS DECIMAL(10,2)
BEGIN
    DECLARE @TOTAL DECIMAL(10,2)
```

8 views

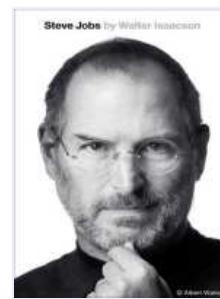
1 like

0 comments

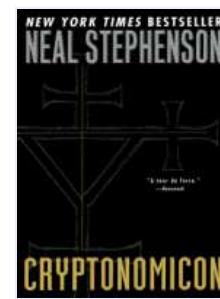
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

234

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el script se implementa la función **ACUMULADOXMESYAN**O en donde se ingresan como el mes y el año, para obtener de ellos aquellos montos realizados en dicho mes y año.

Esta vez se implementa la función **SUM** que permitirá acumular todos los registros que cumplen con la condición de igual mes e igual año. Para ejecutar la función se implementa el siguiente scri

```
DECLARE @MES INT=11
DECLARE @AÑO INT=2011
PRINT 'EL TOTAL DE PAGOS REALIZADOS EN EL MES '+
      LTRIM(STR(@MES))+ ' DEL AÑO '+
      LTRIM(STR(@AÑO))+ ' ES: '+
      LTRIM(STR(DBO.ACUMULADOXMESYAN(@MES,@AÑO)))
GO
```

En el script se declaran dos variables una para el mes y otra para el año inicializados en el mes de noviembre y el año 2011 en donde se realizaron los siguientes pagos, mostramos un pequeño script que calcula el total acumulado de los montos para luego comprobar que es el resultado adecuado:

```
SELECT SUM(MONTO)
      FROM PAGO
     WHERE MONTH(FECHA)=11 AND YEAR(FECHA)=2011
       GROUP BY YEAR(FECHA)
GO
```

En este caso con los registros que se cuenta el resultado sería:

ACUMULADO	
1	1700.00

En resultado de ejecutar el script anterior es:

EL TOTAL DE PAGOS REALIZADOS EN EL MES 11 DEL AÑO 2011 ES: 1700

La función **LTRIM** permite eliminar los espacios en blanco del lado izquierdo de una variable y la función **STR** permite convertir a cadena una variable de un tipo distinto a la que se le aplica. Su trabajo es análogo a la implementada con **CAST** o **CONVERT**.

Como notará es el mismo valor emitido por el script de prueba.

8 views

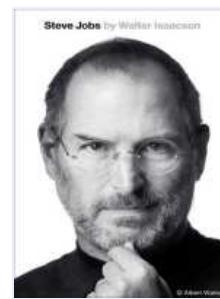
1 like

0 comments

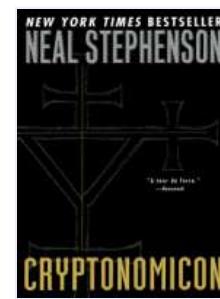
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

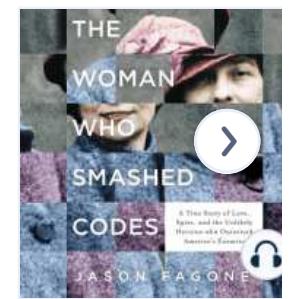
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CASO DESARROLLADO N° 4.34

Script que permita implementar una función que devuelva el factorial de un número ingresado por el usuario. Deberá usar recursividad de la función.

```

IF OBJECT_ID('FACTORIAL') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.FACTORIAL
END
GO

CREATE FUNCTION DBO.FACTORIAL(@N INT)
RETURNS INT
AS
BEGIN
    DECLARE @FACT INT
    IF @N <= 1
        SET @FACT = 1
    ELSE
        SET @FACT = @N * DBO.FACTORIAL(@N-1)
    RETURN @FACT
END

```

En el script se implementa la función FACTORIAL que tiene como parámetro @N de tipo INT. También se define como tipo de dato devuelto a INT ya que el valor factorial no es decimal. La recursividad se logra mediante la llamada a la misma función FACTORIAL(@N-1) dentro del script.

Dentro de la función se declara la variable @FACT que permitirá devolver el valor resultante. El script condiciona al número ingresado; si es menor o igual a uno entonces @FACT=1 caso contrario @FACT multiplicará la variable @N por la llamada a la misma función FACTORIAL(@N-1). Se llamará al número descendente al valor ingresado hasta llegar a uno donde se finalizará la ejecución. Esta actividad se le llama recursividad.

Para ejecutar la función se deberá colocar el siguiente script:

```

DECLARE @N INT=4
SELECT @N AS [NUMERO],dbo.FACTORIAL(@N) AS [FACTORIAL]
GO

```

La imagen siguiente muestra el resultado de la ejecución:

NUMERO	FACTORIAL
1	4

8 views

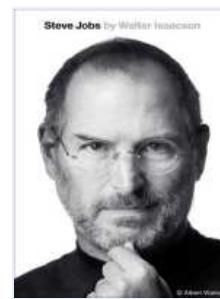
1 like

0 comments

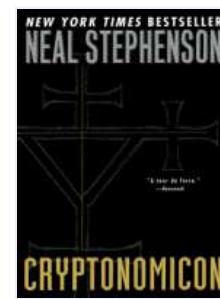
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

236 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

La imagen siguiente muestra el resultado de la ejecución:

EL FACTORIAL DEL NUMERO 4 ES 24

Sintaxis de la función TABLA EN LINEA:

**FUNCION DEFINIDA POR EL USUARIO
TABLA EN LINEA**

```
CREATE FUNCTION [ PROPIETARIO. ] NOMBRE_FUNCION
( [ @PARAMETRO [ AS ] TIPO_DATOS [ = DEFAULT ] ]
)
RETURNS TABLE
[ AS ]
RETURN [ ( CONSULTA ) ]
```

CASO DESARROLLADO N° 4.35

Script que implemente una función de Tabla en Línea que muestre los registros de la tabla



```
IF OBJECT_ID('MISPASAJEROS') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.MISPASAJEROS
END
GO

CREATE FUNCTION MISPASAJEROS()
RETURNS TABLE
AS
    RETURN (SELECT * FROM PASAJERO)
GO
```

En el script se implementa la función **MISPASAJEROS** que tiene por misión devolver los registros de la tabla **PASAJERO**. La diferencia con la función escalar es que cuando se retorna una tabla con **RETURNS TABLE**: devuelve más de un valor. eso quiere decir que la forma de ejecución es

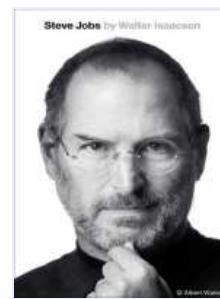
8 views

1 0

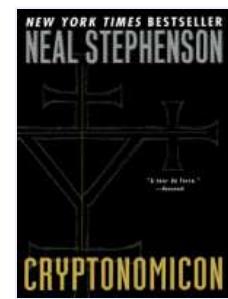
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

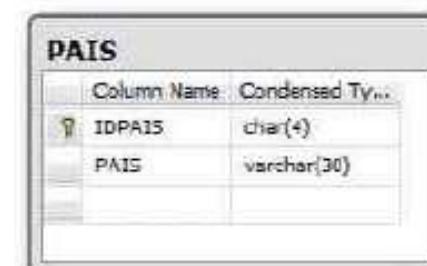
La imagen siguiente muestra el resultado de la ejecución de la función tipo tabla:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4 P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5 P0005	LULU LAZARO MENOR	0001	999999999	LLAZARO@GMAIL.COM
6 P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7 P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8 P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
9 P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
10 P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM

Como notara el resultado es similar a ejecutar una consulta simple con la sentencia SELECT

CASO DESARROLLADO N° 4.36

Script que implemente una función de Tabla en Línea que muestre los registros de la tabla dependiendo del país de proveniencia.



```

IF OBJECT_ID('PASAJEROSXPAIS') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.PASAJEROSXPAIS
END
GO

CREATE FUNCTION PASAJEROSXPAIS(@PAIS VARCHAR(30))
RETURNS TABLE
AS
RETURN (
    SELECT PAS.IDPASAJERO, PAS.NOMBRES, PAI.PAIS,
           PAS.TELEFONO, PAS.EMAIL
    FROM PASAJERO PAS
    JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
    WHERE PAI.PAIS=@PAIS
)
GO

```

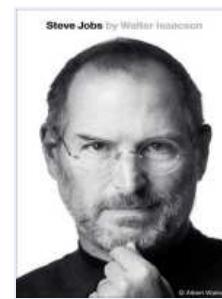
8 views

1 0

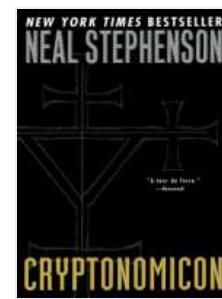
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

238 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

El script siguiente muestra la forma de ejecutar una función de tabla en línea con parámetros:

```
SELECT * FROM DBO.PASAJEROSXPAIS('PERU')
GO
```

El resultado de la ejecución de la función es el siguiente:

IDPASAJERO	NOMBRES	PAÍS	TELEFONO	EMAIL
P0001	ANGELA TORRES LAZARO	PERU	9999999999	ATORRES@HOTMAIL.COM
P0002	FERNANDA TORRES LAZARO	PERU	9999999999	FTORRES@HOTMAIL.COM
P0005	LUZ LAZARO MENOR	PERU	9999999999	LLAZARO@GMAIL.COM

CASO DESARROLLADO N° 4.37

Script que implemente una función de Tabla en Línea que muestre los montos acumulados pagos realizados por un determinado pasajero. Para esto implemente la función que permite obtener el país, otra función que permita acumular los pagos y la función tabla en línea que permite los pasajeros y su monto acumulado.

IDPASAJERO	NOMBRES	PAÍS	ACUMULADO PAGO
P0001	ANGELA TORRES LAZARO	PERU	2700.00
P0002	FERNANDA TORRES LAZARO	PERU	3000.00
P0003	MARIA ZAMORA MEJA	BRASIL	3400.00
P0004	GUADALUPE ACOSTA FERRER	ARGENTINA	NULL
P0005	LUZ LAZARO MENOR	PERU	500.00
P0006	KARLA GALLEGOS SILVA	PARAGUAY	990.00
P0007	NERY CALLE DE LA CRUZ	MEXICO	NULL
P0008	HEIDI RENGIFO REATEGUI	ECUADOR	1500.00
P0009	MARISOL DIAZ ZAMBRANO	ECUADOR	NULL
P0010	LINDA TUME VARAS	URUGUAY	NULL

```
--1.
IF OBJECT_ID('MUESTRAPAI') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.PASAJEROSXPAIS
END
GO

IF OBJECT_ID('ACUMULADOPAGO') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.PASAJEROSXPAIS
END
GO

IF OBJECT_ID('PAGOXPASAJEROS') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.PASAJEROSXPAIS
END
GO
```

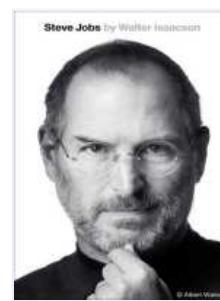
8 views

1 0

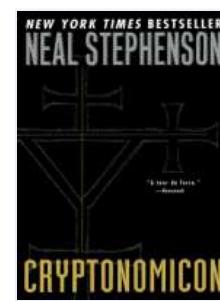
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

```

--3.
CREATE FUNCTION ACUMULADOPAGO(@ID CHAR(5))
RETURNS DECIMAL(10,2)
AS
BEGIN
    RETURN (SELECT SUM(MONTO)
            FROM PAGO PA
            WHERE PA.IDPASAJERO=@ID)
END
GO

--4.
CREATE FUNCTION PAGOXPASAJEROS()
RETURNS TABLE
AS
RETURN(
    SELECT PAS.IDPASAJERO, PAS.NOMBRES,
           DBO.MUESTRAPAIIS(PAS.IDPAIS) AS [PAIS],
           DBO.ACUMULADOPAGO(PAS.IDPASAJERO) AS [ACUMULADO PA]
    FROM PASAJERO PAS
    JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS

)

```

En el punto uno se implementan tres funciones que permitirán verificar que las funciones a se encuentren en la base de datos, en el caso se encontrase será eliminado de la base.

En el punto dos se implementa la función escalar **MUESTRAPAIIS** que tiene por misión nombre del país según su código obtenido desde la tabla **PASAJERO**.

```

CREATE FUNCTION MUESTRAPAIIS(@ID CHAR(4))
RETURNS VARCHAR(30)
AS
BEGIN
    RETURN (SELECT PA.PAIS
            FROM PAIS PA
            WHERE PA.IDPAIS=@ID)
END
GO

```

Como notará en el script no se usó una variable local para guardar el nombre del país valor resultante de la función se está enviando directamente por la cláusula **RETURN**; la otra realizar la misma acción con más líneas de código sería:

```
BEGIN
```

8 views

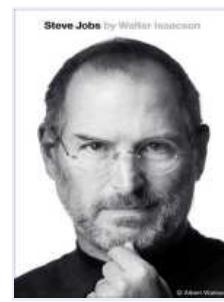
1 like

0 comments

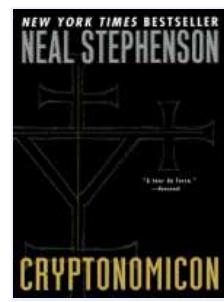
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

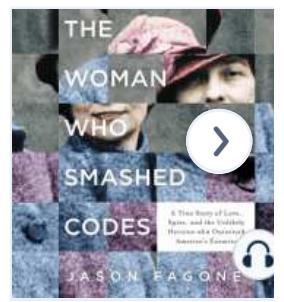
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

240 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el punto tres se implementa la función escalar **ACUMULADOPAGO** que tiene como parámetro el **ID** del pasajero que permitirá condicionar la suma de los montos de acuerdo a dicho valor.

```
CREATE FUNCTION ACUMULADOPAGO(@ID CHAR(5))
RETURNS DECIMAL(10,2)
AS
BEGIN
    RETURN (SELECT SUM(MONTO)
            FROM PAGO PA
            WHERE PA.IDPASAJERO=@ID)
END
GO
```

En el punto cuatro se implementa la función tabla en línea **PAGOXPASAJEROS** sin parámetros que devolverá los registros desde la tabla pasajero mostrando el nombre del pasajero y el acumulado de pago realizado desde la función **MUESTRAPAI**S y el acumulado de pago realizado desde la función **ACUMULADOPAGO**.

```
CREATE FUNCTION PAGOXPASAJEROS()
RETURNS TABLE
AS
RETURN
(
    SELECT PAS.IDPASAJERO, PAS.NOMBRES,
           DBO.MUESTRAPAI(PAS.IDPAIS) AS [PAIS],
           DBO.ACUMULADOPAGO(PAS.IDPASAJERO) AS [ACUMULADO PAGO]
    FROM PASAJERO PAS
)
GO
```

La idea principal del script es no usar **JOIN** para la obtención de valores que se encuentran en el mismo alcance de la tabla **PASAJERO**.

Para ejecutar la función **PAGOXPASAJEROS** deberá ejecutar el siguiente script:

```
SELECT * FROM DBO.PAGOXPASAJEROS()
GO
```

El resultado de la ejecución se muestra a continuación:

	IDPASAJERO	NOMBRES	PAÍS	ACUMULADO PAGO
1	P0001	ANGELA TORRES LAZARO	PERU	2700.00
2	P0002	FERNANDA TORRES LAZARO	PERU	3000.00

8 views

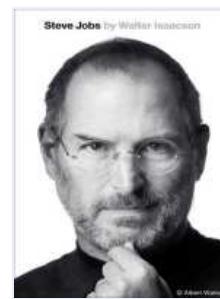
1 like

0 comments

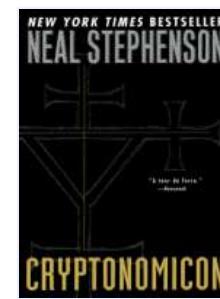
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

Sintaxis de la función TABLA MULTISENTENCIA:

FUNCION DEFINIDA POR EL USUARIO TABLA MULTISENTENCIA

```
CREATE FUNCTION [ PROPIETARIO. ] NOMBRE_FUNCION
( [ @PARAMETRO [ AS ] TIPO_DATO [ = DEFAULT ] [ ,...N ] ] )
RETURNS @RETURN_VARIABLE TABLE <TABLE_TYPE_DEFINITION>
[ WITH <FUNCTION_OPTION> [ ,...N ] ]
[ AS ]
BEGIN
    CUERPO_DE_LA_FUNCION
    RETURN
END
```

CASO DESARROLLADO Nº 4.38

Script que implemente una función de Tabla multisentencia que muestre los registros AEROPUERTO adicionando el nombre del país.

AEROPUERTO	
IDAERO	
NOMBRE	
IDPAIS	

PAÍS	
Column Name	Condensed Type
IDPAIS	char(4)
PAIS	varchar(30)

```
--1.
IF OBJECT_ID('MISAEROPUERTOS') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.MISAEROPUERTOS
END
GO

--2.
CREATE FUNCTION MISAEROPUERTOS()
RETURNS @TABLA TABLE(
    IDA CHAR(4),
    NOM VARCHAR(40),
    PAI VARCHAR(40)
)
AS
BEGIN
--3.
    INSERT INTO @TABLA
        SELECT AER.IDAERO,AER.NOMBRE,PAI.PAIS
        FROM AEROPUERTO AER
        JOIN PAIS PAI ON AER.IDPAIS=PAI.IDPAIS
    RETURN
END
GO
```

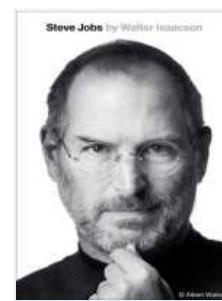
8 views

1 0

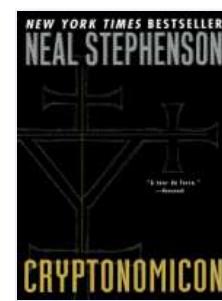
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

242

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el punto tres se realiza la inserción de registros hacia la variable de tipo table @TABLA desde la tabla AEROPUERTO unida con PAÍS. No se olvide que por ser una función de tabla muⁿo tiene valor de salida entonces el operador RETURN se imprime solo.

Para la ejecución de la función se debe colocar el siguiente script:

```
SELECT *
FROM MISAEROPUERTOS()
GO
```

La imagen siguiente muestra el resultado de la ejecución de la consulta que involucra a MISAEROPUERTOS.

ID	NOM	PAI
1 AE01	BARILOCHE	ARGENTINA
2 AE02	MAR DEL PLATA	ARGENTINA
3 AE03	JORGE CHAVEZ	PERU
4 AE04	SANTIAGO	CHILE
5 AE05	AICM	MEXICO
6 AE06	JOSE JOAQUIN DE OLMEDO	ECUADOR
7 AE07	SIMON BOLIVAR	VENEZUELA
8 AE08	SAO PAULO CONGONHAS	BRASIL
9 AE09	SILVIO PETTIROSSI	PARAGUAY
10 AE10	CARRASCO PUERTA DEL SUR	URUGUAY

Como notara en la imagen el encabezado de la consulta resultante no es lo esperado, e a que las variable declaradas como columnas de la variable de tipo table así lo defini podemos mejorar el aspecto si colocamos el siguiente script:

```
SELECT IDA AS CODIGO, NOM AS AEROPUERTO, PAI AS PAIS
FROM MISAEROPUERTOS()
GO
```

La imagen siguiente muestra el resultado final de la ejecución de la función:

	CODIGO	AEROPUERTO	PAIS
1	AE01	BARILOCHE	ARGENTINA
2	AE02	MAR DEL PLATA	ARGENTINA
3	AE03	JORGE CHAVEZ	PERU
4	AE04	SANTIAGO	CHILE
5	AE05	AICM	MEXICO
6	AE06	JOSE JOAQUIN DE OLMEDO	ECUADOR
7	AE07	SIMON BOLIVAR	VENEZUELA

8 views

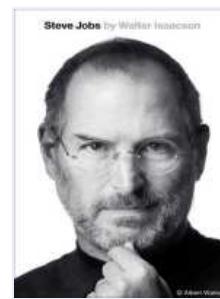
1 like

0 comments

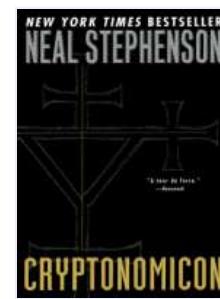
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in History

CASO DESARROLLADO N° 4.39

Script que implemente una función de Tabla multisentencia que muestre los registros de tabla en una misma función, para esto se deberá tomar dos columnas de cada tabla y asignarle una variable de tipo table para poder mostrarlo.

AEROPUERTO	
IDAERO	NOMBRE
IDPAIS	TDAIS

PAÍS	
Column Name	Condensed Type
IDPAIS	char(4)
PAÍS	varchar(30)

PASAJERO	
IDPASAJERO	NOMBRES
IDPAIS	TELEFONO
EMAIL	

```

IF OBJECT_ID('FN_VISTATABLA') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.FN_VISTATABLA
END
GO

CREATE FUNCTION FN_VISTATABLA(@TABLA VARCHAR(20))
RETURNS @MATRIZ TABLE (CÓDIGO CHAR(10),DESCRIPCION VARCHAR(50))
AS
BEGIN
    IF @TABLA='PAÍS'
        INSERT INTO @MATRIZ
            SELECT IDPAIS,PAÍS
            FROM PAÍS
    ELSE IF @TABLA='PASAJERO'
        INSERT INTO @MATRIZ
            SELECT IDPASAJERO,NOMBRES
            FROM PASAJERO
    ELSE IF @TABLA='AEROPUERTO'
        INSERT INTO @MATRIZ
            SELECT IDAERO,NOMBRE
            FROM AEROPUERTO
    RETURN
END
GO

```

En el script se implementa la función **FN_VISTATABLA** con respecto al nombre de la función es necesario especificarlo con la palabra **FN** por delante. Se le adiciona un parámetro a la función multisentencia para que identifique a que tabla necesita mostrar.

Se declara la variable de retorno tipo table llamado **@MATRIZ** este tiene por misión devolver de cualquiera de las tablas involucradas en la consulta. Por ese motivo también es que se declaran las columnas CÓDIGO y DESCRIPCION como dos campos genéricos a todas las tablas.

Luego se condiciona el valor de la variable **@TABLA**, buscando así el identificador de cada tabla cuando la condición devuelva TRUE entonces insertara dentro de la variable tipo table **@MATRIZ** las columnas mediante la sentencia **SELECT**. En este caso al encontrar que la variable es PAÍS insertara los valores de la columna PAÍS en dicha variable y así sucederá con las demás condiciones teniendo en cuenta que para este caso las columnas son del mismo tipo en caso no sea así se utilizará la sentencia **CONVERT** para sincronizar las columnas y no tener errores de tipo desde el motor de base de datos.

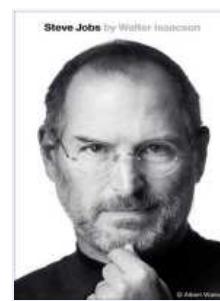
8 views

1 0

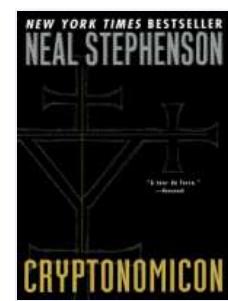
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

244

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En cualquiera de los casos el resultado sería como lo muestra la siguiente imagen:

CODIGO	DESCRIPCION
1	PERU
2	ARGENTINA
3	CHILE
4	ECUADOR
5	BRASIL
6	VENEZUELA
7	PARAGUAY
8	URUGUAY
9	BOLIVIA
10	MEXICO
11	HONDURAS
12	EEUU
13	PUERTO RICO

Para cualquiera de las tablas la ejecución siempre tendrá la misma cabecera es decir **IDPAIS**, **IDPASAJERO** o **IDAERO** siempre con el nombre **CÓDIGO**; lo mismo le sucederá a **DESCRIPCION**.

O en todo caso se podría ejecutar todas las consultas al mismo tiempo de la siguiente forma:

```
SELECT * FROM FN_VISTATABLA('PAIS')
SELECT * FROM FN_VISTATABLA('PASAJERO')
SELECT * FROM FN_VISTATABLA('AEROPUERTO')
GO
```

CODIGO	DESCRIPCION
1	PERU
2	ARGENTINA
3	CHILE
4	ECUADOR
5	BRASIL
6	VENEZUELA
7	PARAGUAY
8	URUGUAY

CODIGO	DESCRIPCION
P0001	ANGELA TORRES LAZARO
P0002	FERNANDA TORRES LAZARO
P0003	MARIA ZAMORA NEJIA
P0004	GUADALUPE ACOSTA FERRER
P0005	LULU LAZARO MENOR
P0006	KARLA GALLEGOS SILVA
P0007	NERY CALLE DE LA CRUZ
P0008	HEIDI RENGIFO REATEGUI

CODIGO	DESCRIPCION
AFO1	RARI LOCHF

8 views

1 like

0 comments

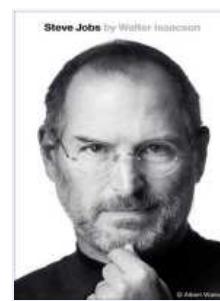
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

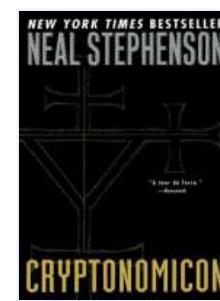
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

4.17. PROCEDIMIENTOS ALMACENADOS

Un procedimiento almacenado es un conjunto de instrucciones de Transact-SQL o una función que se ejecuta en el servidor y se devuelven parámetros proporcionados por el usuario.

Los procedimientos se pueden crear para uso permanente o para uso temporal en una sesión, un procedimiento local temporal, o para su uso temporal en todas las sesiones, un procedimiento temporal global.

El motor de base de datos considera al procedimiento almacenado como un objeto que contiene un conjunto de instrucciones SQL que se ejecutan en conjunto con un mismo nombre y una sola ejecución.

Consideraciones generales sobre los procedimientos almacenados

- Todo procedimiento almacenado se registra en el servidor actual.
- Pueden incluir atributos de seguridad como permisos y cadenas de propiedad; además, pueden asociar certificados.
- Los procedimientos almacenados mejoran la seguridad de una aplicación.
- Los procedimientos almacenados con parámetros pueden ayudar a proteger la aplicación de ataques por inyección de código SQL.
- Permiten una programación modular.
- Puede crear el procedimiento una vez y llamarlo desde el programa tantas veces como sea necesario, lo que puede mejorar el mantenimiento de la aplicación y permitir que las aplicaciones tengan acceso a la base de datos de manera uniforme.
- Pueden reducir el tráfico de red.
- Una operación que necesite centenares de líneas de código Transact-SQL puede realizarla en una sola instrucción que ejecute el código en un procedimiento, en vez de enviar cientos de líneas de código por la red.

Tipos de procedimiento almacenados

Se describen particularmente tres tipos de procedimientos almacenados:

- Procedimientos Almacenados del sistema
- Procedimientos Almacenados definidos por el usuario
- Procedimientos Almacenados Extendidos

4.18. PROCEDIMIENTOS ALMACENADOS DEL SISTEMA

8 views

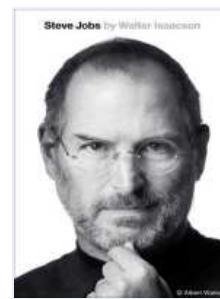
1 like

0 comments

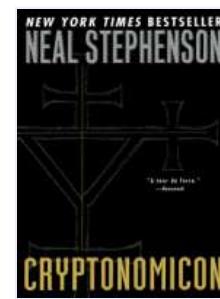
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

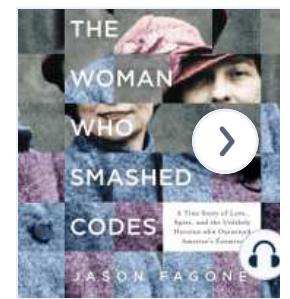
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

246 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Nombraremos algunos procedimientos almacenados del sistema y propondremos algunos relevantes de estos procedimientos:

- sp_catalogs
- sp_column_privileges
- sp_column_privileges_ex
- sp_columns
- sp_columns_ex
- sp_databases
- sp_ExecuteSQL
- sp_datatype_info
- sp_fkeys
- sp_foreignkeys
- sp_indexes
- sp_pkeys
- sp_primarykeys
- sp_server_info
- sp_special_columns
- sp_sproc_columns
- sp_statistics
- sp_table_privileges
- sp_table_privileges_ex
- sp_tables
- sp_tables_ex

CASO DESARROLLADO N° 4.40

Procedimiento almacenado del sistema que muestre los privilegios de las columnas involucradas en la tabla PASAJERO.

PASAJERO	
IDPASAJERO	
NOMBRES	
IDPAIS	
TELEFONO	
EMAIL	

8 views

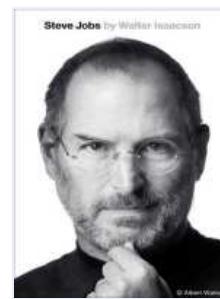
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

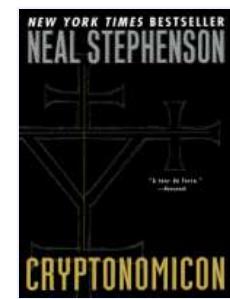
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

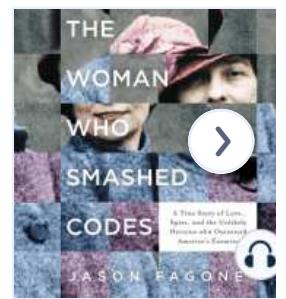
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

La imagen siguiente muestra el resultado de la ejecución del procedimiento almacenado:

TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	GRANTOR	GRANTEE	PRIVILEGE	IS_DEF
1	AGENCIA	PASAJERO	EMAIL	dbo	dbo	INSERT	YES
2	AGENCIA	PASAJERO	EMAIL	dbo	dbo	REFERENCES	YES
3	AGENCIA	PASAJERO	EMAIL	dbo	dbo	SELECT	YES
4	AGENCIA	PASAJERO	EMAIL	dbo	dbo	UPDATE	YES
5	AGENCIA	PASAJERO	IDPAIS	dbo	dbo	INSERT	YES
6	AGENCIA	PASAJERO	IDPAIS	dbo	dbo	REFERENCES	YES
7	AGENCIA	PASAJERO	IDPAIS	dbo	dbo	SELECT	YES
8	AGENCIA	PASAJERO	IDPAIS	dbo	dbo	UPDATE	YES
9	AGENCIA	PASAJERO	IDPASAJERO	dbo	dbo	INSERT	YES
10	AGENCIA	PASAJERO	IDPASAJERO	dbo	dbo	REFERENCES	YES
11	AGENCIA	PASAJERO	IDPASAJERO	dbo	dbo	SELECT	YES
12	AGENCIA	PASAJERO	IDPASAJERO	dbo	dbo	UPDATE	YES
13	AGENCIA	PASAJERO	NOMBRES	dbo	dbo	INSERT	YES
14	AGENCIA	PASAJERO	NOMBRES	dbo	dbo	REFERENCES	YES
15	AGENCIA	PASAJERO	NOMBRES	dbo	dbo	SELECT	YES
16	AGENCIA	PASAJERO	NOMBRES	dbo	dbo	UPDATE	YES
17	AGENCIA	PASAJERO	TELEFONO	dbo	dbo	INSERT	YES
18	AGENCIA	PASAJERO	TELEFONO	dbo	dbo	REFERENCES	YES
19	AGENCIA	PASAJERO	TELEFONO	dbo	dbo	SELECT	YES
20	AGENCIA	PASAJERO	TELEFONO	dbo	dbo	UPDATE	YES

Como notara se muestran todas las columnas de la tabla PASAJERO en **COLUMN_NAME**, los privilegios de cada columna especificado en **PRIVILEGE**.

CASO DESARROLLADO N° 4.41

Procedimiento almacenado del sistema que muestre las columnas de la tabla PASAJERO.



```
SP_COLUMNS PASAJERO
GO
```

En el script se ejecuta el procedimiento almacenado del sistema que permita mostrar las columnas de la tabla PASAJERO.

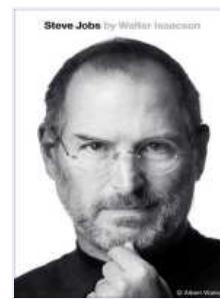
8 views

1 0

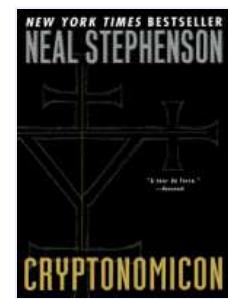
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

248 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 4.42

Procedimiento almacenado del sistema que muestre las bases de datos del servidor activo.

```
SP_DATABASES
GO
```

Este procedimiento tiene la capacidad de mostrar todas las bases de datos del servidor, tamaño asignado a cada una.

	DATABASE_NAME	DATABASE_SIZE	REMARKS
1	AGENCIA	12288	NULL
2	ALMACEN2011	2880	NULL
3	AUTOSERVIS	2880	NULL
4	BD_PRACTICANTES	2880	NULL
5	COLEGIO	4928	NULL
6	Contratos	3548	NULL
7	ejemplo	2880	NULL
8	INMOBILIARIA	2880	NULL
9	master	5376	NULL
10	model	3072	NULL
11	msdb	15872	NULL
12	Negocios2011	2880	NULL
13	PERUSEGURO	2880	NULL
14	planillas	2880	NULL
15	ReportServer	9728	NULL
16	ReportServerTempDB	3136	NULL
17	tempdb	8704	NULL
18	TUKASA	2880	NULL
19	VENTAS2011	2880	NULL
20	VETERINARIA	120000	NULL

CASO DESARROLLADO N° 4.43

Procedimiento almacenado del sistema que muestre el pasajero de código IDPASAJERO= tabla PASAJERO.



```
EXECUTE SP_EXECUTESQL
N'SELECT * FROM PASAJERO
WHERE IDPASAJERO = @COD',
N'@COD CHAR(5)',
@COD = 'P0001'
```

GO

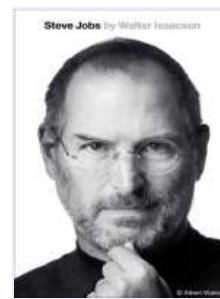
8 views

1 0

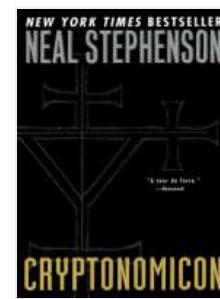
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CASO DESARROLLADO N° 4.44

Procedimiento almacenado del sistema que devuelva una lista de nombres de atributos y correspondientes para SQL Server, la puerta de enlace de la base de datos o el origen subyacente del servidor activo.

```
SP_SERVER_INFO
GO
```

El resultado de la ejecución se muestra en la siguiente imagen:

attribute_id	attribute_name	attribute_value
1	DBMS_NAME	Microsoft SQL Server
2	DBMS_VER	Microsoft SQL Server Yukon - 10.0.1600
3	OWNER_TERM	owner
4	TABLE_TERM	table
5	MAX_OWNER_NAME_LENGTH	128
6	TABLE_LENGTH	128
7	MAX_QUAL_LENGTH	128
8	COLUMN_LENGTH	128
9	IDENTIFIER_CASE	MIXED
10	TX_ISOLATION	2
11	COLLATION_SEQ	charset=iso_1 collation=Modern_Spanish_CI_AS
12	SAVEPOINT_SUPPORT	Y

CASO DESARROLLADO N° 4.45

Procedimiento almacenado del sistema que devuelva una lista de todos los índices y estaciones de la tabla PASAJERO.



```
SP_STATISTICS PASAJERO
GO
```

En la siguiente imagen se muestra el resultado de la ejecución:

TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	NON_UNIQUE	INDEX_QUALIFIER	INDEX_NAME	TYPE	SEQ_IN_INDEX
-----------------	-------------	------------	------------	-----------------	------------	------	--------------

8 views

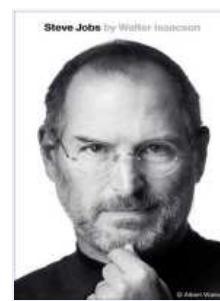
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

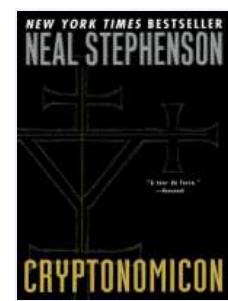
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon

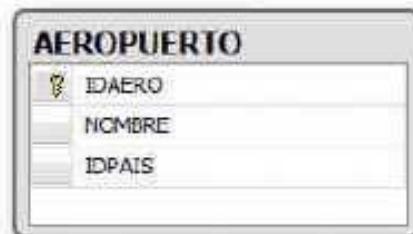


The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

250 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 4.46

Procedimiento almacenado del sistema que devuelva una lista de permisos de tabla (co DELETE, UPDATE, SELECT o REFERENCES) para la tabla AEROPUERTO.



```
EXECUTE SP_TABLE_PRIVILEGES 'AEROPUERTO'
GO
```

En el script se permiten mostrar los privilegios asignados a la tabla AEROPUERTO como lo siguiente imagen:

TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	GRANTOR	GRANTEE	PRIVILEGE	IS_GRANTABLE
1 AGENCIA	dbo	AEROPUERTO	dbo	dbo	DELETE	YES
2 AGENCIA	dbo	AEROPUERTO	dbo	dbo	INSERT	YES
3 AGENCIA	dbo	AEROPUERTO	dbo	dbo	REFERENCES	YES
4 AGENCIA	dbo	AEROPUERTO	dbo	dbo	SELECT	YES
5 AGENCIA	dbo	AEROPUERTO	dbo	dbo	UPDATE	YES

CASO DESARROLLADO N° 4.47

Procedimiento almacenado del sistema que devuelva las tablas de la base de datos AGENCIA.

```
USE AGENCIA
GO

SP_TABLES
GO
```

En la imagen siguiente se muestra el resultado de la ejecución del script:

TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	TABLE_TYPE	REMARKS
1 AGENCIA	dbo	AEROLINEA	TABLE	NULL
2 AGENCIA	dbo	AEROPUERTO	TABLE	NULL
3 AGENCIA	dbo	ASIENTO	TABLE	NULL
4 AGENCIA	dbo	AVION	TABLE	NULL
5 AGENCIA	dbo	Customers	TABLE	NULL
6 AGENCIA	dbo	MISPASAJEROS	TABLE	NULL
7 AGENCIA	dbo	Orders	TABLE	NULL

8 views

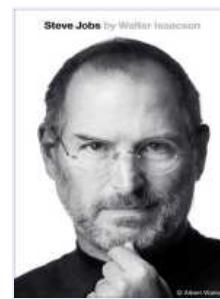
1 like

0 comments

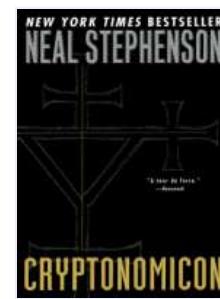
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT SQL

4.19. INSTRUCCIÓN EXECUTE Y SP_EXECUTESQL

La instrucción EXECUTE indica que SqlClient, ODBC, OLE DB o DB-Library debe ejecutar las instrucciones Transact-SQL preparadas, claro está que dichas instrucciones deben cumplir la sintaxis evaluada desde el motor de base de datos de SQL.

Sintaxis de EXECUTE:

EJECUCIÓN EXECUTE	
[{ EXEC EXECUTE }]	{
	[@RETURN_STATUS =]
	MODULE_NAME [,NUMBER] @MODULE_NAME_VAR
	[[@PARAMETER =] { VALUE
	@VARIABLE [OUTPUT]
	[DEFAULT]
	}
]

El SP_EXECUTESQL es un procedimiento almacenado que ejecuta una cadena Unicode en la que se incluye la sustitución de parámetros. Considere que el valor deberá ser mínimamente de tipo NVARCHAR. El procedimiento deberá ejecutarse con la instrucción EXEC obligatoriamente.

Sintaxis de SP_EXECUTESQL:

EJECUCIÓN SP_EXECUTESQL	
SP_EXECUTESQL [@STMT =] STMT	[
	, [@PARAMS=] N'@PARAMETER_NAME DATA_TYPE [OUT OUTPUT][,...N]'
	, [@PARAM1 =] 'VALUE1' [,...N]
]

Veamos dos ejemplos que explicarán el uso de la instrucción EXEC y el procedimiento almacenado SP_EXECUTESQL.

```
DECLARE @X VARCHAR(MAX)
SET @X='SELECT * FROM PASAJERO'
EXEC(@X)
GO
```

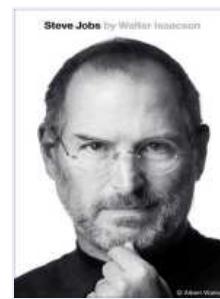
8 views

1 0

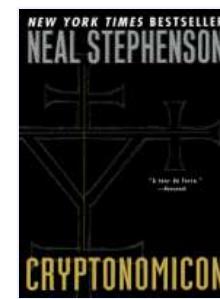
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

252 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el script se cambia de tipo de datos de la variable @N por NVARCHAR y se le asigna el valor de la consulta para que lo ejecute el procedimiento almacenado SP_EXECUTESQL con parámetros.

En la imagen siguiente se muestra el resultado de ambos scripts:

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4 P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5 P0013	LUCERO ERAZO	0004	388888888	LERAZO@GMAIL.COM
6 P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7 P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8 P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
9 P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
10 P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM
11 P0011	JANETH VILCHEZ	0002	388888888	JVILCHEZ@HOTMAIL.COM

4.20. PROCEDIMIENTOS ALMACENADOS DEFINIDOS POR EL USUARIO

Son procedimientos que se implementan en forma personalizada según la necesidad del usuario. Podrá emplear cualquier instrucción vista hasta el momento.

Sintaxis:

PROCEDIMIENTO DEFINIDO POR EL USUARIO CREATE PROCEDURE

```
CREATE PROC | PROCEDURE [PROPIETARIO.] NOMBRE_PROCEDIMIENTO
    [ @PARAMETRO TIPO_DATOS [ = DEFAULT ] [ OUT | OUTPUT ] [ READONLY ]
AS
[BEGIN]
    CUERPO_DEL_PROCEDIMIENTO
[END]
```

CASO DESARROLLADO N° 4.48

Procedimiento almacenado que permite devolver los 5 costos más altos registrados en la tabla RESERVA.

RESERVA		
IDRESERVA		
COSTO		
FECHA		

```
IF OBJECT_ID('TOP5RESERVAS') IS NOT NULL
```

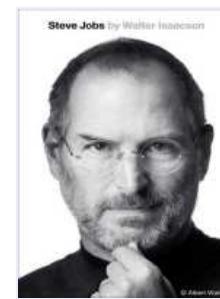
8 views

1 0

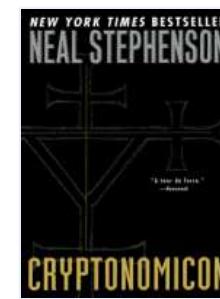
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

En el script se implementa el procedimiento almacenado TOP5RESERVAS el cual tendrá que mostrar los 5 costos más altos registrados en la tabla RESERVA, hay que tener en cuenta que usar la cláusula TOP para dicho proceso.

Se asigna el número de registros a 5 con la con el siguiente script **SET ROWCOUNT 5** implementa la sentencia SELECT que permite mostrar los registros ordenados por la columna COSTO limitados por la asignación del ROWCOUNT en 5.

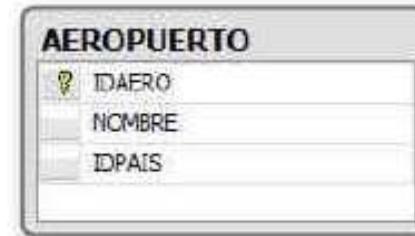
Para la ejecución del procedimiento almacenado colocar el siguiente script **EXEC TOP5RES**

La imagen siguiente muestra el resultado de la ejecución del procedimiento almacenado:

IDRESERVA	COSTO	FECHA
1 10	11008.6365	2012-04-02
2 23	11008.6365	2012-04-02
3 21	7950.6818	2012-01-15
4 8	7950.6818	2012-01-15
5 11	7339.0911	2012-04-08

CASO DESARROLLADO N° 4.49

Implemente un Procedimiento Almacenado que permita mostrar los registros de la Tabla AEROPUERTO donde se visualice el nombre del país, defina de manera adecuada la cabecera del listado.



```

IF OBJECT_ID('MUESTRAAEROPUERTO') IS NOT NULL
BEGIN
    DROP PROCEDURE MUESTRAAEROPUERTO
END
GO

CREATE PROCEDURE MUESTRAAEROPUERTO
AS
    SELECT AER.IDAERO AS [CODIGO],
           AER.NOMBRE AS [AEROPUERTO],
           PAI.PAIS AS [PAIS]
      FROM AEROPUERTO AER
     JOIN PAIS PAI ON AER.IDPAIS=PAI.IDPAIS
GO
  
```

En el script se implementa el procedimiento almacenado que tiene por misión mostrar las columnas CODIGO, AEROPUERTO y PAIS de la tabla AEROPUERTO y adicionalmente mostrar el nombre del país donde la tabla del mismo nom-

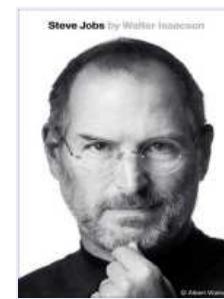
8 views

1 0

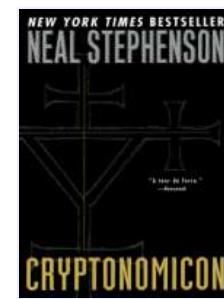
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

254

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Otra forma de ejecutar el mismo procedimiento es colocando directamente el nombre del procedimiento sin necesidad de colocar la instrucción EXEC así como se muestra en el siguiente script:

```
MUESTRAAEROPUERTO
GO
```

El resultado de la ejecución del procedimiento almacenado se muestra en la siguiente imagen:

	CODIGO	AEROPUERTO	PAIS
1	AE01	BARILOCHE	ARGENTINA
2	AE02	MAR DEL PLATA	ARGENTINA
3	AE03	JORGE CHAVEZ	PERU
4	AE04	SANTIAGO	CHILE
5	AE05	AICM	MEXICO
6	AE06	JOSE JOAQUIN DE OLMEDO	ECUADOR
7	AE07	SIMON BOLVAR	VENEZUELA
8	AE08	SAO PAULO CONGONHAS	BRASIL
9	AE09	SILVIO PETTIROSSI	PARAGUAY
10	AE10	CARRASCO PUERTA DEL SUR	URUGUAY

CASO DESARROLLADO N° 4.50

Implemente un Procedimiento Almacenado que permita mostrar los registros de la Tabla PASAJERO donde se visualice el nombre del Distrito por medio de la función MUESTRAPAIS(), definir una adecuada cabecera del listado.



Antes de implementar el procedimiento almacenado necesitamos ver la implementación de la función **MUESTRAPAIS** que tendrá por misión devolver el nombre del país de acuerdo a un código de país. En el script siguiente se muestra su implementación:

```
IF OBJECT_ID('MUESTRAPAIS') IS NOT NULL
BEGIN
    DROP FUNCTION DBO.PASAJEROSXPAIS
END
GO
```

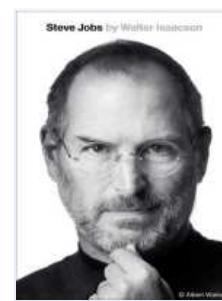
8 views

1 0

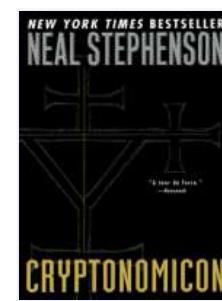
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

Como notará en el script primero se verifica la existencia de la función, si existe se eliminará; de lo contrario se podrá ejecutar la función sin ningún problema, ahora si usted ya lo había definido anteriormente no habrá problema.

Ahora, mostraremos el script perteneciente a la implementación del procedimiento almacenado que soluciona el caso:

Empezaremos por verificar si el procedimiento **MUESTRAPASAJERO** existe en la base de datos, en caso lo fuera se eliminará de la base para eso ejecute el siguiente script:

```
IF OBJECT_ID('MUESTRAPASAJERO') IS NOT NULL
BEGIN
    DROP PROCEDURE MUESTRAPASAJERO
END
GO

CREATE PROCEDURE MUESTRAPASAJERO
AS
    SELECT PAS.IDPASAJERO AS [CODIGO],
           PAS.NOMBRES AS [PASAJERO],
           dbo.MUESTRAPAIS(PAS.IDPAIS) AS [PAIS],
           PAS.EMAIL
      FROM PASAJERO PAS
GO
```

El procedimiento almacenado **MUESTRAPASAJERO** adiciona además de sus columnas principales una nueva columna que es la invocación de la función creada en el script anterior; claro está que siempre se debe especificar el proveedor de datos al momento de invocarlo (**dbo**) y también su parámetro que en este caso es el identificador que se obtiene desde la tabla **PASAJERO**, como notará la función devolverá un valor que se convierte automáticamente en una columna al momento de mostrar el resultado.

Para ejecutar el procedimiento debe colocar el siguiente script:

```
EXEC MUESTRAPASAJERO
GO
```

La siguiente imagen muestra el resultado de la ejecución:

	CODIGO	PASAJERO	PAÍS	EMAIL
1	P0091	ANGELA TORRES LAZARO	PERU	ATORRES@HOTMAIL.COM
2	P0002	FERNANDA TORRES LAZARO	PERU	FTORRES@HOTMAIL.COM
3	P0003	MARIA ZAMORA MEJIA	BRASIL	MZAMORA@GMAIL.COM
4	P0004	GUADALUPE ACOSTA FERRER	ARGENTINA	GACOSTA@HOTMAIL.COM
5	P0005	LUZ LAZARO MENOR	PERU	LLAZARO@GMAIL.COM

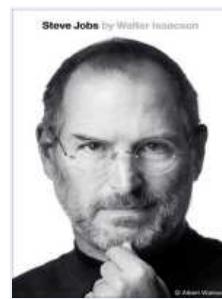
8 views

1 0

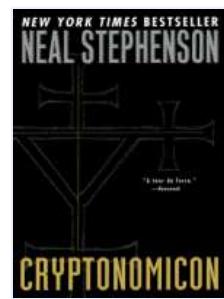
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

256 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 4.51

Implemente un Procedimiento Almacenado que muestre el listado de los países y su total de pasajeros.



```
--1.
IF OBJECT_ID('PASAJEROSXPAIS') IS NOT NULL
BEGIN
    DROP PROCEDURE PASAJEROSXPAIS
END
GO

--2.
CREATE PROCEDURE PASAJEROSXPAIS
AS
    SELECT PAI.PAIS AS [PAIS] ,COUNT(*) AS [TOTAL]
    FROM PASAJERO PAS
    JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
    GROUP BY PAI.PAIS
GO
```

En el punto dos se implementa el procedimiento **PASAJEROSXPAIS** en donde se usa la función **COUNT** que permitirá el conteo de los pasajeros según la agrupación especificada en la cláusula **BY PAI.PAIS**.

Para visualizar el resultado deberá ejecutar el siguiente script:

```
EXEC PASAJEROSXPAIS
GO
```

La imagen siguiente muestra el resultado de la ejecución del procedimiento:

	PAÍS	TOTAL
1	ARGENTINA	1
2	BRASIL	1
3	CHILE	1
4	ECUADOR	2
5	HAITI	1

8 views

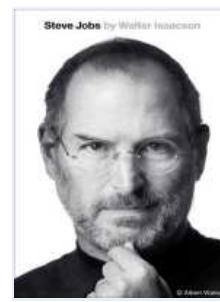
1 like

0 comments

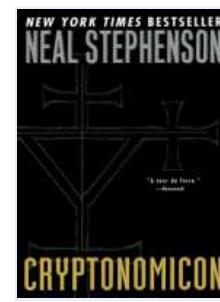
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CASO DESARROLLADO N° 4.52

Implemente un Procedimiento Almacenado que permite mostrar el monto acumulado por la tabla PAGO.



```
--1.
IF OBJECT_ID('MONTOXAÑO') IS NOT NULL
BEGIN
    DROP PROCEDURE MONTOXPAIS
END
GO

--2.
CREATE PROCEDURE MONTOXAÑO
AS
    SELECT YEAR(PAG.FECHA) AS [AÑO],
           SUM(PAG.MONTO) AS [MONTO ACUMULADO]
    FROM PAGO PAG
    GROUP BY YEAR(PAG.FECHA)
GO
```

En el punto dos se implementa el procedimiento almacenado **MONTOXAÑO** en la cual se usa **SUM(PAG.MONTO)** para acumular los montos de la tabla **PAGO** según la agrupación **YEAR(PAG.FECHA)**.

Ejecutar el siguiente script para ver el resultado **EXEC MONTOXAÑO**

La imagen siguiente muestra el resultado de la ejecución:

	AÑO	MONTO ACUMULADO
1	2011	5190.00
2	2012	6900.00

CASO DESARROLLADO N° 4.53

Implemente un Procedimiento almacenado que permite mostrar los países que aún no tiene registrado.

[PASAJERO](#)
[PAÍS](#)

8 views

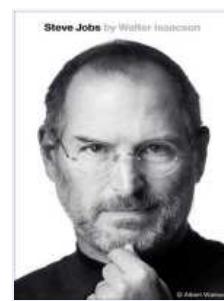
1 like

0 comments

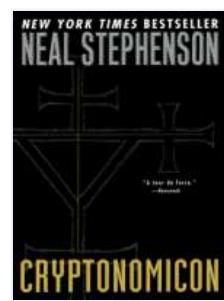
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

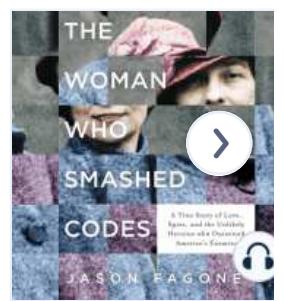
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

258 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

```
--1.
IF OBJECT_ID('PAISNOREGISTRADO') IS NOT NULL
BEGIN
    DROP PROCEDURE PAISNOREGISTRADO
END
GO

--2.
CREATE PROCEDURE PAISNOREGISTRADO
AS
    SELECT PAI.IDPAIS,PAI.PAIS
    FROM PAIS PAI
    LEFT JOIN PASAJERO PAS ON PAI.IDPAIS=PAS.IDPAIS
    WHERE PAS.IDPAIS IS NULL
GO
```

En el punto dos se implementa el procedimiento almacenado **PAISNOREGISTRADO** en la cláusula **LEFT JOIN** para poder mostrar aquellos países que no tienen pasajeros a través de **WHERE PAS.IDPAIS IS NULL**.

Para ejecutar el procedimiento almacenado ejecutar el siguiente script **EXEC PAISNOREGI**

La imagen siguiente muestra el resultado de la ejecución del procedimiento.

IDPAIS	PAÍS
1	VENEZUELA
2	URUGUAY
3	BOLIVIA
4	HONDURAS
5	EEUU
6	PUERTO RICO

4.21. PROCEDIMIENTOS ALMACENADOS CON PARÁMETROS DE ENTRADA

CASO DESARROLLADO N° 4.54

Implemente un Procedimiento almacenado que permita mostrar las reservas de un d pasajero para este caso deberá considerar como parámetro de entrada, el nombre del pasa de manera adecuada la cabecera del listado e implemente un mensaje de error en el pasajero no existe.

8 views

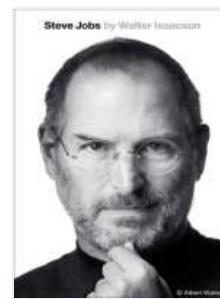
1 like

0 comments

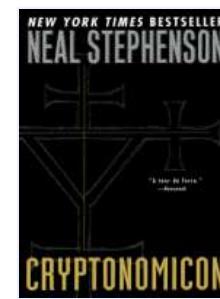
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

CAP. 4: PROGRAMACIÓN TRANSACT S

```
--1.
IF OBJECT_ID('RESERVASXPASAJERO') IS NOT NULL
BEGIN
    DROP PROCEDURE RESERVASXPASAJERO
END
GO

--2.
CREATE PROCEDURE RESERVASXPASAJERO(@PASAJERO VARCHAR(30))
AS
BEGIN
    IF EXISTS(SELECT IDPASAJERO
              FROM PASAJERO
              WHERE NOMBRES=@PASAJERO)
        SELECT RES.*,
               FROM RESERVA RES
               JOIN PAGO PAG ON RES.IDRESERVA=PAG.IDRESERVA
               WHERE PAG.IDPASAJERO=(SELECT IDPASAJERO
                                      FROM PASAJERO
                                      WHERE NOMBRES=@PASAJERO)
    ELSE
        PRINT 'EL PASAJERO NO ESTA REGISTRADO EN LA BASE'
END
GO
```

En el punto dos se implementa el procedimiento almacenado **RESERVASXPASAJERO** el cual tiene un parámetro de entrada a **@PASAJERO** en la cual se enviará al procedimiento un nombre de pasajero. Luego se condiciona la existencia del pasajero dentro de la tabla del mismo nombre ya que si no existe se emitirá el mensaje **EL PASAJERO NO ESTA REGISTRADO EN LA BASE**, caso contrario se buscará las reservas realizadas por dicho pasajero, para esto debe unir las tablas **RESERVA** y **PAGO**, que para determinar las reservas por pasajero se tiene que pasar por la tabla **PAGO**.

Para ejecutar el procedimiento almacenado con parámetros de entrada se debe colocar el siguiente script:

```
EXEC RESERVASXPASAJERO @PASAJERO='FERNANDA TORRES LAZARO'
```

O también de la siguiente manera:

```
EXEC RESERVASXPASAJERO 'FERNANDA TORRES LAZARO'
```

8 views

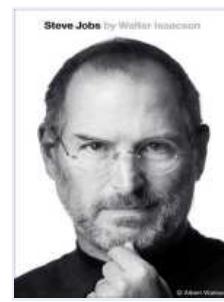
1 like

0 comments

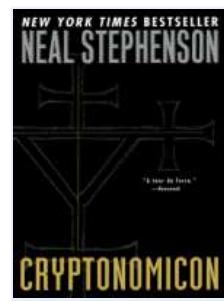
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

260 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 4.55

Implemente un Procedimiento almacenado que permita registrar a un nuevo pasajero, para ello deberá definir como parámetros de entrada todos los campos referentes al pasajero a excepción de IDPAIS; aquí deberá ingresar el nombre del país, en caso no exista emitir un mensaje como “PASAJERO REGISTRADO EN LA BASE”. Finalmente, si el pasajero se registra correctamente emitir un “PASAJERO REGISTRADO CON EXITO”.



```
--1.
IF OBJECT_ID('NUEVOPASAJERO') IS NOT NULL
BEGIN
    DROP PROCEDURE DBO.NUEVOPASAJERO
END
GO
--2.
CREATE PROCEDURE NUEVOPASAJERO(
    @IDPAS CHAR(5), @NOM VARCHAR(40), @PAI VARCHAR(40),
    @TEL VARCHAR(15), @EMA VARCHAR(40)
)
AS
BEGIN TRY
    IF EXISTS(SELECT IDPAIS FROM PAIS WHERE PAIS=@PAI)
    BEGIN
        DECLARE @IDPAI CHAR(4)
        SELECT @IDPAI = IDPAIS FROM PAIS WHERE PAIS=@PAI
        INSERT INTO PASAJERO
            VALUES (@IDPAS, @NOM, @IDPAI, @TEL, @EMA)
        PRINT 'PASAJERO REGISTRADO CON EXITO'
    END
    ELSE
        PRINT 'PAIS NO REGISTRADO EN LA BASE'
END TRY
BEGIN CATCH
    PRINT 'OCURRIO UN ERROR AL INSERTAR'
END CATCH
GO
```

En el punto dos se implementa el procedimiento almacenado NUEVOPASAJERO que permite registrar a un nuevo pasajero en la tabla PASAJERO, pero debemos tener en cuenta que dicha ta-

8 views

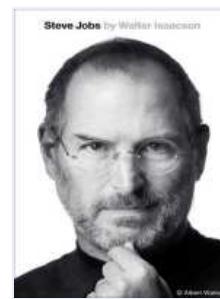
1 like

0 comments

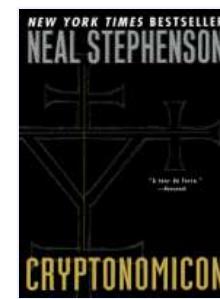
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

Luego se implementó el BEGIN TRY para controlar los errores esto se activará cuando ocurrir insertar el nuevo pasajero podría suceder cuando el Primary Key se duplique, el país ingresar el parámetro @PAI también tiene que evaluarse ya que se debería enviar un mensaje si existe en la tabla del mismo nombre.

```
BEGIN TRY
    IF EXISTS(SELECT IDPAIS FROM PAIS WHERE PAIS=@PAI)
        BEGIN
```

Seguidamente se declara la variable @IDPAI que capturará el IDPAIS desde la consulta.

```
DECLARE @IDPAI CHAR(4)
SELECT @IDPAI = IDPAIS FROM PAIS WHERE PAIS=@PAI
```

Luego se aplica la inserción a la tabla PASAJERO pasando por la comprobación de la existencia en su tabla y haber capturado el IDPAIS ya que en el procedimiento no se ingresó el código nombre del país.

```
INSERT INTO PASAJERO
    VALUES (@IDPAS,@NOM,@IDPAI,@TEL,@EMA)
    PRINT 'PASAJERO REGISTRADO CON EXITO'
END
ELSE
    PRINT 'PAIS NO REGISTRADO EN LA BASE'
```

Finalmente, se tiene que ejecutar el procedimiento almacenado para lo cual mostramos dos ejemplos:

```
EXEC NUEVOPASAJERO 'P0011','JANETH CRUZ',
    'HONDURAS','999999999','JCRUZ@HOTMAIL.COM'
GO
```

Y el segundo es:

```
EXEC NUEVOPASAJERO 'P0012','JANETH CRUZ',
    'HAITI','999999999','JCRUZ@HOTMAIL.COM'
GO
```

La imagen siguiente muestra el resultado con la segunda ejecución.

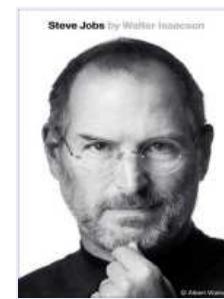
8 views

1 0

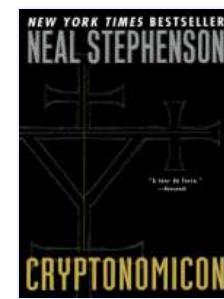
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

262 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 4.56

Implemente un Procedimiento que permita registrar un nuevo País para lo cual deberá comprobar si existe el parámetro de entrada al nombre del país, aquí se deberá comprobar que dicho país no esté registrado antes si fuera el caso emitir un mensaje de “PAÍS YA REGISTRADO”, el código de autogenerado; por lo tanto, no se ingresará como parámetro. Finalmente, si todo es correcto emitir un mensaje de “PAÍS REGISTRADO CORRECTAMENTE”.

PAÍS	
Column Name	Condensed Type
IDPAIS	char(4)
PAIS	varchar(30)

```

IF OBJECT_ID('NUEVOPAIS') IS NOT NULL
BEGIN
    DROP PROCEDURE DBO.NUEVOPAIS
END
GO

CREATE PROCEDURE NUEVOPAIS(@PAIS VARCHAR(40))
AS
--1.
SET ROWCOUNT 1
DECLARE @IDPAI CHAR(4),@NUEVOID CHAR(4)
SELECT @IDPAI=IDPAIS FROM PAIS ORDER BY PAIS.IDPAIS DESC
IF LEN(@IDPAI+1)=1 SET @NUEVOID='000'+CAST(@IDPAI+1 AS CHAR(4))
IF LEN(@IDPAI+1)=2 SET @NUEVOID='00'+CAST(@IDPAI+1 AS CHAR(4))
IF LEN(@IDPAI+1)=3 SET @NUEVOID='0'+CAST(@IDPAI+1 AS CHAR(4))

--2.
IF NOT EXISTS(SELECT PAIS FROM PAIS WHERE PAIS=@PAIS)
BEGIN
    INSERT INTO PAIS VALUES (@NUEVOID,@PAIS)
    PRINT 'PAÍS REGISTRADO CORRECTAMENTE'
END
ELSE
    PRINT 'EL PAÍS YA SE ENCUENTRA REGISTRADO'
GO

```

En el punto uno se implementa el script que permitirá autogenerar el código del nuevo país en la tabla, hay que tener en cuenta que en el parámetro del procedimiento almacenado se pasa el nombre del país a través del parámetro @PAIS por que la idea del script es autogenerar

Lo primero que debemos entender es que necesitamos un script que permita capturar el país registrado en la tabla y adicionarle un número para generar el nuevo código: por ejemplo

8 views

1 like

0 comments

eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)

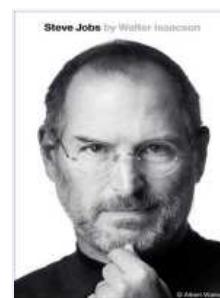
Save

Embed

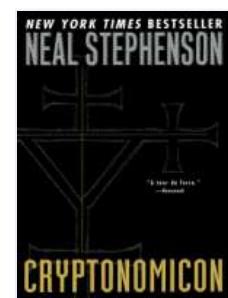
Share

Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

Permite capturar el IDPAIS desde el ingreso del nombre del país, recordemos que no se ingresa sólo el nombre del nuevo país a registrar por tal motivo se autogenera el código obtenido anterior, entonces el nuevo código se obtendrá cuando se asigne una de las acciones específicas de la siguiente condicional:

```
IF LEN(@IDPAI+1)=1 SET @NUEVOID='000'+CAST(@IDPAI+1 AS CHAR(4))
IF LEN(@IDPAI+1)=2 SET @NUEVOID='00'+CAST(@IDPAI+1 AS CHAR(4))
IF LEN(@IDPAI+1)=3 SET @NUEVOID='0'+CAST(@IDPAI+1 AS CHAR(4))
```

En el punto dos se verifica la existencia del país ingresado desde el parámetro del procedimiento almacenado realizado a la tabla PAIS. Una vez verificado que dicho país no esté registrado y obtenido el ID del país nuevo se procederá a registrar dicho país.

```
IF NOT EXISTS(SELECT PAIS FROM PAIS WHERE PAIS=@PAIS)
BEGIN
    INSERT INTO PAIS VALUES (@NUEVOID,@PAIS)
    PRINT 'PAIS REGISTRADO CORRECTAMENTE'
END
```

Para probar si se inserta correctamente el país en su tabla se deberá ejecutar el siguiente comando:

```
EXEC NUEVOPAIS 'HAITI'
GO
```

Ahora, podrá revisar el listado de los países y poder comprobar si se registró o no el país ingresado.

```
(1 filas afectadas)
PAIS REGISTRADO CORRECTAMENTE
```

CASO DESARROLLADO Nº 4.57

Implemente un Procedimiento Almacenado que permita actualizar los datos de un pasajero para lo cual deberá definir como parámetros de entrada los campos de la tabla PASAJERO teniendo en cuenta que al actualizar no debe enviar el código del país sino su nombre del país existencia del mismo con un mensaje de "País no registrado en la base". Finalmente, si todo es correcto enviar un mensaje de "**PASAJERO ACTUALIZADO CON EXITO**".



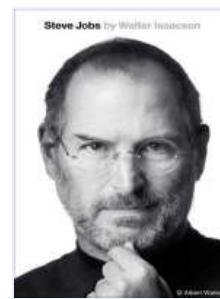
8 views

1 0

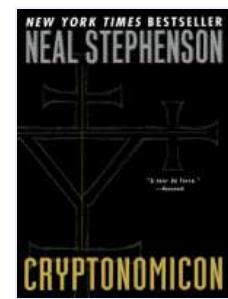
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

264

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

```
--1.
IF OBJECT_ID('ACTUALIZAPASAJERO') IS NOT NULL
BEGIN
    DROP PROCEDURE DBO.ACTUALIZAPASAJERO
END
GO

--2.
CREATE PROCEDURE ACTUALIZAPASAJERO(
    @IDPAS CHAR(5),@NOM VARCHAR(40),@PAI VARCHAR(40),
    @TEL VARCHAR(15),@EMA VARCHAR(40)
)
AS
BEGIN TRY
    IF EXISTS(SELECT IDPAIS FROM PAIS WHERE PAIS=@PAI)
    BEGIN
        DECLARE @IDPAI CHAR(4)
        SELECT @IDPAI = IDPAIS FROM PAIS WHERE PAIS=@PAI
        UPDATE PASAJERO
        SET NOMBRES=@NOM,
            IDPAIS=@IDPAI,
            TELEFONO=@TEL,
            EMAIL=@EMA
        WHERE IDPASAJERO=@IDPAS
        PRINT 'PASAJERO ACTUALIZADO CON EXITO'
    END
    ELSE
        PRINT 'PAIS NO REGISTRADO EN LA BASE'
    END TRY
    BEGIN CATCH
        PRINT 'OCURRIO UN ERROR AL ACTUALIZAR'
    END CATCH
GO
```

En el punto dos se implementa el procedimiento almacenado ACTUALIZAPASAJERO que misión actualizar todos los valores registrados en la tabla PASAJERO según el código del pa

Primero, se verifica la existencia del país ingresado como parámetro en el procedimiento asigna el IDPAIS a la variable @IDPAI para que se pueda registrar en la tabla PASAJERO.

```
IF EXISTS(SELECT IDPAIS FROM PAIS WHERE PAIS=@PAI)
BEGIN
    DECLARE @IDPAI CHAR(4)
    SELECT @IDPAI = IDPAIS FROM PAIS WHERE PAIS=@PAI
    UPDATE PASAJERO
    SET NOMBRES=@NOM,
        IDPAIS=@IDPAI,
        TELEFONO=@TEL,
```

8 views

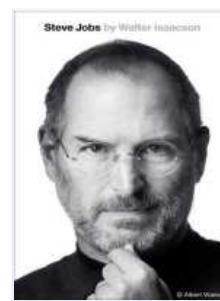
1 like

0 comments

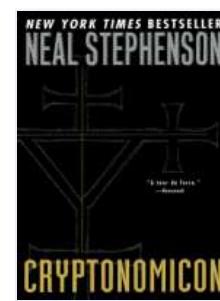
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

Finalmente, se muestra el mensaje siempre y cuando se encuentre un error en la actualización el mensaje será visualizado sólo si se implementó el BEGIN TRY y BEGIN CATCH respectivo.

```
BEGIN CATCH
    PRINT 'OCURRIO UN ERROR AL ACTUALIZAR'
END CATCH
```

Para ejecutar el procedimiento almacenado se debe colocar el siguiente script:

```
EXEC ACTUALIZAPASAJERO 'P0011', 'JANETH CRUZ',
'HAITI', '999999999', 'JCRUZ@HOTMAIL.COM'
GO
```

4.22. PROCEDIMIENTOS ALMACENADOS CON PARÁMETROS DE ENTRADA Y SALIDA

CASO DESARROLLADO N° 4.58

Implemente un Procedimiento Almacenado que retorne el total de RESERVAS de un determinado país. Asigne como parámetro de entrada al nombre del país y como parámetro de salida al total de proveedores.



```
IF OBJECT_ID('TOTALRESERVAS') IS NOT NULL
BEGIN
    DROP PROCEDURE DBO.TOTALRESERVAS
END
GO
```

8 views

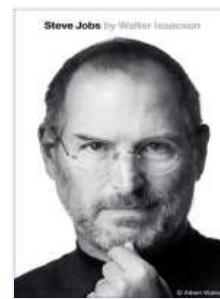
1 like

0 comments

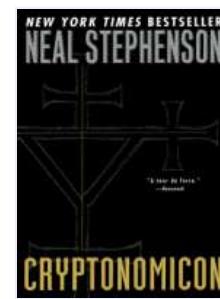
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

266 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Cuando se definió parámetros de entrada en los procedimientos almacenados no tuvo ningún especial para su identificación, cuando se define un parámetro de salida se debe colocar OUTPUT. La demás implementación es similar a los procedimientos anteriores.

Lo que si varía es la forma de ejecución del procedimiento almacenado, aquí se muestra ejecución:

```
DECLARE @T INT
EXEC TOTALRESERVAS 'ANGELA TORRES LAZARO',@TOTAL=@T OUTPUT
PRINT      'EL PASAJERO TIENE CANTIDAD DE RESERVAS ES:' +
CAST(@T AS VARCHAR(10))
GO
```

Considere que cuando existe un parámetro de salida alguien debe recepcionar su valor desde decir desde la llamada; en el caso se declara la variable @T que tiene por finalidad obtener de salida desde el procedimiento almacenado dicha especificación se realizó en EXEC TOTALRESERVAS 'ANGELA TORRES LAZARO',@TOTAL=@T OUTPUT.

Ahora, dicha variable @T se podrá imprimir. La imagen siguiente muestra el resultado de la ejecución del procedimiento almacenado.

CASO DESARROLLADO N° 4.59

Implemente el Procedimiento almacenado que permita determinar el número de reservas acumulado de las reservas realizadas en un determinado país, este deberá ingresarse por su nombre. Use parámetros de salida.

PASAJERO	
IDPASAJERO	
NOMBRES	
IDPAIS	
TELEFONO	
EMAIL	

PAIS	
Column Name	Condensed Type
IDPAIS	char(4)
PAIS	varchar(30)

PAGO	
NUMPAGO	
IDRESERVA	
IDPASAJERO	
FECHA	
MONTO	

```
IF OBJECT_ID('TPAGOSXPAIS') IS NOT NULL
BEGIN
    DROP PROCEDURE DBO.TPAGOSXPAIS
END
GO

CREATE PROCEDURE TPAGOSXPAIS
    @PAIS VARCHAR(40),
```

8 views

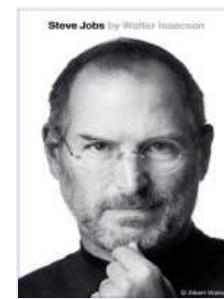
1 0

eMacro - Programación Transact SQL Server 2012-4.pdf

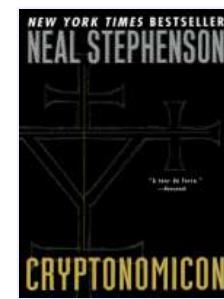
Uploaded by [anon_61286589](#)[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Betrayal, and the Unlikely Heroine who Outwitted America's Enemies

CAP. 4: PROGRAMACIÓN TRANSACT S

Para ejecutar el procedimiento almacenado deberá colocar el siguiente script:

```
DECLARE @T INT,@M MONEY
DECLARE @PAIS VARCHAR(40)='PERU'
EXEC TPAGOSXPAIS @PAIS,@TOTAL=@T OUTPUT,@ACUMULADO=@M OUTPUT
PRINT      'EL PAÍS '+@PAIS+' TIENE '+CAST(@T AS VARCHAR(10))+' RESERVAS '+
          'Y ACUMULA UN MONTO DE: '+CAST(@M AS VARCHAR(10))
GO
```

Los parámetros de un procedimiento almacenado pueden ser de entrada o salida sirviendo declaraciones, pero de acuerdo a la cantidad de parámetros, ya sea de entrada o salida, tiene que enviar cuando se intenta ejecutar el procedimiento, ahora en este caso particular @T y @M para recepcionar los valores devueltos desde el procedimiento y @PAIS tendrá nombre del país que será enviado al procedimiento y será tomado como parámetro de entrada.

En caso se omita un parámetro de salida el motor de base de datos emite el siguiente mensaje:

```
Mens 201, Nivel 16, Estado 4. Procedimiento TPAGOSXPAIS. Línea 0
El procedimiento o la función 'TPAGOSXPAIS' esperaba el parámetro '@ACUMULADO',
que no se ha especificado.
```

Si todo es correcto y se envían los parámetros necesarios del procedimiento entonces es visualizado los resultados esperados.

```
EXEC TPAGOSXPAIS @PAIS,@TOTAL=@T OUTPUT,@ACUMULADO=@M OUTPUT
```

Como notará al obtener el valor resultante del procedimiento se tiene que especificar de manera: @PARAMETRO_SALIDA = @VARIABLE_LOCAL OUTPUT. La imagen siguiente muestra el resultado de ejecución correcta:

```
EL PAÍS PERU TIENE 5 RESERVAS Y ACUMULA UN MONTO DE: 6200.00
```

CASO DESARROLLADO N° 4.60

Procedimiento almacenado que muestre todos los procedimientos almacenados registrados en la base de datos AGENCIA.

```
SELECT * FROM SYS.SYSOBJECTS P
WHERE P.XTYPE='P'
GO
```

La imagen siguiente muestra el resultado de la ejecución de la consulta:

8 views

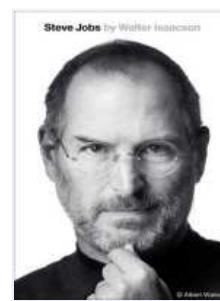
1 like

0 comments

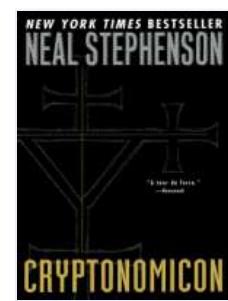
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

268 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

4.23. MODIFICAR LA IMPLEMENTACIÓN DE UN PROCEDIMIENTO ALMACENADO

La modificación de un procedimiento almacenado se realiza sólo cuando ha pasado por de la instrucción CREATE. Los cambios se realizan a todo el script contenido dentro del procedimiento almacenado.

Sintaxis:

PROCEDIMIENTO DEFINIDO POR EL USUARIO ALTER PROCEDURE

```
ALTER PROC | PROCEDURE [PROPIETARIO.] NOMBRE_PROCEDIMIENTO
    [ @PARAMETRO TIPO_DATOS [ = DEFAULT ] [ OUT | OUTPUT ] [READONLY] ]
AS
[BEGIN]
    CUERPO_DEL_PROCEDIMIENTO
[END]
```

CASO DESARROLLADO N° 4.61

Modifique el procedimiento almacenado que determinaba el número de reservas acumulado de las reservas realizadas en un determinado país, el cambio a realizar es que calcular el acumulado se tiene que calcular el promedio de los montos y el máximo de los m̄es. Para m̄s información vea el CASO DESARROLLADO N° 4.59. Use parámetros de salida.

PASAJERO	
IDPASAJERO	
NOMBRES	
IDPAIS	
TELEFONO	
EMAIL	

PAÍS	
Column Name	Condensed Type
IDPAIS	char(4)
PAÍS	varchar(30)

PAGO	
IDLMPAGO	
IDRESERVA	
IDPASAJERO	
FECHA	
MONTO	

```
ALTER PROCEDURE TPAGOSXPAIS
    @PAIS VARCHAR(40),
    @TOTAL INT OUTPUT,
    @PROMEDIO MONEY OUTPUT,
    @MAYOR MONEY OUTPUT
AS
SELECT @TOTAL= COUNT(*),
       @PROMEDIO=AVG(MONTO),
       @MAYOR=MAX(MONTO)
FROM PAÍS PAI
JOIN PASAJERO PAS ON PAI.IDPAIS=PAS.IDPAIS
JOIN PAGO PAG ON PAG.IDPASAJERO=PAS.IDPASAJERO
```

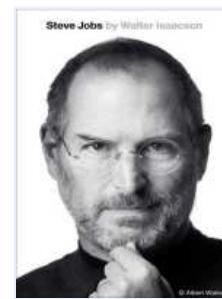
8 views

1 0

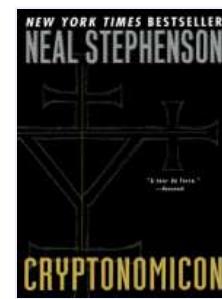
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

Para la ejecución del procedimiento almacenado se tiene que ejecutar el siguiente script:

```

DECLARE @T INT,@P MONEY, @M MONEY
DECLARE @PAIS VARCHAR(40)='PERU'
EXEC TPAGOSXPAIS @PAIS,@TOTAL=@T OUTPUT,@PROMEDIO=@P OUTPUT,
@MAYOR=@M OUTPUT
PRINT 'PAIS EVALUADO      >'+@PAIS
PRINT '-----'
PRINT 'TOTAL DE RESERVAS    > '+CAST(@T AS VARCHAR(10))
PRINT 'PROMEDIO DE MONTOS   > '+CAST(@P AS VARCHAR(10))
PRINT 'MAYOR MONTO          > '+CAST(@M AS VARCHAR(10))
GO

```

La imagen siguiente muestra el resultado de la ejecución:

PAIS	EVALUADO	>PERU
TOTAL DE RESERVAS	> 5	
PROMEDIO DE MONTOS	> 1240.00	
MAYOR MONTO	> 1800.00	

4.24. ELIMINAR PROCEDIMIENTOS ALMACENADOS

Quita uno o más procedimientos almacenados de la base de datos actual, hay que considerar que no hay forma de poder recuperarlo una vez eliminado.

Sintaxis:

PROCEDIMIENTO DEFINIDO POR EL USUARIO DROP PROCEDURE

```
DROP [ PROC | PROCEDURE ] [ PROPIETARIO. ] NOMBRE_PROCEDIMIENTO
```

CASO DESARROLLADO N° 4.62

Implemente un script que permita eliminar el procedimiento almacenado usando el nombre existencia del mismo. Para este caso tomaremos en referencia el procedimiento almacenado TPAGOSXPAIS.

```

IF OBJECT_ID('TPAGOSXPAIS') IS NOT NULL
BEGIN
    DROP PROCEDURE DBO.TPAGOSXPAIS
    PRINT 'PROCEDIMIENTO ELIMINADO CORRECTAMENTE'
END

```

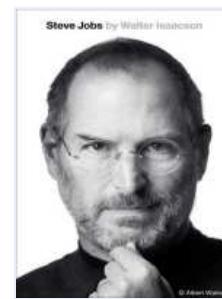
8 views

1 0

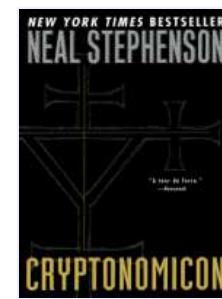
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

270 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

4.26. VISUALIZAR LA IMPLEMENTACIÓN DE UN PROCEDIMIENTO ALMACENADO

En muchas ocasiones nos tocará procedimientos almacenados que ya se encuentran ejecutando en la base de datos pero no se visualiza su implementación, para poder modificarlos se tiene que acceder a la fuente del procedimiento.

Sintaxis:

**PROCEDIMIENTO DEFINIDO POR EL USUARIO
SP_HELPTEXT**

```
SP_HELPTEXT NOMBRE_PROCEDIMIENTO
GO
```

CASO DESARROLLADO N° 4.63

Un script que permite visualizar la implementación del procedimiento almacenado TPAGOSXPAIS

```
SP_HELPTEXT TPAGOSXPAIS
GO
```

La imagen siguiente muestra la implementación del procedimiento almacenado TPAGOSXPAIS

```

Text
1
2 CREATE PROCEDURE TPAGOSXPAIS
3   @PAIS VARCHAR(40)
4   @TOTAL INT OUTPUT,
5   @ACUMULADO MONEY OUTPUT
6   AS
7   SELECT @TOTAL=COUNT(*),
8   @ACUMULADO=SUM(MONTO)
9   FROM PAIS PAI
10  JOIN PASAJERO PAS ON PAI.IDPAI
11  JOIN PAGO PAG ON PAG.IDPASAJ
12  WHERE PAI.PAIS=@PAIS

```

Desde aquí se podrá seleccionar el contenido visualizado y arrastrarlo al editor de consultas para incluir su contenido.

4.27. PROCEDIMIENTOS ALMACENADOS Y CURSORES

Cuando se habló del tema Cursos resultaba un poco incómodo ejecutarlos ya que se tenía que seleccionar todo el bloque que encierra al cursor o ejecutarlo por bloques, los procedimientos almacenados reducen el tiempo de ejecución llamando al cursor por medio del nombre.

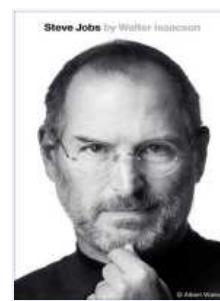
8 views

1 0

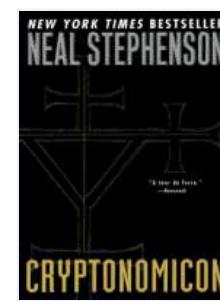
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S



```
--1.
IF OBJECT_ID('REPORTE_PASAJEROS') IS NOT NULL
BEGIN
    DROP PROCEDURE DBO.REPORTE_PASAJEROS
END
GO

--2.
CREATE PROCEDURE REPORTE_PASAJEROS (@PAIS VARCHAR(30))
AS
BEGIN
    DECLARE @IDPA CHAR(5),@NOM CHAR(30),@PAI CHAR(30),
            @TEL CHAR(15),@EMA VARCHAR(40),@TOTAL INT=0

    DECLARE MICURSOR CURSOR
    FOR SELECT PAS.IDPASAJERO,PAS.NOMBRES,PAI.PAIS,
              PAS.TELEFONO,PAS.EMAIL
              FROM PASAJERO PAS
              JOIN PAIS PAI ON PAS.IDPAIS=PAI.IDPAIS
              WHERE PAI.PAIS=@PAIS

    OPEN MICURSOR

    FETCH MICURSOR INTO @IDPA,@NOM,@PAI,@TEL,@EMA
    PRINT 'CODIGO      PASAJERO      PAIS      TELEFONO      EMAIL'
    PRINT '-----'
    WHILE @@FETCH_STATUS=0
    BEGIN
        SET @TOTAL+=1
        PRINT @IDPA+SPACE(5)+@NOM+SPACE(5)+RTRIM(@PAI)+
              SPACE(5)+RTRIM(@TEL)+SPACE(5)+@EMA
        FETCH MICURSOR INTO @IDPA,@NOM,@PAI,@TEL,@EMA
    END

    PRINT '-----'
    PRINT 'EL TOTAL DE PASAJEROS ES: >> '+CAST(@TOTAL AS CHAR(6))
    CLOSE MICURSOR
    DEALLOCATE MICURSOR
END
GO
```

En el punto se verifica la existencia del procedimiento, en el punto dos se implementa el pro

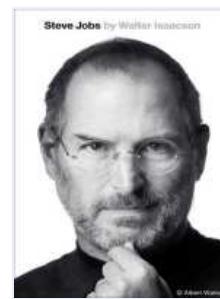
8 views

1 0

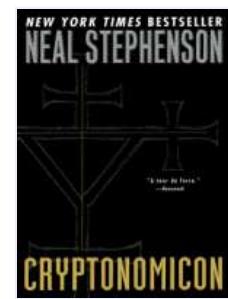
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

272 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

La imagen siguiente muestra el resultado de la ejecución:

CODIGO	PASAJERO	PAÍS	TELÉFONO	EMAIL
P0001	ANGELA TORRES LAZARO	PERU	999999999	ATORRES@HOTMAIL.COM
P0002	FERNANDA TORRES LAZARO	PERU	333333333	FTORRES@HOTMAIL.COM
P0003	LJUZ LAZARO MENOR	PERU	999999999	LLAZARO@GMAIL.COM

EL TOTAL DE PASAJEROS ES: >> 3

CASO DESARROLLADO N° 4.65

Implemente un procedimiento almacenado que permita reportar los registros de la tabla por medio de un CURSOR, considere que sólo se mostrarán los pasajeros de un determinado país ingresado por el usuario. Debe adicionar el nombre del país a la consulta.



```
--1.
IF OBJECT_ID('REPORTE_SERVIDOR') IS NOT NULL
BEGIN
    DROP PROCEDURE DBO.REPORTE_SERVIDOR
END
GO

--2.
CREATE PROCEDURE REPORTE_SERVIDOR(@SERVIDOR VARCHAR(20))
AS
BEGIN
    DECLARE @IDPA CHAR(5),@NOM CHAR(30),
            @EMA CHAR(40),@TOTAL INT=0

    DECLARE MICURSOR CURSOR
        FOR SELECT PAS.IDPASAJERO,PAS.NOMBRES,PAS.EMAIL
        FROM PASAJERO PAS
        WHERE PAS.EMAIL LIKE '%' +@SERVIDOR+'%'

    OPEN MICURSOR

    FETCH MICURSOR INTO @IDPA,@NOM,@EMA
    PRINT 'CODIGO      PASAJERO          EMAIL'
    PRINT '-----'
    WHILE @@FETCH_STATUS=0
    BEGIN
        SET @TOTAL+=1
        PRINT @IDPA+','+@NOM+','+@EMA
    END
END
```

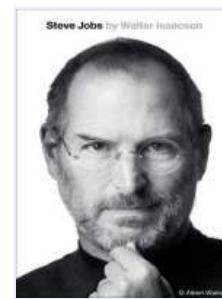
8 views

1 0

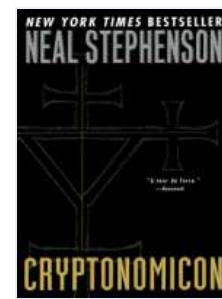
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

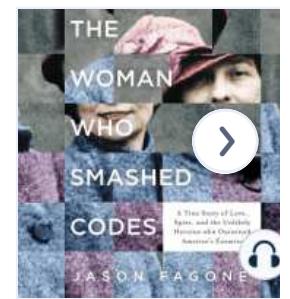
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

En el punto uno se verifica la existencia del procedimiento, en el punto dos se imprime el procedimiento esta vez el parámetro será el nombre de algún servidor como por ejemplo GMAIL. Para probar el procedimiento deberá ejecutar el siguiente script:

```
EXEC REPORTE_SERVIDOR 'GMAIL'
```

La imagen siguiente muestra el resultado de los pasajeros según el servidor ingresado es GMAIL.

CODIGO	PASAJERO	EMAIL
P0003	MARIA ZAMORA MEJIA	MZAMORA@GMAIL.COM
P0005	LUZ LAZARO MENOR	LLAZARO@GMAIL.COM
P0007	NERY CALLE DE LA CRUZ	NCALLE@GMAIL.COM
P0009	MARISOL DIAZ ZAMBRANO	MDIAZ@GMAIL.COM
P0010	GUADALUPE ACOSTA FERRER	GACOSTA@GMAIL.COM
EL TOTAL DE PASAJEROS CON SERVIDOR GMAIL ES: 5		

4.28. TRANSACCIONES EN TRANSACT SQL

Una transacción es un conjunto de órdenes que se ejecutan formando una unidad de trabajo en forma indivisible o atómica.

Un SGBD se dice transaccional, si es capaz de mantener la integridad de los datos, haciendo que las transacciones no puedan finalizar en un estado intermedio. Cuando por alguna causa el sistema tiene que cancelar la transacción, empieza a deshacer las órdenes ejecutadas hasta dejar la base de datos en su estado inicial (llamado punto de integridad), como si la orden de la transacción nunca hubiera sido realizada.

Para esto, el motor de base de datos de SQL Server, provee los mecanismos para especificar qué conjunto de acciones deben constituir una transacción. Se cuenta específicamente con tres tipos de órdenes:

- **BEGIN TRAN:** especifica que va a empezar una transacción.
- **COMMIT TRAN:** le indica al motor de base de datos que puede considerar la transacción finalizada con éxito.
- **ROLLBACK TRAN:** indica que se ha alcanzado un fallo y que debe restablecer la base de datos en su punto de integridad.

BEGIN TRANSACTION representa un punto en el que los datos a los que hace referencia la conexión son lógica y físicamente coherentes. Si se producen errores, se pueden revertir las modificaciones realizadas en los datos después de **BEGIN TRANSACTION** para devolverlos al punto conocido de coherencia. Cada transacción dura hasta que se completa sin errores.

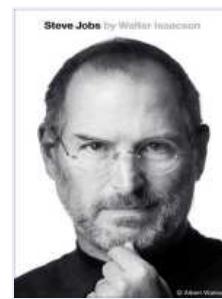
8 views

1 0

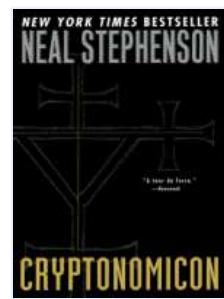
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

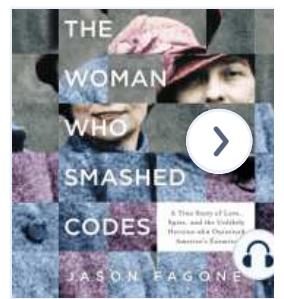
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

274

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Sintaxis:

IMPLEMENTACION DE TRANSACCIONES
BEGIN TRANSACTION

```
BEGIN [ TRAN | TRANSACTION ] NOMBRE_TRANSACCION
COMMIT TRAN NOMBRE_TRANSACCION
ROLLBACK TRAN NOMRE_TRANSACCION
```

CASO DESARROLLADO N° 4.66

Implemente un procedimiento almacenado que permita registrar un pasajero contra inserción por medio de una transacción, emita un mensaje en cada ocasión, es decir, si todo emitir PASAJERO REGISTRADO CON EXITO, caso contrario OCURRIO UN ERROR AL INSERTA



```
--1.
IF OBJECT_ID('NUEVOPASAJERO') IS NOT NULL
BEGIN
    DROP PROCEDURE DBO.NUEVOPASAJERO
END
GO

--2.
CREATE PROCEDURE NUEVOPASAJERO(
    @IDPAS CHAR(5),@NOM VARCHAR(40),@PAI VARCHAR(40),
    @TEL VARCHAR(15),@EMA VARCHAR(40)
)
AS
BEGIN TRANSACTION TPASAJERO
DECLARE @IDPAI CHAR(4)
SELECT @IDPAI = IDPAIS FROM PAIS WHERE PAIS=@PAI
INSERT INTO PASAJERO
VALUES (@IDPAS,@NOM,@IDPAI,@TEL,@EMA)

IF @@ERROR=0
BEGIN
    PRINT 'PASAJERO REGISTRADO CON EXITO'
    COMMIT TRAN TPASAJERO
END
ELSE
BEGIN
    PRINT 'OCURRIO UN ERROR AL INSERTAR'
    ROLLBACK TRAN TPASAJERO
END
```

8 views

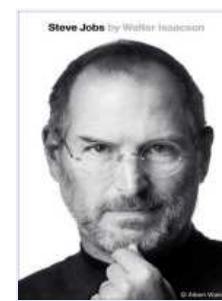
1 like

0 comments

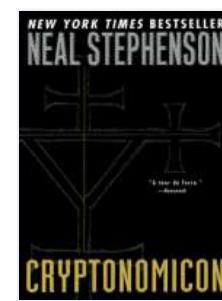
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

Para poder encontrar si hubo algún error se tuvo que invocar al variable global @@ERROR el error y lo muestra por medio de sus funciones pero en este caso comprobamos si no hay con IF @@ERROR=0 entonces se mostrará el mensaje de confirmación exitosa y allí se ejecuta la sentencia COMMIT TRAN asegurando el proceso confirmado.

El mismo caso sucede si encuentra un error mostrará un mensaje y aplicará la sentencia TRAN esto anulará todo el proceso, es decir, desde donde se inició la transacción.

Para ejecutar el procedimiento se necesita el siguiente script:

```
EXEC NUEVOPASAJERO 'P0015','MERY CALLE','PERU',
'988888888','MCALLE@GMAIL.COM'
GO
```

La imagen siguiente muestra el resultado de la ejecución (primera vez):

```
(1 filas afectadas)
PASAJERO REGISTRADO CON EXITO
```

La siguiente imagen se muestra sólo si ejecuta nuevamente el script anterior, ocurre un error porque el IDPASAJERO registrado se intenta registrar nuevamente:

```
Mens 2627, Nivel 14, Estado 1, Procedimiento NUEVOPASAJERO, Línea 10
Infracción de la restricción PRIMARY KEY 'PK_PASAJERO_165590CB2CCE124'.
No se pudo insertar una clave duplicada en el objeto 'dbo.PASAJERO'.
Se terminó la instrucción.
OCURRIÓ UN ERROR AL INSERTAR
```

La siguiente imagen muestra el error si el nombre del país ingresado no se encuentra registrada en la tabla de origen.

```
Mens 515, Nivel 16, Estado 2, Procedimiento NUEVOPASAJERO, Línea 10
No se pudo insertar el valor NULL en la columna 'IDPAI', tabla 'AGENCIA.dbo.PASAJERO'.
La columna no admite valores NULL. Error de INSERT.
Se terminó la instrucción.
OCURRIÓ UN ERROR AL INSERTAR
```

Ahora implementaremos el mismo caso, pero usando el control de error BEGIN TRY...BEGIN CATCH...END CATCH así evitar la comparación de la variable @@ERROR.

```
CREATE PROCEDURE NUEVOPASAJERO(
    @IDPAS CHAR(5),@NOM VARCHAR(40),@PAI VARCHAR(40),
    @TEL VARCHAR(15),@EMA VARCHAR(40)
)
AS
BEGIN TRANSACTION TPASAJERO
BEGIN TRY
    DECLARE @IDPAI CHAR(4)
```

8 views

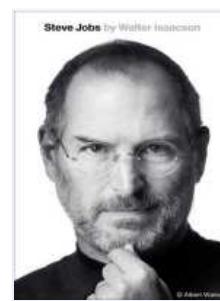
1 like

0 comments

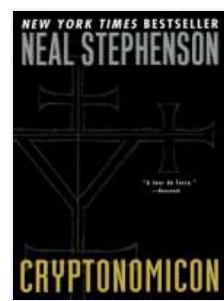
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

276 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En el script sólo se muestra una versión similar al script anterior, la diferencia es que contiene errores con **BEGIN TRY** ya no dependemos de la variable global **@@ERROR**.

El mensaje de error es mejor controlado desde BEGIN TRY y BEGIN CATCH como lo muestra la imagen:

(0 filas afectadas)
OCURRIO UN ERROR AL INSERTAR

CASO DESARROLLADO N° 4.67

Implemente un procedimiento almacenado que permita registrar el pago y a la vez debe actualizar los valores de la tabla reserva involucradas en los pagos, emitiendo mensajes para cada caso.



```

--1.
IF OBJECT_ID('REGISTRAPAGO') IS NOT NULL
BEGIN
    DROP PROCEDURE REGISTRAPAGO
END
GO

--2
CREATE PROC REGISTRAPAGO(
    @NUM INT,
    @IDRES INT,
    @TPAS CHAR(5)
)
    
```

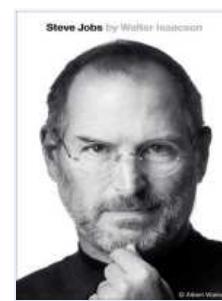
8 views

1 0

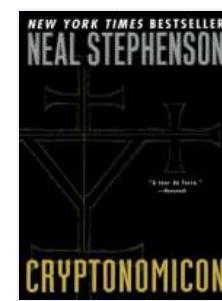
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Hidden Heroine Behind a Conspiracy

CAP. 4: PROGRAMACIÓN TRANSACT S

```
--5.
UPDATE RESERVA
    SET COSTO+=@MON
    WHERE IDRESERVA=@IDRES
    PRINT 'ACTUALIZACION DE RESERVA CORRECTA'

    COMMIT TRAN TPAGO
END TRY
BEGIN CATCH
    --6.
    PRINT 'OCURRIO UN ERROR AL REGISTRAR PAGO'
    PRINT @@ERROR
    ROLLBACK TRAN TPAGO
END CATCH
GO
```

En el punto uno se verifica si el procedimiento ya ha sido implementado, si esto sucede eliminamos.

En punto dos se implementa el procedimiento almacenado REGISTRAPAGO el cual tiene 5 de entrada y son proporcionales al número de columnas de la tabla PAGO.

```
CREATE PROC REGISTRAPAGO(
    @NUM INT,
    @IDRES INT,
    @IDPAS CHAR(5),
    @FEC DATE,
    @MON MONEY
)
```

En el punto tres se define el inicio de la transacción en este caso se llamó TPAGO y den controlará los errores ocasionados por el script con BEGIN TRY y BEGIN CATCH.

```
BEGIN TRAN TPAGO
BEGIN TRY
```

En el punto cuatro se aplica la sentencia INSERT INTO para registrar los valores ingresados parámetros y enviarlos a la tabla PAGO.

```
INSERT INTO PAGO VALUES (@NUM,@IDRES,@IDPAS,@FEC,@MON)
```

En el punto cinco se actualiza los valores de la tabla RF\$FRVA siempre y cuando el IDF

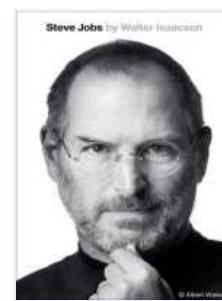
8 views

1 0

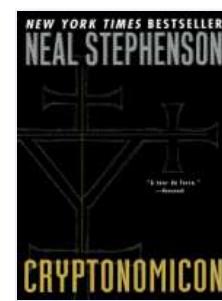
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

278 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

```
UPDATE RESERVA
    SET COSTO+=@MON
    WHERE IDRESERVA=@IDRES
    PRINT 'TRANSACCION CORRECTA'

    COMMIT TRAN TPAGO
```

En el punto seis se implementa el script que devolver en caso hubiera un error en la transacción. Para anular todo el proceso se tiene que colocar la instrucción **ROLLBACK TRAN TPAGO** como instrucción incluida dentro del bloque **BEGIN TRAN TPAGO** queda anulada.

```
PRINT 'OCURRIO UN ERROR AL REGISTRAR PAGO'
PRINT @@ERROR
ROLLBACK TRAN TPAGO
```

Para comprobar el procedimiento almacenado, primero veamos cual es el contenido de las tablas **PAGO** y **RESERVA**, para esto ejecute en conjunto el siguiente script:

```
SELECT * FROM PAGO WHERE IDRESERVA=1
SELECT * FROM RESERVA WHERE IDRESERVA=1
GO
```

En la imagen siguiente muestra el resultado de las consultas y notamos que en la tabla **PAGO** encuentra registrada la reserva con IDRESERVA 1 a costo 0 y lo reservó el día 01 Octubre de 2011.

NUMPAGO	IDRESERVA	IDPASAJERO	FECHA	MONTO
IDRESERVA	COSTO	FECHA		
1	1	0.00	2011-10-01	

Ejecutaremos el procedimiento almacenado pensando en el IDRESERVA 1 y así comprobaremos si la transacción es correcta o no, ejecute el siguiente script:

```
EXEC REGISTRAPAGO 1,1,'P0003','02/10/2011',400
GO
```

La imagen siguiente muestra el resultado de la ejecución y como notará aparecen dos consultas.

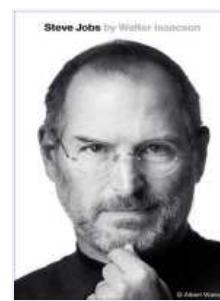
8 views

1 0

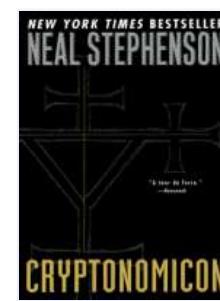
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Unlikely Heroine Who Outwitted America's Enemies

Ahora, veamos nuevamente como están los registros de ambas tablas.

NUMPAGO	IDRESERVA	IDPASAJERO	FECHA	MONTO
1	1	P0003	2011-10-02	400.00
IDRESERVA	COSTO	FECHA		
1	1	400.00	2011-10-01	

Como verá en la tabla PAGO se registró el primer pago realizado por el pasajero de código P0003 con un monto de 400.00 y a la vez observamos que la tabla RESERVA actualizó su costo a 400.00 en su primer pago.

Si ejecutáramos un segundo pago del mismo pasajero entonces tendríamos que ejecutar el siguiente script:

```
EXEC REGISTRAPAGO 2,1,'P0003','05/10/2011',800
GO
```

En esta simulación de registro se envió 800.00 a la tabla y; por lo tanto, la actualización en la tabla RESERVA debería ser 400+800 por ser una columna acumulativa es por eso que observe que la columna COSTO de la tabla RESERVA muestra 1200.00.

NUMPAGO	IDRESERVA	IDPASAJERO	FECHA	MONTO
1	1	P0003	2011-10-02	400.00
2	2	P0003	2011-10-05	800.00
IDRESERVA	COSTO	FECHA		
1	1	1200.00	2011-10-01	

4.29. TRIGGERS

Un trigger (o disparador) en una Base de datos, es un procedimiento que se ejecuta cuando se cumple una determinada condición al realizar una operación. Dependiendo de la base de datos, pueden ser de inserción (INSERT), actualización (UPDATE) o borrado (DELETE). Algunas bases de datos permiten ejecutar triggers al crear, borrar o editar usuarios, tablas, bases de datos u otros objetos.

Usos:

- Nos permite registrar, auditar y monitorear los procesos de cambio de valores a las tablas.

8 views

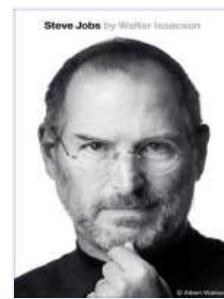
1 like

0 comments

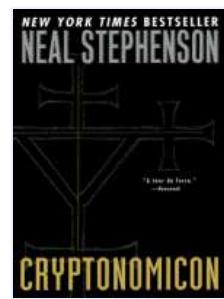
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

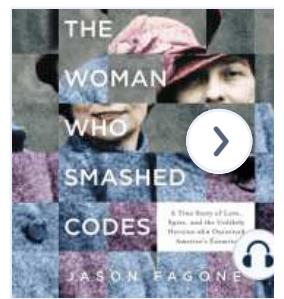
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

280 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Ventajas:

- Un trigger ofrece chequeos de seguridad en valores con respecto a los objetos tablas o de datos.
- Fuerzan restricciones dinámicas de integridad de datos y de integridad referencial.
- Aseguran que las operaciones relacionadas se realizan juntas en forma implícita.
- Ofrecen un mayor control sobre los objetos de una base de datos.

Desventajas:

- Se tiene que programar anticipadamente lo que debe realizar un trigger.
- Siendo un procedimiento no se puede invocar directamente.
- Los triggers siempre serán creados para un conjunto de registros y no para uno solo ya por operación DML.
- Los triggers no son aplicables en tablas temporales.

Hay dos tipos de triggers:

- Triggers DML
- Triggers DDL

Sintaxis para triggers DML:

IMPLEMENTACION DE TRIGGER CREATE TRIGGER

```
CREATE TRIGGER NOMBRE_TRIGGER
ON [ TABLE | VIEW ]
FOR | AFTER | INSTEAD OF
[ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ]
AS
SENTENCIA_SQL
```

Sintaxis para triggers DDL:

IMPLEMENTACION DE TRIGGER CREATE TRIGGER

```
CREATE TRIGGER NOMBRE_TRIGGER
ON [ ALL SERVER | DATABASE ]
FOR | AFTER
AS
SENTENCIA_SQL
```

8 views

1 like

0 comments

eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)

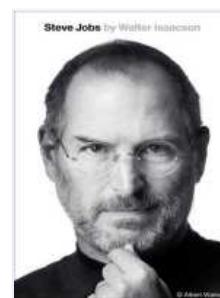
Save

Embed

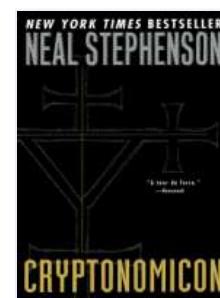
Share

Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Unlikely Heroine Who Outwitted America's Smartest Cryptographers

CAP. 4: PROGRAMACIÓN TRANSACT S

- **FOR | AFTER:** AFTER especifica que el desencadenador DML sólo se activa cuando las operaciones especificadas en la instrucción SQL desencadenadora se han ejecutado completamente. Además, todas las acciones referenciales en cascada y las comprobaciones de restricciones deben ser correctas para que este desencadenador se ejecute.

AFTER es el valor predeterminado cuando sólo se especifica la palabra clave trigger. Los desencadenadores AFTER no se pueden definir en las vistas.

- **INSTEAD OF:** especifica que se ejecuta el desencadenador DML en vez de la instrucción desencadenadora, por lo que se suplantan las acciones de las instrucciones desencadenadas. Sin embargo, INSTEAD OF no se puede especificar para los desencadenadores DDL.

Como máximo, se puede definir un desencadenador INSTEAD OF por cada instrucción INSERT, UPDATE o DELETE en cada tabla o vista. No obstante, en las vistas es posible definir otras que no tengan su propio desencadenador INSTEAD OF.

Opciones del Trigger

- Mostrar todos los triggers de la base de datos

```
SELECT * FROM SYS.TRIGGERS
GO
```

- Eliminar un trigger

```
IF OBJECT_ID('MENSAJE_PASAJERO') IS NOT NULL
BEGIN
    DROP TRIGGER MENSAJE_PASAJERO
    PRINT 'TRIGGER ELIMINADO CORRECTAMENTE'
END
ELSE
    PRINT 'TRIGGER NO EXISTE'
GO
```

- Inhabilitar un trigger: la inhabilitación indica que la tabla podrá ejecutar sus sentencias sin el mayor control del trigger inhabilitado.

```
DISABLE TRIGGER MENSAJEXNUEVOPASAJERO
ON PASAJERO
GO
```

- Habilitar un trigger:

```
ENABLE TRIGGER MENSAJEXNUEVOPASAJERO
```

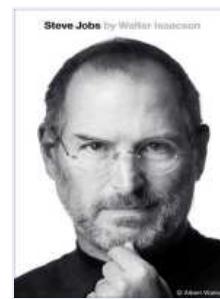
8 views

1 0

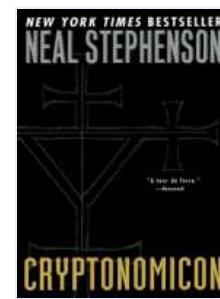
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

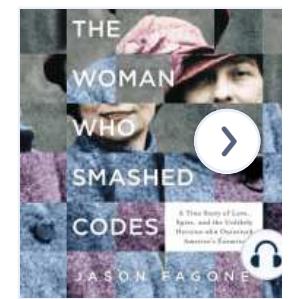
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

282 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

4.30. CASOS DESARROLLADOS PARA TRIGGERS DML

CASO DESARROLLADO N° 4.68

Implemente un trigger que permita mostrar un mensaje cada vez que se inserte o actualice en la tabla PASAJERO.

PASAJERO	
IDPASAJERO	
NOMBRES	
IDPAÍS	
TELEFONO	
EMAIL	

```
CREATE TRIGGER MENSAJE_PASAJERO
ON PASAJERO
FOR INSERT, UPDATE
AS
PRINT 'MENSAJE DISPARADO POR LA INSERCIÓN O
ACTUALIZACIÓN DE LA TABLA PASAJERO'
GO
```

En el script se implementa un trigger que por insertar un nuevo registro (INSERT INTO) o (UPDATE) se debe mostrar el mensaje enviado desde la función PRINT.

Para probar el trigger se tiene que registrar un nuevo pasajero colocando el siguiente script:

```
INSERT INTO PASAJERO
VALUES('P0012', 'KATHY CACSIRE', '0001',
'988888889', 'KCACSIRE@HOTMAIL.COM')
GO
```

La imagen siguiente muestra el resultado de la inserción del nuevo pasajero:

```
MENSAJE DISPARADO POR LA INSERCIÓN O
ACTUALIZACIÓN DE LA TABLA PASAJERO
(1 filas afectadas)
```

Observe que desde el trigger se envía el mensaje y desde el motor de base de datos se ejecutan las sentencias (1 fila afectada).

CASO DESARROLLADO N° 4.69

Implemente un trigger que permita enviar un correo electrónico al administrador de base de datos.

8 views

1 like

0 comments

eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)

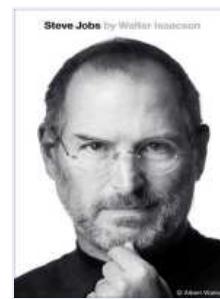
Save

Embed

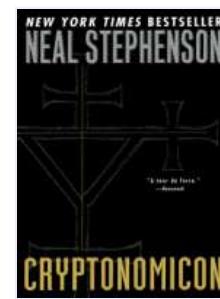
Share

Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

4.31. CASOS DESARROLLADOS PARA TRIGGERS DDL

CASO DESARROLLADO N° 4.75

Implemente un trigger que permita bloquear la aplicación de las instrucciones ALTER y DROP en la base de datos AGENCIA.

```
IF EXISTS (SELECT * FROM SYS.TRIGGERS
           WHERE PARENT_CLASS = 0 AND NAME = 'POLITICA')
    DROP TRIGGER POLITICA ON DATABASE
GO

CREATE TRIGGER POLITICA
ON DATABASE
FOR DROP_TABLE, ALTER_TABLE
AS
    ROLLBACK TRANSACTION
    PRINT 'SI DESEA ELIMINAR O ALTERAR SUS TABLAS'
    PRINT 'PRIMERO DEBE INHABILITAR EL TRIGGER POLITICA'
GO
```

Para probar el script se tiene que eliminar una tabla que no tenga referencia a otra tabla, e no tenga llaves foráneas asociadas, entonces ejecute el siguiente script:

```
DROP TABLE PASAJEROSBAK
GO
```

La imagen siguiente muestra el resultado al tratar de eliminar la tabla PASAJEROBAK, e implementó en un caso anterior y tenía por misión almacenar la réplica de los pasajeros pero no tenía asociación a ningún objeto de tipo tabla.

```
SI DESEA ELIMINAR O ALTERAR SUS TABLAS
PRIMERO DEBE INHABILITAR EL TRIGGER POLITICA
Meme. 2609, Nivel 15, Estado 2, Línea 1
La transacción terminó en el desencadenador. Se anuló el lote.
```

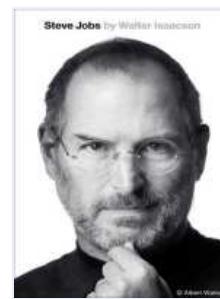
8 views

1 0

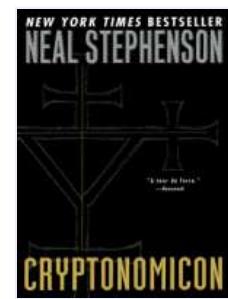
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACTS

```

CREATE TRIGGER MENSAJEXNUEVOPASAJERO
ON PASAJERO
FOR INSERT
AS

DECLARE @MENSAJE VARCHAR(200)
DECLARE @IDPAS CHAR(5)

SELECT @IDPAS = (SELECT IDPASAJERO FROM INSERTED)
SELECT @MENSAJE = 'NUEVO PASAJERO REGISTRADO '+@IDPAS

EXEC MASTER.DBO.XP_SENDFILE
    @RECIPIENTS = 'WEBMASTER@SERVIDOR.COM.PE',
    @SUBJECT = '** NUEVO USUARIO **',
    @MESSAGE = @MENSAJE
GO

```

CASO DESARROLLADO N° 4.70

Implemente un trigger que permita crear un histórico de los registros realizados a la tabla PAGO. Cada vez que se realice un pago por parte de un pasajero se deberá enviar un correo electrónico informando el IDPASAJERO y el total de pagos realizados por dicho pasajero a una nueva llamada CUENTAPAGOXPASAJERO.



```

--1.
CREATE TABLE CUENTAPAGOXPASAJERO(
    IDPASAJERO      CHAR(5) NOT NULL,
    TOTAL           INT
)
GO

--2.
IF OBJECT_ID('TX_CUENTAPAGOXPASAJERO') IS NOT NULL
BEGIN
    DROP TRIGGER TX_CUENTAPAGOXPASAJERO
    PRINT 'TRIGGER ELIMINADO CORRECTAMENTE'
END
ELSE
    PRINT 'TRIGGER NO EXISTE'
GO

```

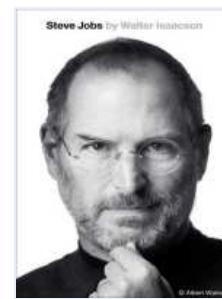
8 views

1 0

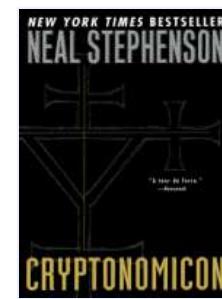
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Hidden Heroine Behind a Conspiracy

285

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

```

SELECT @TOTAL=COUNT(*)
FROM INSERTED,PAGO
WHERE PAGO.IDPASAJERO=INSERTED.IDPASAJERO

SELECT @PAS=INSERTED.IDPASAJERO FROM INSERTED

IF EXISTS(SELECT * FROM CUENTAPAGOXPASAJERO
WHERE IDPASAJERO=@PAS)
    UPDATE CUENTAPAGOXPASAJERO
    SET TOTAL=@TOTAL
    WHERE IDPASAJERO=@PAS
ELSE
    INSERT INTO CUENTAPAGOXPASAJERO VALUES(@PAS,@TOTAL)
GO

```

En el punto uno se implementa el script de creación de la tabla histórica llamada CUENTAPAGC hay que tener en cuenta que sólo tendrá dos columnas en una se encontrará el IDPASAJER y en otra columna el total de pagos realizados por el mismo pasajero.

En el punto dos se implementa el script que determinará si el trigger existe en la base eliminarlo enviando mensajes de confirmación en ambos casos.

En el punto tres se implementa el script que permite crear el trigger TX_CUENTAPAGOXP/ debe especificar que las acciones se realizan dentro de la tabla PAGO (ON PAGO). Toda del trigger deberán ocurrir sólo después de insertado un registrado a la tabla PAGO por t especificación AFTER INSERT.

Se declaran dos variables locales, @TOTAL tiene por misión obtener el número total de pag por un determinado pasajero y @PAS tiene por misión capturar el IDPASAJERO proveniente realizado a la tabla PAGO. Por medio de la sentencia SELECT se registrarán los valores nec la tabla CUENTAPAGOXPASAJERO.

```

DECLARE @TOTAL INT,@PAS VARCHAR(40)
SELECT @TOTAL=COUNT(*)
FROM INSERTED,PAGO
WHERE PAGO.IDPASAJERO=INSERTED.IDPASAJERO

```

No se olvide que la tabla INSERTED proviene de los datos que se intentan registrar a la tak hizo una comparación entre IDPASAJERO de la tabla INSERTED y la tabla PAGO para pode conteo verídico del pasajero.

Luego se tiene que capturar el IDPASAJERO obtenido desde la tabla INSERTED es importante ya que es la única forma de comparar si hay duplicidad en la nueva tabla histó

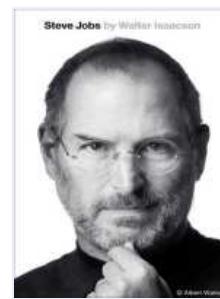
8 views

1 0

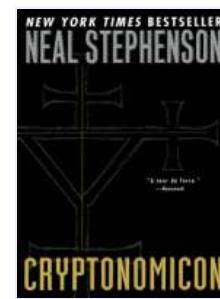
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT SQL

Finalmente, comparamos la existencia IDPASAJERO dentro de la nueva tabla CUENTAPAGOCXPASAJERO de acuerdo a la condición se actualizará el valor o se insertará dependiendo del código de

```

IF EXISTS(SELECT * FROM CUENTAPAGOXPASAJERO
WHERE IDPASAJERO=@PAS)
    UPDATE CUENTAPAGOXPASAJERO
    SET TOTAL=@TOTAL
    WHERE IDPASAJERO=@PAS
ELSE
    INSERT INTO CUENTAPAGOXPASAJERO VALUES(@PAS,@TOTAL)

```

Para probar el trigger debemos insertar registros en la tabla PAGO, para este caso se está insertando 5 registros; ejecute el siguiente script:

```

INSERT INTO PAGO VALUES(4,2,'P0002','08/10/2011',200)
GO
INSERT INTO PAGO VALUES(5,2,'P0002','10/10/2011',300)
GO
INSERT INTO PAGO VALUES(6,6,'P0009','18/12/2011',1000)
GO
INSERT INTO PAGO VALUES(7,6,'P0009','19/12/2011',700)
GO
INSERT INTO PAGO VALUES(8,6,'P0009','20/12/2011',200)
GO

```

Ahora, comprobamos los registros de la tabla PAGO, inicialmente se mostraba de la siguiente manera:

NUMPAGO	IDRESERVA	IDPASAJERO	FECHA	MONTO
1	1	P0003	2011-10-02	400.00
2	2	P0003	2011-10-05	800.00
3	3	P0003	2011-10-09	100.00

Luego de insertar los cinco registros se ve de la siguiente manera:

NUMPAGO	IDRESERVA	IDPASAJERO	FECHA	MONTO
1	1	P0003	2011-10-02	400.00
2	2	P0003	2011-10-05	800.00
3	3	P0003	2011-10-09	100.00
4	2	P0002	2011-10-08	200.00
5	5	P0002	2011-10-10	300.00
6	6	P0009	2011-12-10	1000.00
7	7	P0009	2011-12-19	700.00
8	8	P0009	2011-12-20	200.00

8 views

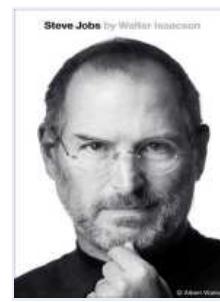
1 like

0 comments

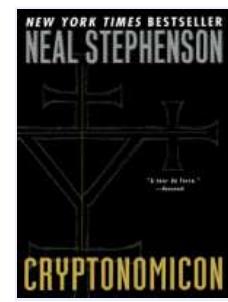
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

287 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 4.71

Implemente un trigger que permita controlar los registros de la tabla PAIS si el nombre de país registrado entonces deberá mostrar con que código fue registrado, caso contrario emitir PAIS REGISTRADO CORRECTAMENTE.

PAIS	
Column Name	Condensed Type
IDPAIS	char(4)
PAIS	varchar(30)

```
--1.
IF OBJECT_ID('VALIDAPAIS') IS NOT NULL
BEGIN
    DROP TRIGGER VALIDAPAIS
END
GO

--2.
CREATE TRIGGER VALIDAPAIS
ON PAIS
FOR INSERT
AS
    IF (SELECT COUNT(*) FROM INSERTED,PAIS
        WHERE INSERTED.PAIS=PAIS.PAIS)>1
    BEGIN
        DECLARE @PAI VARCHAR(30),@IDPA CHAR(5)
        SELECT @PAI=PAIS FROM INSERTED
        ROLLBACK
        SELECT @IDPA=IDPAIS FROM PAIS
        WHERE PAIS.PAIS=@PAI
        PRINT 'NOMBRE DE PAIS YA REGISTRADO EN LA TABLA'
        PRINT ''
        PRINT 'EL PAIS '+@PAI+
        ' SE ENCUENTRA REGISTRADO CON EL CODIGO: '+@IDPA
    END
    ELSE
        PRINT 'PAIS REGISTRADO CORRECTAMENTE'
    GO
```

En el punto uno se implementa un script que permite eliminar el trigger si ya se encontró la base de datos.

8 views

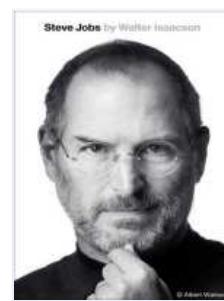
1 like

0 comments

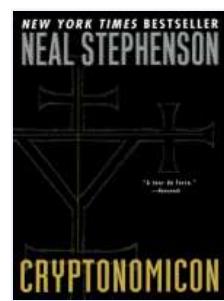
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

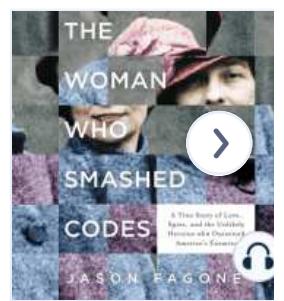
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

288 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

En este caso probaremos con un país nuevo colocando el siguiente script:

```
INSERT INTO PAIS VALUES('0015','HUATEMALA')
GO
```

La imagen siguiente muestra el resultado de la inserción del país Guatemala que no se encuentra en la tabla PAIS:

PAÍS REGISTRADO CORRECTAMENTE
(1 filas afectadas)

CASO DESARROLLADO N° 4.72

Implemente un trigger que permita controlar la eliminación de un registro de la tabla Pasajero. Si dicho pasajero tiene un pago registrado no permita su eliminación mostrando un mensaje contrario mostrando el mensaje de eliminación correcta.

PASAJERO	
	IDPASAJERO
	NOMBRES
	IDPAIS
	TELEFONO
	EMAIL

PAGO	
	NIIMPAGO
	IDRESERVA
	IDPASAJERO
	FECHA
	MONTO

```
--1.
IF OBJECT_ID('ELIMINAPASAJERO') IS NOT NULL
BEGIN
    DROP TRIGGER ELIMINAPASAJERO
END
GO

--2.
CREATE TRIGGER ELIMINAPASAJERO
ON PASAJERO
INSTEAD OF DELETE
AS
    IF EXISTS(SELECT * FROM PAGO
              WHERE PAGO.IDPASAJERO=(SELECT DELETED.IDPASAJERO
                                         FROM DELETED))
        BEGIN
            ROLLBACK TRANSACTION
            PRINT 'EL PASAJERO TIENE PAGOS REGISTRADOS, NO PUEDE ELIMINARSE'
        END
```

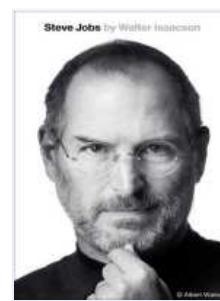
8 views

1 0

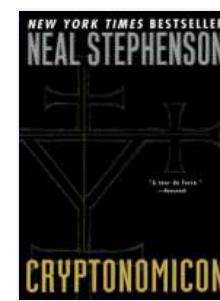
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Hidden Heroine Behind the Inventor of Public Key Cryptography

Luego se condiciona que el país que se intenta registrar no se encuentre en la tabla, en este caso se usó la función COUNT para determinar el número de veces que dicho país se cuenta registrando en la tabla, si el resultado es mayor a uno quiere decir que es la segunda vez que intenta registrarse y esto sucede porque el registro del nuevo país se graba primero en la tabla INSERTED.

```
IF (SELECT COUNT(*) FROM INSERTED,PAIS
    WHERE INSERTED.PAIS=PAIS.PAIS)>1
```

Luego declaramos dos variables , la primera @PAI tiene por misión obtener el nombre del país que se intenta registrar, lo obtendremos desde la tabla INSERTED, la segunda @IDPA tiene por misión obtener el código del país desde la tabla PAIS, es decir, buscarlo dentro de la tabla país y mostrar el resultado si se encuentra registrado. Aquí hay algo importante que mencionar, dicho código de país es un dato sencillo de buscar ya que existen dos códigos similares que son INSERTED.IDPAIS y PAIS.IDPAIS, por lo tanto para no generar ambigüedades es que se invoca a la instrucción ROLLBACK que permite anular la intervención de la tabla INSERTED quedando sólo la tabla PAÍS para la búsqueda.

```
DECLARE @PAI VARCHAR(30),@IDPA CHAR(5)
SELECT @PAI=PAIS FROM INSERTED
```

```
ROLLBACK
```

Una vez anulada a la tabla INSERTED se puede obtener el valor del IDPAIS desde la tabla PAIS.

```
SELECT @IDPA=IDPAIS FROM PAIS
WHERE PAIS.PAIS=@PAI
```

Finalmente, se imprimen los resultados obtenidos, para este caso se imprime el mensaje se muestra el nombre del país y su código encontrado.

```
PRINT 'NOMBRE DE PAIS YA REGISTRADO EN LA TABLA'
PRINT ''
PRINT 'EL PAIS '+@PAI+
' SE ENCUENTRA REGISTRADO CON EL CODIGO: '+@IDPA
```

Para probar el trigger necesita ejecutar el siguiente script:

```
INSERT INTO PAIS VALUES('0016','CHILE')
GO
```

8 views

1 like

0 comments

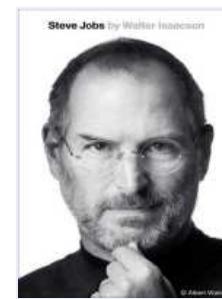
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

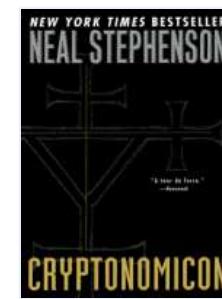
[Full description](#)

Save Embed Share Print

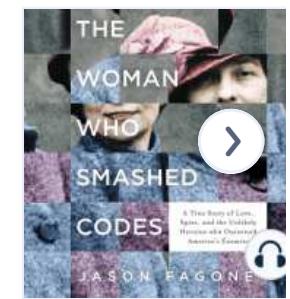
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

En el punto dos se implementa el trigger ELIMINAPASAJERO direccionando a la tabla PASAJE involucrada en el caso. Luego se condiciona que el trigger sea el responsable de la eliminación solicitado, ya que la verificación del IDPASAJERO se tiene que realizar en la tabla PAGO porque menciona que se eliminará solo si no tiene pagos realizados, entonces para este caso se usa DELETE eso quiere decir que no se ejecutará la sentencia DELETE hasta que termine de validar

```
CREATE TRIGGER ELIMINAPASAJERO
ON PASAJERO
INSTEAD OF DELETE
```

Luego verificamos la existencia del pasajero en la tabla PAGO todo esto será realizado por el operador EXISTS.

```
IF EXISTS(SELECT * FROM PAGO
          WHERE PAGO.IDPASAJERO=(SELECT DELETED.IDPASAJERO
          FROM DELETED))
```

Finalmente, se anula el proceso e imprime el mensaje solicitado para el caso:

```
ROLLBACK TRANSACTION
PRINT 'EL PASAJERO TIENE PAGOS REGISTRADOS, NO PUEDE ELIMINARSE'
```

Para poder probar la eliminación, ejecutaremos el siguiente script:

```
DELETE PASAJERO WHERE IDPASAJERO='P0003'
GO
```

La imagen siguiente muestra el resultado de la eliminación:

```
EL PASAJERO TIENE PAGOS REGISTRADOS, NO PUEDE ELIMINARSE
Mesa 3609, Nivel 16, Estado 1, Línea 1
La transacción terminó en el desencadenador. Se anuló el lote.
```

El pasajero de código P0003 si tiene pagos realizados; por lo tanto, no puede ser eliminado.

CASO DESARROLLADO N° 4.73

Implemente un trigger que permita controlar el registro de un pago en la cual se evalúe el monto registrado para que no se registre un valor inferior a cero en la columna Monto.

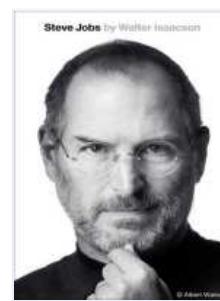
8 views

1 0

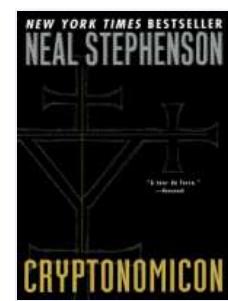
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

291

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

```
--1.
IF OBJECT_ID('VALIDAPAGO') IS NOT NULL
BEGIN
    DROP TRIGGER VALIDAPAGO
END
GO

--2.
CREATE TRIGGER VALIDAPAGO
ON PAGO
FOR INSERT
AS
    IF (SELECT MONTO FROM INSERTED)<=0
    BEGIN
        ROLLBACK TRANSACTION
        PRINT 'NO PUEDE REGISTRAR MONTO CERO'
    END
    ELSE
        PRINT 'PAGO REGISTRADO CORRECTAMENTE'
GO
```

En el punto uno se implementa el script que permitirá determinar si el trigger existe o no de datos.

En el punto dos se implementa el trigger VALIDAPAGO donde se involucra a la tabla PAGO allí donde se registrarán los pagos de los pasajeros. En el trigger se tiene que condicionar que no sea cero ya que no tiene sentido que un pasajero cancele un pago con monto cero, aquí mensaje de error y se anula la transacción.

```
IF (SELECT MONTO FROM INSERTED)<=0
BEGIN
    ROLLBACK TRANSACTION
    PRINT 'NO PUEDE REGISTRAR MONTO CERO'
END
```

CASO DESARROLLADO N° 4.74

Implemente un trigger que permita crear una replica de los registros insertados en la tabla Pasajero. Para dicho proceso debe implementar una nueva tabla llamada PasajeroBAK con las mismas columnas que la tabla Pasajero.



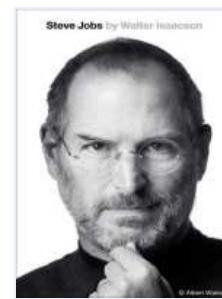
8 views

1 0

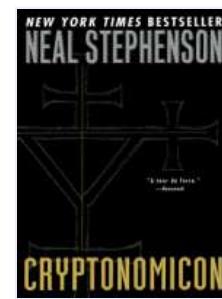
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

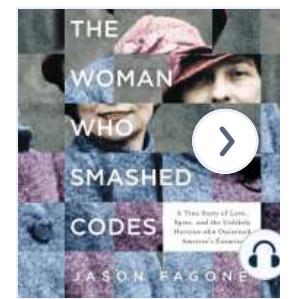
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Betrayal, and the Violent History of Mathematics

CAP. 4: PROGRAMACIÓN TRANSACT S

```
--1.
IF OBJECT_ID('PASAJEROSBAK') IS NOT NULL
BEGIN
    DROP TABLE PASAJEROSBAK
END
GO

--2.
CREATE TABLE PASAJEROSBAK(
    IDPASAJERO CHAR(5) NOT NULL PRIMARY KEY,
    NOMBRES VARCHAR(50) NOT NULL,
    IDPAIS CHAR(4) NOT NULL,
    TELEFONOCHAR(15) NOT NULL,
    EMAIL VARCHAR(50) NOT NULL
)
GO

--3.
IF OBJECT_ID('REPLICAPASAJERO') IS NOT NULL
BEGIN
    DROP TRIGGER REPLICAPASAJERO
END
GO

--4.
CREATE TRIGGER REPLICAPASAJERO
ON PASAJERO
AFTER INSERT
AS
BEGIN
    INSERT PASAJEROSBAK
        SELECT * FROM INSERTED
END
GO
```

En el punto uno se implementa el script que permitirá verificar que la tabla PASAJEROSBAK tenga en cuenta que este script sólo debe ejecutarse una vez ya que luego tendrán registr

En el punto dos se implementa el script que permitirá crear la tabla PASAJEROSBAK con columnas de la tabla PASAJERO.

En el punto tres se verifica que el trigger no exista en caso sea así se eliminará de la base de datos.

En el punto cuatro se implementa el trigger REPLICAPASAJERO que tiene por misión grabar los cambios realizados en la tabla PASAJERO en la tabla PASAJEROSBAK.

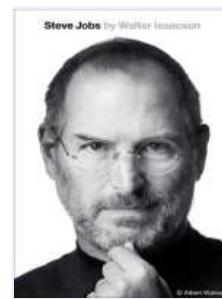
8 views

1 0

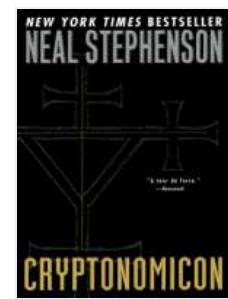
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

293 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Primero, veamos antes de la inserción como se encontraban los registros de las tablas inv

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4 P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5 P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
6 P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
7 P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
8 P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM

Ahora ejecutaremos el script que permite insertar un registro a la tabla PASAJERO.

```
INSERT INTO PASAJERO
VALUES('P0013', 'LUCERO ERAZO', '0004', '98888888', 'LERAZO@GMAIL.COM')
GO
```

(1 filas afectadas)

(1 filas afectadas)

Finalmente, comprobaremos que el registro se ha replicado correctamente y que en la tabla también se registró.

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1 P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2 P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4 P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5 P0013	LUCERO ERAZO	0004	98888888	LERAZO@GMAIL.COM
6 P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7 P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8 P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
9 P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM

8 views

1 0

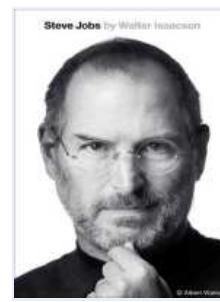
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

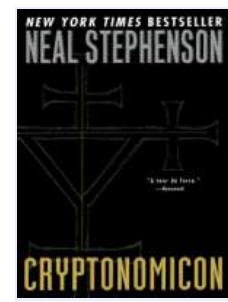
[Full description](#)

   
Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who
Smashed
Codes: A Tru

8 views

1 0

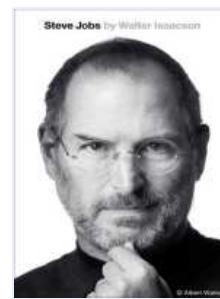
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

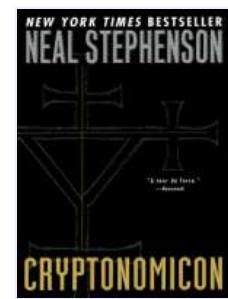
[Full description](#)

 Save  Embed  Share  Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics



XML con SQL

CAPACIDAD:

El lector podrá entender e implementar un script de operaciones con archivos XML, hacer uso de variables de tipo XML y usaremos archivos de tipo XML para la manipulación de registros tanto internos como externos de XML.

Al final del capítulo se tratarán imágenes dentro de una tabla y administrarla dentro de la aplicación SQL Image Viewer.

CONTENIDO:

- *Introducción*
- *Modelos de datos relacionales o XML*
- *Ventajas de almacenar valores en XML*
- *Elección de la tecnología XML*

8 views

1 0

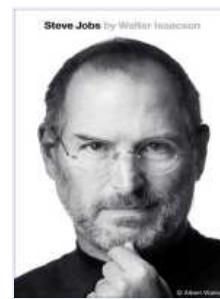
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

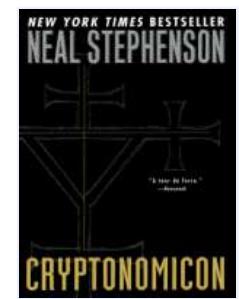
[Full description](#)

   
Save Embed Share Print

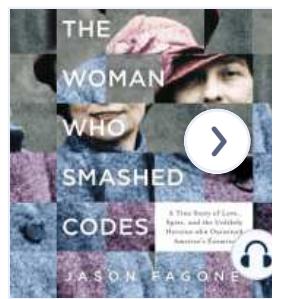
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

8 views

1 like

0 comments

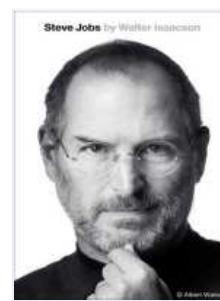
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

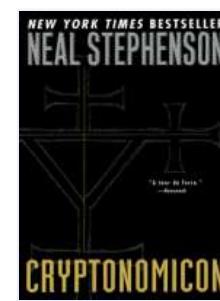
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

CAP. 5: XML CON SQL SERVER

5.1. INTRODUCCIÓN

En ocasiones anteriores al SQL Server 2005 para almacenar un valor de tipo XML se tenía que convertir el XML en texto plano o como tipo TEXT, desde esa versión hacia adelante se puede implementar el tipo de datos XML.

El tipo de datos XML permite almacenar documentos y fragmentos XML en una base de datos de forma nativa. Un fragmento XML es una instancia XML en la que falta un solo elemento de nivel.

El tipo de datos XML y los métodos asociados ayudan a integrar el XML en el marco relacional de SQL Server.

5.2. MODELO DE DATOS RELACIONALES o XML

Si los datos están muy estructurados con un esquema conocido, el modelo relacional es más apropiado. Sin embargo, si los datos tienen probabilidades de funcionar mejor para el almacenamiento de datos. SQL Server proporciona una gran cantidad de funcionalidad y las herramientas necesarias. Por otra parte, si los datos están semiestructurados y no tienen un esquema conocido, el modelo XML es más apropiado. Si los datos están estructurados, o no se conoce su estructura, debe contemplar la posibilidad de crear un esquema para los datos.

XML es una buena opción si desea un modelo independiente de la plataforma para garantizar la portabilidad de los datos mediante el uso de marcado estructural y semántico. Además, es una buena opción si se cumplen algunas de las siguientes propiedades:

- Los datos están dispersos o no se conoce la estructura de los mismos o la estructura de los datos puede cambiar de manera importante en el futuro.
- Los datos representan una jerarquía de inclusión, en lugar de referencias entre entidades que deben ser recursivos.
- El orden es inherente a los datos.
- Desea realizar consultas en los datos o actualizar parte de ellos, basándose en su estructura.

Si no se cumple ninguna de estas condiciones, debe utilizar el modelo de datos relacional. Sin embargo, si los datos tienen formato XML pero la aplicación sólo utiliza la base de datos para almacenarlos y recuperar los datos, sólo necesitará una columna [N]VARCHAR(MAX).

5.3. VENTAJAS DE ALMACENAR VALORES EN XML

Siempre presentará muchas ventajas mientras sepa qué hacer con los resultados que puede obtener de una consulta XML.

8 views

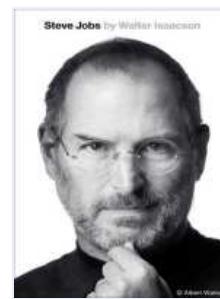
1 like

0 comments

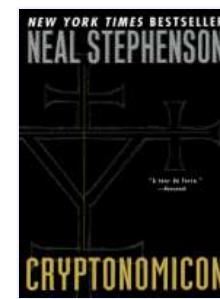
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

298 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

- Se puede extraer alguna sección de un documento XML o insertar una nueva sección si todo el documento.
- Se pueden operar con datos relacionales y datos XML dentro de una misma aplicación.
- En la actualidad el XML es totalmente compatible con la mayoría de lenguajes de programación como .NET, Java o PHP.
- Un XML tiene acceso a SOAP, ADO.NET y OLE DB.
- También se puede usar para realizar copias de seguridad, recuperaciones y réplicas de las tablas en una base de datos.

5.4. ELECCIÓN DE LA TECNOLOGÍA XML

La elección de la tecnología XML, XML nativo frente a vista XML, en general depende de los siguientes factores:

- **Opciones de almacenamiento:** los datos XML pueden ser más apropiados para el almacenamiento de objetos grandes (por ejemplo, el manual de un producto) o más sensibles al almacenamiento en columnas relacionales (por ejemplo, un elemento de línea convertido a XML). Cada almacenamiento preserva la fidelidad del documento en distinta medida.
- **Funciones de consultas:** es posible que una opción de almacenamiento le parezca más apropiada que otra en función de la naturaleza de las consultas y del nivel de detalle con que se desean manipular los datos XML. La consulta detallada de los datos XML (por ejemplo, evaluación de expresiones complejas en nodos XML) se admite en diversos grados en las dos opciones de almacenamiento.
- **Indizar datos XML:** tal vez deseé indizar los datos XML para acelerar el rendimiento de la aplicación. Las opciones de indización varían según las opciones de almacenamiento; es necesario elegir la elección apropiada para optimizar la carga de trabajo.
- **Funciones para la modificación de datos:** algunas cargas de trabajo implican una manipulación detallada de los datos XML. Este es el caso de agregar una sección nueva a un documento XML. Sin embargo, otras cargas de trabajo, como el contenido Web, no implican la modificación directa de los datos XML. La compatibilidad con el lenguaje de modificación de datos puede ser crucial para la aplicación.
- **Compatibilidad con esquemas:** los datos XML se pueden describir mediante un esquema XML. Un esquema XML define qué tipo de datos se admiten en un documento XML. La compatibilidad con XML enlazado a un esquema XML depende de la tecnología XML.

CASO DESARROLLADO N° 5.1

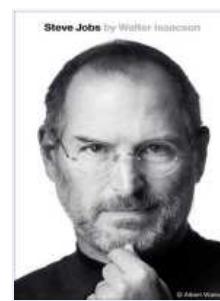
8 views

1 0

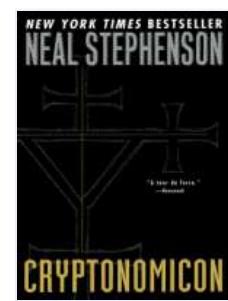
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 5: XML CON S

```
--1.
CREATE TABLE PASAJEROXML(
    NUMPASAJERO INT IDENTITY NOT NULL PRIMARY KEY,
    EMAIL        XML
)
GO

--2.
INSERT INTO PASAJEROXML
VALUES ('<email>vcastillo@gmail.com</email>')
GO
```

En el punto uno se muestra el script que permite crear la tabla PASAJEROXML definiendo la columna NUMPASAJERO que será autonumérico y llave de los registros y la columna tiene el tipo de datos XML.

La imagen siguiente muestra la consulta realizada la tabla PASAJEROXML:

	NUMPASAJERO	EMAIL
1	1	<email>vcastillo@gmail.com</email>

Qué pasaría en el caso que el pasajero no cuenta con sólo una cuenta de correo electrónico, necesite registrar por lo menos 2 emails por cada pasajero. Ejecutemos el siguiente script para 3 emails para dos nuevos pasajeros.

```
DECLARE @PXML AS XML
SET @PXML = '<email>mtorres@peru.com</email>
                <email>mtores@usa.net</email>
                <email>mtores@gmail.com</email>'

INSERT INTO PASAJEROXML(EMAIL) VALUES (@PXML)

SET @PXML = '<email>llazaro@hotmail.com</email>
                <email>llazaro@peru.net</email>
                <email>luz_lazaro_1982@hotmail.com</email>'

INSERT INTO PASAJEROXML(EMAIL) VALUES (@PXML)
GO
```

El resultado de visualizar los registros de la tabla PASAJEROXML lo muestra la siguiente imagen:

	NUMPASAJERO	EMAIL
1	1	<email>vcastillo@gmail.com</email>
2	2	<email>mtores@peru.com</email><email>mtores@usa..
3	3	<email>llazaro@hotmail.com</email><email>llazaro@p..

8 views

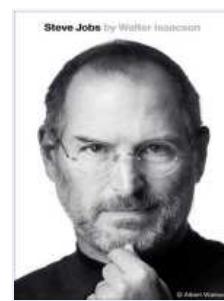
1 like

0 comments

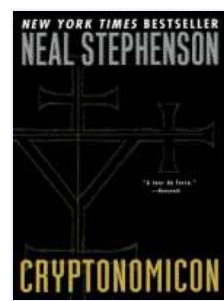
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Hidden Heroine Behind a Conspiracy

300 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 5.2

Script que permita crear una tabla llamada PASAJEROXML en donde se registren dos filas de tiempo, esta vez implementará dentro del XML la columna IDPASAJERO y EMAIL por los correos por cada pasajero.

PASAJEROXML	
Nombre de columna	Tipo comprimido
NUMPASAJERO	int
EMAIL	xml

```
--1.
IF OBJECT_ID('PASAJEROXML') IS NOT NULL
    DROP TABLE PASAJEROXML
GO

--2.
CREATE TABLE PASAJEROXML(
    NUMPASAJERO INT IDENTITY NOT NULL PRIMARY KEY,
    EMAIL           XML
)
GO

--3.
DECLARE @PXML AS XML
SET @PXML = '<pasajeros>
    <pasajero>
        <idpasajero>P0001</idpasajero>
        <email>mtorres@peru.com</email>
        <email>mtoress@usa.net</email>
        <email>mtoress@gmail</email>
    </pasajero>
    <pasajero>
        <idpasajero>P0002</idpasajero>
        <email>llazaro@gmail.com</email>
        <email>llazaro@peru.com</email>
        <email>luz_lazaro_1982@hotmail.com</email>
    </pasajero>
</pasajeros>'
INSERT INTO PASAJEROXML(EMAIL) VALUES (@PXML)
GO
```

En el punto uno se implementa el script que permite eliminar la tabla PASAJEROXML si base de datos.

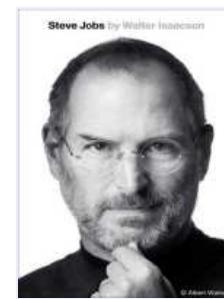
8 views

1 0

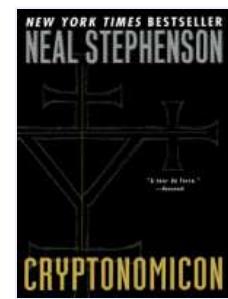
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



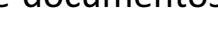
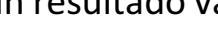
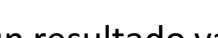
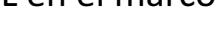
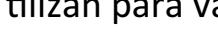
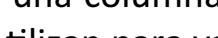
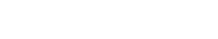
Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing



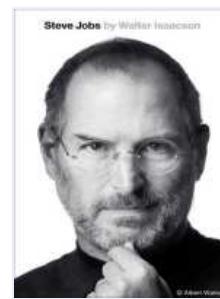
8 views

1 0

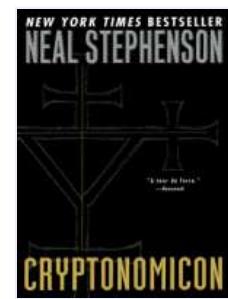
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

302 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Sintaxis:
**COLUMNAS Y
VARIABLES XML**

```
CREATE TABLE NOMBRE_TABLA(COLUMNA XML)
```

```
DECLARE @X XML
```

Tanto en la creación de una columna o una variable se especifica con XML ya que es un tipo del motor de base de datos.

CASO DESARROLLADO N° 5.3

Script que permite listar los registros de la tabla PASAJEROXML. Use el método QUERY.

PASAJEROXML	
Nombre de columna	Tipo comprimido
NUMPASAJERO	int
EMAIL	xml

Primero, debemos analizar la estructura XML con la que se compone un registro de la tabla PA

```
<pasajeros>
  <pasajero>
    <idpasajero>P0001</idpasajero>
    <email>MTORRES@PERU.COM</email>
    <email>MTORRES@USA.NET</email>
    <email>MTORRES@GMAIL</email>
  </pasajero>
  <pasajero>
    <idpasajero>P0002</idpasajero>
    <email>LLAZARO@GMAIL.COM</email>
    <email>LLAZARO@PERU.COM</email>
    <email>LUZ_LAZARO_1982@HOTMAIL.COM</email>
  </pasajero>
</pasajeros>
```

Como notará el esquema principal del XML está conformado por PASAJEROS como luego viene PASAJERO como un nodo de PASAJEROS, aquí si hay que tener muchísimo cuidado con la sensibilidad de las mayúsculas y minúsculas ya que por lo general todo script XML está case sensitive, nos referimos a la estructura XML no a los valores contenidos entre sus etiquetas.

```
SELECT NUMPASAJERO, EMAIL.query('pasajeros') AS [CORREOS REGISTRADOS]
FROM PASAJEROXML
```

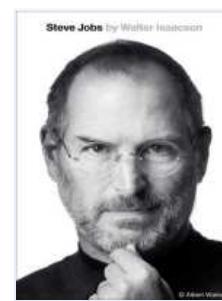
8 views

1 0

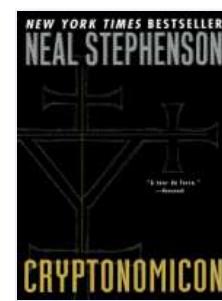
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



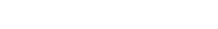
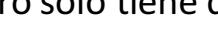
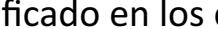
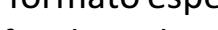
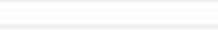
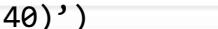
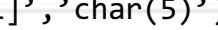
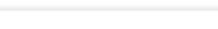
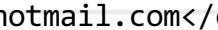
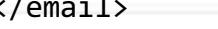
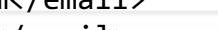
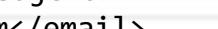
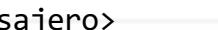
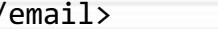
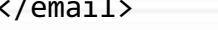
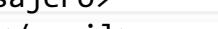
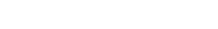
Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Unlikely Heroine Who Outwitted America's Smartest Codebreakers



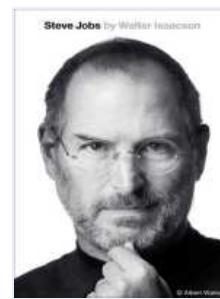
8 views

1 0

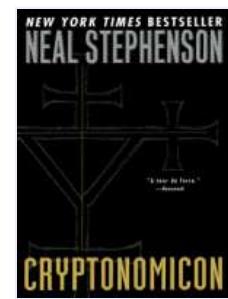
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

304

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012**CASO DESARROLLADO N° 5.5**

Script que permite mostrar el registro según un número de pasajeros ingresados por el usuario en la tabla PASAJEROXML. Use la función EXIST del tipo de datos XML.

PASAJEROXML	
Nombre de columna	Tipo comprimido
NUMPASAJERO	int
EMAIL	xml

```
DECLARE @N INT=1
SELECT P.NUMPASAJERO,P.EMAIL
FROM PASAJEROXML P
WHERE P.EMAIL.exist('/pasajeros/pasajero/email[3]')=@N
GO
```

El resultado de la ejecución se muestra en la siguiente imagen:

	NUMPASAJERO	EMAIL
1	1	<pasajeros><pasajero><idpasajero>P0001</idpasaje...

CASO DESARROLLADO N° 5.6

Script que permite insertar los registros de los pasajeros almacenados en un archivo XML en la tabla PASAJEROXML. Use BULK.

PASAJEROXML	
Nombre de columna	Tipo comprimido
NUMPASAJERO	int
EMAIL	xml

```
INSERT INTO PASAJEROXML
SELECT EMAIL
FROM (SELECT *
      FROM OPENROWSET(BULK 'C:\pasajeros.xml',
                      SINGLE_CLOB) AS PASAJEROS)
      AS P(EMAIL)
GO
```

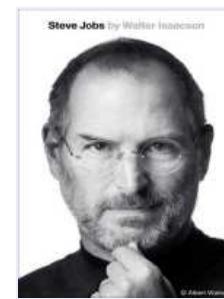
8 views

1 0

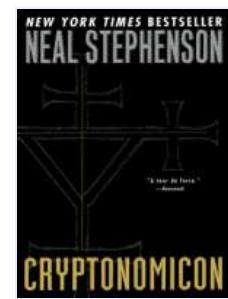
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

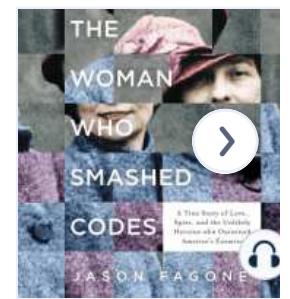
RELATED TITLES



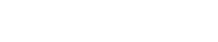
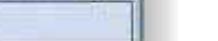
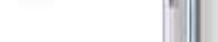
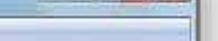
Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics



8 views

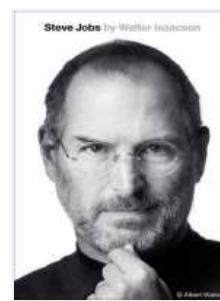
1 like

0 comments

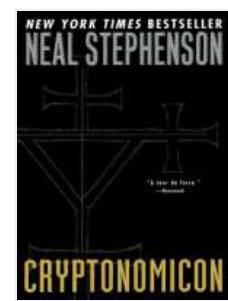
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

306 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

5.7. FOR XML Y OPENXML

Puede ejecutar consultas SQL para obtener resultados en formato XML en lugar de con formato estándar. Estas consultas pueden ejecutarse directamente o desde procedimientos y funciones definidas por el usuario. Para obtener los resultados directamente, utilice la cláusula FOR XML de la instrucción SELECT. A continuación, dentro de la cláusula FOR XML, un modo de XML: RAW, AUTO, EXPLÍCITO o PATH.

Sintaxis:

```

TRANSACT SQL
CAST Y CONVERT

[ FOR BROWSE | <XML> ]
<XML> ::=

XML
{
  { RAW [ ('ELEMENTNAME') ] | AUTO }
  [
    <COMMONDIRECTIVES>
    [ , XMLDATA | XMLSCHEMA [ ('TARGETNAMESPACEURI') ] ]
    [ , ELEMENTS [ XSINIL | ABSENT ] ]
  ]
  | EXPLICIT
  [
    <COMMONDIRECTIVES>
    [ , XMLDATA ]
  ]
  | PATH [ ('ELEMENTNAME') ]
  [
    <COMMONDIRECTIVES>
    [ , ELEMENTS [ XSINIL | ABSENT ] ]
  ]
}
```

RAW[('ElementName')]: obtiene el resultado de la consulta y transforma cada fila de los resultados en un elemento XML con un identificador genérico `<row />` como etiqueta de elemento raíz. Opcionalmente, puede especificar un nombre para el elemento de fila cuando se utiliza este modo. El XML resultante utilizará el ElementName especificado como el elemento de fila generado para cada fila. Para obtener más información, vea Usar el modo RAW.

AUTO: devuelve los resultados de la consulta en un árbol anidado XML sencillo. Cada fila de la consulta se convierte en un elemento XML. Si al menos una columna de la cláusula SELECT se representa en la cláusula FROM, se crea un elemento XML para cada fila. A las columnas presentadas en la cláusula SELECT se les asignan los nombres de los elementos XML apropiados. Para obtener más información, vea Usar el modo AUTO.

EXPLICIT: especifica que la forma del árbol XML resultante está definida explícitamente. En este modo, es necesario escribir las consultas de una cierta manera, de modo que se pueda obtener de forma explícita información adicional acerca de la anidación deseada. Para obtener más información, vea Usar el modo EXPLICIT.

8 views

1 like

0 comments

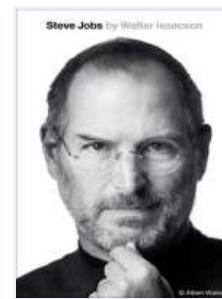
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

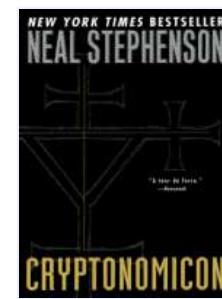
[Full description](#)

Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics



THE WOMAN WHO SMASHED CODES JASON FAGONE

& The Story of Lenore Horwitz and a Challenging Mathematical Journey

By Jason Fagone

Read by Jason Fagone

Length: 10 hrs 10 mins

Release date: Jun 10, 2014

Author: Jason Fagone

Genre: Biographies & Memoirs

Language: English

Published: Jun 10, 2014

Format: MP3 CD

ISBN: 9781442372587

ASIN: B00KJLWV0U

Label: Random House Audio

Producer: Random House Audio

Editor: Random House Audio

Copyright: © 2014 Jason Fagone

Number of Discs: 1

Disc Length: 10 hrs 10 mins

Number of Pages: 352

Language Note: English

Text-to-Speech: Enabled

ACX: Enabled

DRM: Enabled

File Size: 10 hrs 10 mins

Bit Rate: 128 kbps

Sample Rate: 48 kHz

Number of Formats: 1

Format Type: MP3

Number of Discs: 1

Disc Length: 10 hrs 10 mins

Number of Pages: 352

Language Note: English

Text-to-Speech: Enabled

ACX: Enabled

DRM: Enabled

File Size: 10 hrs 10 mins

Bit Rate: 128 kbps

Sample Rate: 48 kHz

Number of Formats: 1

Format Type: MP3

Number of Discs: 1

Disc Length: 10 hrs 10 mins

Number of Pages: 352

Language Note: English

Text-to-Speech: Enabled

ACX: Enabled

DRM: Enabled

File Size: 10 hrs 10 mins

Bit Rate: 128 kbps

Sample Rate: 48 kHz

Number of Formats: 1

Format Type: MP3

Number of Discs: 1

Disc Length: 10 hrs 10 mins

Number of Pages: 352

Language Note: English

Text-to-Speech: Enabled

ACX: Enabled

DRM: Enabled

File Size: 10 hrs 10 mins

Bit Rate: 128 kbps

Sample Rate: 48 kHz

Number of Formats: 1

Format Type: MP3

Number of Discs: 1

Disc Length: 10 hrs 10 mins

Number of Pages: 352

Language Note: English

Text-to-Speech: Enabled

ACX: Enabled

DRM: Enabled

File Size: 10 hrs 10 mins

Bit Rate: 128 kbps

Sample Rate: 48 kHz

Number of Formats: 1

Format Type: MP3

Number of Discs: 1

Disc Length: 10 hrs 10 mins

Number of Pages: 352

Language Note: English

Text-to-Speech: Enabled

ACX: Enabled

DRM: Enabled

File Size: 10 hrs 10 mins

Bit Rate: 128 kbps

Sample Rate: 48 kHz

Number of Formats: 1

Format Type: MP3

Number of Discs: 1

Disc Length: 10 hrs 10 mins

Number of Pages: 352

Language Note: English

Text-to-Speech: Enabled

ACX: Enabled

DRM: Enabled

File Size: 10 hrs 10 mins

Bit Rate: 128 kbps

Sample Rate: 48 kHz

Number of Formats: 1

Format Type: MP3

Number of Discs: 1

Disc Length: 10 hrs 10 mins

Number of Pages: 352

Language Note: English

Text-to-Speech: Enabled

ACX: Enabled

DRM: Enabled

File Size: 10 hrs 10 mins

Bit Rate: 128 kbps

Sample Rate: 48 kHz

Number of Formats: 1

Format Type: MP3

Number of Discs: 1

Disc Length: 10 hrs 10 mins

Number of Pages: 352

Language Note: English

Text-to-Speech: Enabled

ACX: Enabled

DRM: Enabled

File Size: 10 hrs 10 mins

Bit Rate: 128 kbps

Sample Rate: 48 kHz

Number of Formats: 1

Format Type: MP3

Number of Discs: 1

Disc Length: 10 hrs 10 mins

Number of Pages: 352

Language Note: English

Text-to-Speech: Enabled

ACX: Enabled

DRM: Enabled

File Size: 10 hrs 10 mins

Bit Rate: 128 kbps

Sample Rate: 48 kHz

Number of Formats: 1

Format Type: MP3

Number of Discs: 1

Disc Length: 10 hrs 10 mins

Number of Pages: 352

Language Note: English

Text-to-Speech: Enabled

ACX: Enabled

DRM: Enabled

File Size: 10 hrs 10 mins

Bit Rate: 128 kbps

Sample Rate: 48 kHz

Number of Formats: 1

Format Type: MP3

Number of Discs: 1

Disc Length: 10 hrs 10 mins

Number of Pages: 352

Language Note: English

Text-to-Speech: Enabled

ACX: Enabled

DRM: Enabled

File Size: 10 hrs 10 mins

Bit Rate: 128 kbps

Sample Rate: 48 kHz

Number of Formats: 1

Format Type: MP3

Number of Discs: 1

Disc Length: 10 hrs 10 mins

Number of Pages: 352

Language Note: English

Text-to-Speech: Enabled

ACX: Enabled

DRM: Enabled

File Size: 10 hrs 10 mins

Bit Rate: 128 kbps

Sample Rate: 48 kHz

Number of Formats: 1

Format Type: MP3

Number of Discs: 1

Disc Length: 10 hrs 10 mins

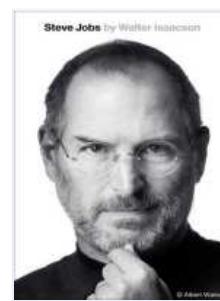
8 views

1 0

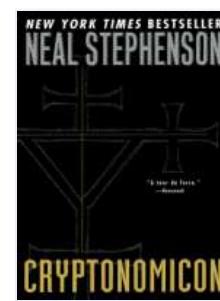
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

308

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

```
SELECT AE.IDAERO,AE.NOMBRE,PAI.PAIS
      FROM AEROPUERTO AE
      JOIN PAIS PAI ON AE.IDPAIS=PAI.IDPAIS
      FOR XML RAW
      GO
```

El resultado de la ejecución de la consulta se muestra en la siguiente imagen:

XML_F52E2B61-18A1-11d1-B105-00805F499168	
1	<row IDAERO="AE01" NOMBRE="BARILLOCHE" PAIS="ARGE"

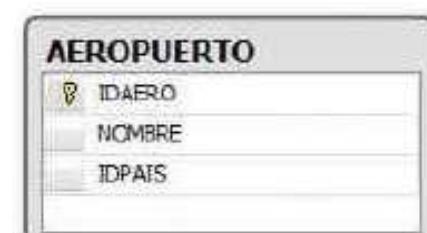
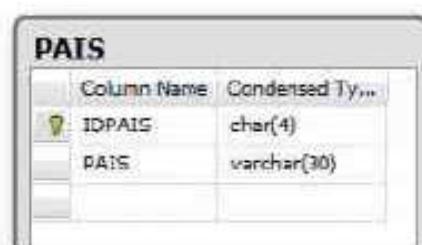
La imagen siguiente muestra el linko hacia el registro mostrado en la consulta.

```
<row IDAERO="AE01" NOMBRE="BARILLOCHE" PAIS="ARGENTINA" />
<row IDAERO="AE02" NOMBRE="MAR DEL PLATA" PAIS="ARGENTINA" />
<row IDAERO="AE03" NOMBRE="JORGE CHAVEZ" PAIS="PERU" />
<row IDAERO="AE04" NOMBRE="SANTIAGO" PAIS="CHILE" />
<row IDAERO="AE05" NOMBRE="AICM" PAIS="MEXICO" />
<row IDAERO="AE06" NOMBRE="JOSE JOAQUIN DE OLMEDO" PAIS="ECUADOR" />
<row IDAERO="AE07" NOMBRE="SIMON BOLIVAR" PAIS="VENEZUELA" />
<row IDAERO="AE08" NOMBRE="SAO PAULO CONGONHAS" PAIS="BRASIL" />
<row IDAERO="AE09" NOMBRE="SILVIO PETTIROSSI" PAIS="PARAGUAY" />
<row IDAERO="AE10" NOMBRE="CARRASCO PUERTA DEL SUR" PAIS="URUGUAY" />
```

Como notará en la imagen se crea un elemento XML genérico llamado row por cada fila desde la tabla AEROPUERTO.

CASO DESARROLLADO N° 5.8

Script que permite mostrar los registros de la tabla AEROPUERTO en un elemento XML generando la cláusula FOR XML en modo RAW, ELEMENTS.



```
SELECT AE.IDAERO,AE.NOMBRE,PAI.PAIS
      FROM AEROPUERTO AE
      JOIN PAIS PAI ON AE.IDPAIS=PAI.IDPAIS
      FOR XML RAW, ELEMENTS
      GO
```

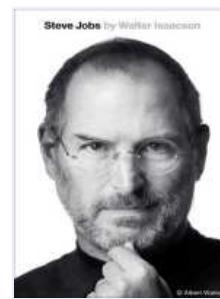
8 views

1 0

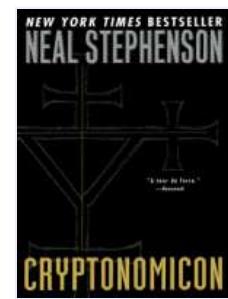
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics



[View details](#) [Buy](#)

The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

By Jason Fagone

Published on April 1, 2014

ISBN: 9780393346202

Language: English

Pages: 352

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

Dimensions: 6 x 1.2 x 9 inches

Weight: 1.2 pounds

Language: English

Format: Kindle Edition

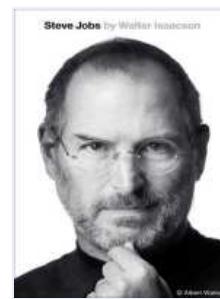
8 views

1 0

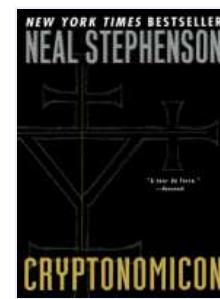
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

310 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 5.9

Script que permite insertar registros desde un archivo XML hacia la tabla PASAJERO.



```

INSERT INTO PASAJERO(IDPASAJERO,NOMBRES, IDPAIS,TELEFONO,EMAIL)
SELECT
    P.pasajeros.query('IDPASAJERO').value('.','CHAR(5)'),
    P.pasajeros.query('NOMBRES').value('.','VARCHAR(50)'),
    P.pasajeros.query('IDPAIS').value('.','CHAR(5)'),
    P.pasajeros.query('TELEFONO').value('.','VARCHAR(15)'),
    P.pasajeros.query('EMAIL').value('.','VARCHAR(50)')
FROM (SELECT CAST(P AS XML),
            SINGLE_BLOB) AS PAS(P)
)
AS PAS(P)
CROSS APPLY P.nodes('pasajeros/pasajero') AS P(pasajeros);
GO

```

Tenga mucho cuidado con la sensibilidad de las funciones mostradas en el script pues estrictos con respecto a las minúsculas y mayúsculas, recomiendo que lo digite todo el mi

Primero debemos comprobar los registros de la tabla PASAJERO, veamos:

	IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
1	P001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
2	P002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
3	P003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4	P004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5	P0013	LUCERO ERAZO	0004	988888888	LERAZO@GMAIL.COM
6	P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL...
7	P0007	NERY CALLE DE LACRUZ	0010	957564526	NCALLE@GMAIL.COM
8	P0008	HEIDI RENGIRO REATEGUI	0004	957564526	HRENGIRO@HOTMAIL.C...
9	P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
10	P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM
11	P0011	JANETH VILCHEZ	0002	988888888	JVILCHEZ@HOTMAIL.COM
12	P0012	KATHY CACSIRE	0001	988888889	KCACSIRE@HOTMAIL.COM

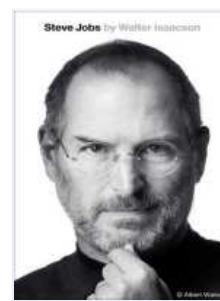
8 views

1 0

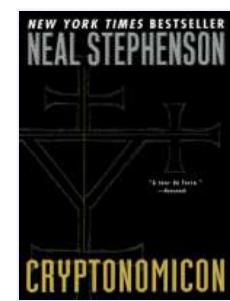
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



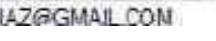
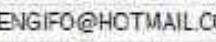
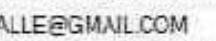
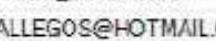
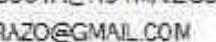
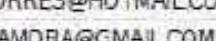
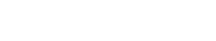
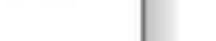
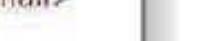
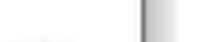
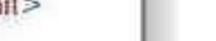
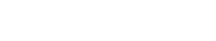
Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America



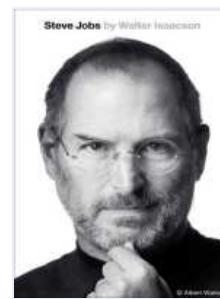
8 views

1 0

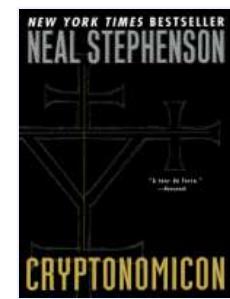
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

312 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 5.10

Script que permita buscar un determinado pasajero mediante su IDPASAJERO dentro de XML.

```

DECLARE @PXML XML
SET @PXML = '<pasajeros>
    <pasajero idpasajero="P1001">
        <nombres>MARIA LOPEZ</nombres>
        <idpais>0006</idpais>
        <telefono>999999999</telefono>
        <email>MLOPEZ@GMAIL.COM</email>
    </pasajero>
    <pasajero idpasajero="P1002">
        <nombres>JULIA ZEVALLOS</nombres>
        <idpais>0001</idpais>
        <telefono>999999999</telefono>
        <email>JZEVALLOS@GMAIL.COM</email>
    </pasajero>
    <pasajero idpasajero="P1003">
        <nombres>MARIAELENA MELO</nombres>
        <idpais>0001</idpais>
        <telefono>999999999</telefono>
        <email>MMELO@HOTMAIL.COM</email>
    </pasajero>
    <pasajero idpasajero="P1004">
        <nombres>LOURDES MORILLAS</nombres>
        <idpais>0002</idpais>
        <telefono>999999999</telefono>
        <email>LMORILLAS@GMAIL.COM</email>
    </pasajero>
    <pasajero idpasajero="P1005">
        <nombres>BRISA SPELUCIN</nombres>
        <idpais>0002</idpais>
        <telefono>999999999</telefono>
        <email>BSPELUCIN@HOTMAIL.COM</email>
    </pasajero>
</pasajeros>'
SELECT @PXML.query('/pasajeros/pasajero[./@idpasajero="P1003"]')
GO

```

El contenido del archivo XML es el siguiente:

```

<?xml version="1.0"?>
<pasajeros>
    <pasajero>
        <idpasajero>P1001</idpasajero>
        <nombres>MARIA LOPEZ</nombres>
        <idpais>0006</idpais>
        <telefono>999999999</telefono>
        <email>MLOPEZ@GMAIL.COM</email>
    </pasajero>
    <pasajero>
        <idpasajero>P1002</idpasajero>
        <nombres>JULIA ZEVALLOS</nombres>
        <idpais>0001</idpais>
        <telefono>999999999</telefono>
        <email>JZEVALLOS@GMAIL.COM</email>
    </pasajero>
    <pasajero>
        <idpasajero>P1003</idpasajero>
        <nombres>MARIAELENA MELO</nombres>
        <idpais>0001</idpais>
        <telefono>999999999</telefono>
        <email>MMELO@HOTMAIL.COM</email>
    </pasajero>
    <pasajero>
        <idpasajero>P1004</idpasajero>
        <nombres>LOURDES MORILLAS</nombres>
        <idpais>0002</idpais>
        <telefono>999999999</telefono>
        <email>LMORILLAS@GMAIL.COM</email>
    </pasajero>
    <pasajero>
        <idpasajero>P1005</idpasajero>
        <nombres>BRISA SPELUCIN</nombres>
        <idpais>0002</idpais>
        <telefono>999999999</telefono>
        <email>BSPELUCIN@HOTMAIL.COM</email>
    </pasajero>
</pasajeros>

```

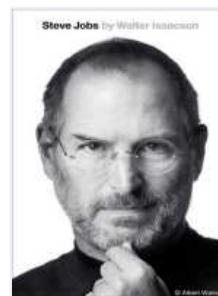
8 views

 1  0

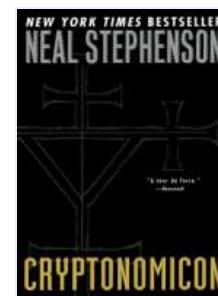
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by **anon_61286589**

Full description



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A Tri

 Save  Embed  Share  Print

1

<>

3

1

CAP. 5: XML con S

El resultado de la consulta ejecutada se muestra en la imagen siguiente:

```
<pasajero idpasajero="P1003">
    <nombres>MARIAELENA MELO</nombres>
    <idpais>0001</idpais>
    <telefono>9999999999</telefono>
    <email>MMELO@HOTMAIL.COM</email>
</pasajero>
```

Ahora pruebe con otro código cambiando `[./@idpasajero="P1003"]` por los otros códigos en el archivo XML.

5.8. MANEJO DE DATOS MASIVOS EN SQL SERVER

El motor de base de datos de Microsoft SQL Server permite importar y exportar masivamente entre una tabla de SQL Server y un archivo de datos que podría ser un texto plano txt.

SQL Server hacía referencia a la importación y exportación de la siguiente manera:

- **Exportación masiva:** se refiere a la copia de datos de una tabla de SQL Server en un archivo.
 - **Importación masiva:** significa cargar datos de un archivo de datos a una tabla de SQL Server.

Los métodos básicos disponibles en el motor de base de datos son:

- **BCP**: utilidad de línea de comandos que permite la importación y exportación de datos en formato de archivo.
 - **BULK INSERT**: instrucción de Transact SQL que importa datos directamente de un archivo en una tabla de una base de datos. Esta instrucción no permite la exportación de datos.
 - **OPENROWSET(BULK)**: instrucción Transact SQL que usa el proveedor de conjunto de datos OPENROWSET para importar masivamente desde un archivo externo hacia una tabla de datos. Es especialmente usado para la importación de imágenes masivas.

5.9. INSTRUCCIÓN BULK INSERT

La instrucción Bulk Insert importa un archivo de datos en una tabla con un formato definido por el usuario.

Sintaxis:

VOLCADO MASIVO BLIK INSERT

8 views

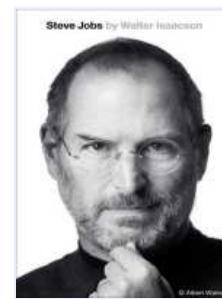
1 like

0 comments

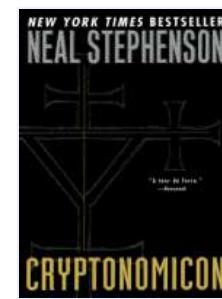
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

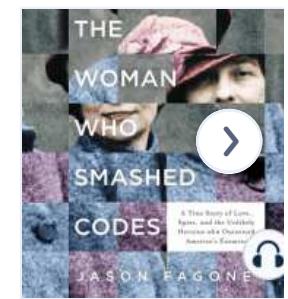
RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

314

PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

- **BASEDATOS:** es el nombre de la base de datos en la que reside la tabla o vista específica, es la base de datos actual.
- **ESQUEMA:** es el nombre del esquema de la tabla o vista. schema_name es opcional si predeterminado para el usuario que realiza la operación de importación masiva es el esquema de tabla o vista especificada. Si no se especifica schema y el esquema predeterminado del usuario realiza la operación de importación masiva es diferente de la tabla o vista especificada devuelve un mensaje de error y se cancela la operación de importación masiva.
- **TABLA:** es el nombre de la tabla o vista en la que se va a realizar una importación masiva. Sólo se pueden utilizar vistas en las que todas las columnas hagan referencia a la misma tabla.
- **'ARCHIVO':** es la ruta de acceso completa al archivo de datos que contiene los datos a importar en la tabla o vista especificada. BULK INSERT puede importar datos desde (incluidos una ubicación de red, disquete, disco duro, etc.).
- **FIELDTERMINATOR = 'SIMBOLO_CAMPO_TERMINADOR':** especifica el terminador de campo que se va a utilizar para archivos de datos de tipo char y widechar. El terminador de campo predeterminado es \t (tabulador).
- **FIRSTROW = NUMERO_FILA:** especifica el número de la primera fila que se va a cargar. El valor predeterminado es la primera fila del archivo de datos especificado.
- **ROWTERMINATOR = 'SIMBOLO_FILA_TERMINADOR':** especifica el terminador de fila que se va a utilizar para archivos de datos de tipo char y widechar. El terminador de fila predeterminado es el carácter de nueva línea.

CASO DESARROLLADO N° 5.11

Script que permite volcar los registros almacenados en un archivo txt hacia la tabla PASAJERO usando BULK INSERT.

PASAJERO	
IDPASAJERO	
NOMBRES	
IDPAIS	
TELEFONO	
EMAIL	

```
BULK INSERT PASAJERO
FROM 'C:\PASAJEROS.TXT'
WITH(
```

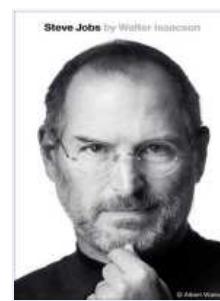
8 views

1 0

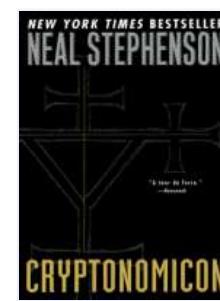
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

CAP. 5: XML CON SQL SERVER

Antes de explicar el script veamos cómo se encontraban los registros de la tabla PASAJERC

IDPASAJERO	NOMBRES	IDPAIS	TELEFONO	EMAIL
P0001	ANGELA TORRES LAZARO	0001	999999999	ATORRES@HOTMAIL.COM
P0002	FERNANDA TORRES LAZARO	0001	999999999	FTORRES@HOTMAIL.COM
P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
P0013	LUCERO ERAZO	0004	988888888	LERAZO@GMAIL.COM
P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM
P0011	JANETH VILCHEZ	0002	988888888	JVILCHEZ@HOTMAIL.COM
P0012	KATHY CACSIRE	0001	988888888	KCACSIRE@HOTMAIL.COM
P1001	MARIA LOPEZ	0006	999999999	MLOPEZ@GMAIL.COM
P1002	JULIA ZEVALLOS	0001	999999999	JZEVALLOS@GMAIL.COM
P1003	MARIAELENA MELO	0001	999999999	MMELO@HOTMAIL.COM
P1004	LOURDES MORILLAS	0002	999999999	LMORILLAS@GMAIL.COM
P1005	BRISA SPELUCIN	0002	999999999	BSPELUCIN@HOTMAIL.COM

En el script se hace referencia a la tabla destino en este caso PASAJERO será la tabla que recibe los valores volcados. En la cláusula FROM se especifica de dónde provienen los datos, habiendo en cuenta que dicho archivo se debe encontrar físicamente en dicha unidad y que tenga extensión .txt.

Si visualiza los campos especificados dentro del archivo de texto notará que por cada campo de una misma fila se usa el símbolo / pero podría ser cualquier símbolo como un asterisco (*) o coma (,) pero también hay que tener en cuenta que esos símbolos no intervengan en los espacios en blanco. Por ejemplo, el nombre del pasajero se separa por comas entre los apellidos y sus nombres; pero no se podrá usar el símbolo coma (,) para determinar el símbolo de finalización de campo, en tal caso particular se usa el símbolo '/' para dicho cambio.

En la cláusula ROWTERMINATOR se tiene que especificar cómo se realiza el cambio de fila. El volcado debe estar seguro de cuando se trata de otro registro, para eso usamos el operador de línea '\n'.

Y finalmente se especifica desde qué fila se tomarán los valores, esto es debido a que las aplicaciones de construcción de registros en texto plano ubican en la primera fila el nombre de las columnas de donde provienen, pero esto dependerá de cómo se realizó la conversión de texto. En este caso no hay nombres de columnas, la primera fila es un registro; por lo tanto, se coloca el nombre de la columna en el script y sea el caso que no lo implemente dentro del script también representará desde la primera fila. Entonces el script podría quedar de la siguiente manera y funcionará de la misma manera:

```
BULK INSERT PASAJERO
FROM 'C:\PASAJEROS.TXT'
```

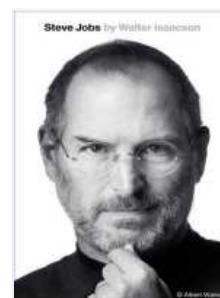
8 views

1 0

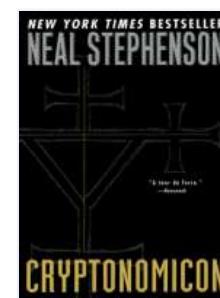
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

316 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Veamos los registros del archivo PASAJEROS.TXT para que los diferencie con los valores encontraban en la tabla, se modificó su código a P9 como valores iniciales a los valores p del archivo txt.

```

pasajeros.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
P9001/Barzola Antay, Basualdo/0001/988888888/bbarzola@hotmail.com
P9002/Bazan Fuentes, Nathaly Flor/0002/988888888/nbazan@gmail.com
P9003/Campos Flores, Flor/0001/988888888/fcampos@hotmail.com
P9004/Canicela Cobos, Fernando/0002/988888888/fcanicela@hotmail.com
P9005/Colonio Napa, Christian Jefferson/0006/988888888/ccolonia@hotmail.com
P9006/Flores Sanchez, Jhair Alberto/0001/988888888/jflores@gmail.com
P9007/Flores Ore, Nestor/0001/988888888/nflores@hotmail.com
P9008/Gálvez Velezmoro, Mayra Alejandra/0006/988888888/mgámez@hotmail.com
P9009/Garay Pimentel, Mary Elizabeth/0002/988888888/mgaray@gmail.com
P9010/Mestas Oxa, Jose Miguel/0006/988888888/jmestas@hotmail.com

```

Una vez ejecutado el script se muestra el siguiente mensaje desde el motor de base de datos que confirma que los registros del archivo han sido volcados a la tabla PASAJERO.

(10 filas afectadas)

Finalmente, revisemos si se volcó los registros desde el archivo PASAJEROS.TXT hacia la tabla PASAJERO.

IDPASAJERO	NOMERES	IDPAIS	TELEFONO	EMAIL
3 P0003	MARIA ZAMORA MEJIA	0005	957564526	MZAMORA@GMAIL.COM
4 P0004	GUADALUPE ACOSTA FERRER	0002	957564526	GACOSTA@HOTMAIL.COM
5 P0013	LUCERO ERAZO	0004	98888888	LERAZO@GMAIL.COM
6 P0006	KARLA GALLEGOS SILVA	0007	957564526	KGALLEGOS@HOTMAIL.COM
7 P0007	NERY CALLE DE LA CRUZ	0010	957564526	NCALLE@GMAIL.COM
8 P0008	HEIDI RENGIFO REATEGUI	0004	957564526	HRENGIFO@HOTMAIL.COM
9 P0009	MARISOL DIAZ ZAMBRANO	0004	957564526	MDIAZ@GMAIL.COM
10 P0010	LINDA TUME VARAS	0008	957564526	LTUME@HOTMAIL.COM
11 P0011	JANETH VILCHEZ	0002	988888888	JVILCHEZ@HOTMAIL.COM
12 P0012	KATHY CACSIRE	0001	988888889	KCACSIRE@HOTMAIL.COM
13 P1001	MARIA LOPEZ	0006	999999999	MLOPEZ@GMAIL.COM
14 P1002	JULIA ZEVALLOS	0001	999999999	JZEVALLOS@GMAIL.COM
15 P1003	MARIAELENA MELO	0001	999999999	MMELO@HOTMAIL.COM
16 P1004	LOURDES MORILLAS	0002	999999999	LMORILLAS@GMAIL.COM
17 P1005	BRISA SPELUCIN	0002	999999999	BSPELLUCIN@HOTMAIL.COM
18 P9001	Barzola Antay, Basualdo	0001	988888888	bbarzola@hotmail.com
19 P9002	Bazan Fuentes, Nathaly Flor	0002	988888888	nbazan@gmail.com
20 P9003	Campos Flores, Flor	0001	988888888	fcampos@hotmail.com
21 P9004	Canicela Cobos, Fernando	0002	988888888	fcanicela@hotmail.com
22 P9005	Colonio Napa, Christian Jefferson	0006	988888888	ccolonia@hotmail.com
23 P9006	Flores Sanchez, Jhair Alberto	0001	988888888	jflores@gmail.com
24 P9007	Flores Ore, Nestor	0001	988888888	nflores@hotmail.com

8 views

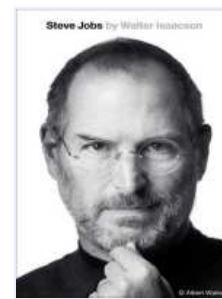
1 like

0 comments

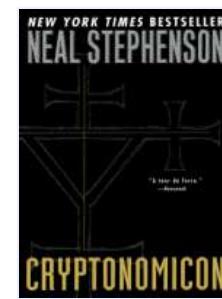
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics



CAP. 5: XML CON SQL SERVER

5.10. INSTRUCCIÓN OPENROWSET

Contiene toda la información de conexión necesaria para tener acceso a datos remotos desde datos OLE DB. Es un método alternativo para tener acceso a las tablas de un servidor al mismo tiempo, es un método ad hoc para conectarse y tener acceso a datos remotos utilizando DB.

Características:

- Se puede hacer referencia a la función OPENROWSET en la cláusula FROM de una consulta fuera el nombre de una tabla.
- También se puede hacer referencia a la función OPENROWSET como tabla de destino en instrucción INSERT, UPDATE o DELETE, sujeta a la funcionalidad del proveedor OLE DB.
- Aunque la consulta puede devolver varios conjuntos de resultados, OPENROWSET sólo devuelve el primero.
- OPENROWSET también admite operaciones masivas a través de un proveedor integrado que permite que los datos se lean y se devuelvan como un conjunto de filas.

Sintaxis:

**VOLCADO MASIVO
OPENROWSET**

```
OPENROWSET
(
    BULK 'ARCHIVO_DATOS' , SINGLE_BLOB | SINGLE_CLOB | SINGLE_NCLOB
)
```

- **BULK:** utiliza el proveedor de conjuntos de filas BULK para que OPENROWSET lea los datos de un archivo. En SQL Server, OPENROWSET puede leer datos de un archivo sin necesidad de crear una tabla de destino. Esto le permite utilizar OPENROWSET con una instrucción SELECT. Los argumentos de la opción BULK le permiten elegir dónde empezar y acabar la lectura, cómo abordar los errores y cómo interpretar los datos. Por ejemplo, puede especificar qué tipo de datos se lee como un conjunto de filas de una sola fila y una sola columna de tipo varchar o nvarchar.
- **SINGLE_BLOB:** devuelve el contenido de ARCHIVO_DATOS como un conjunto de filas de tipo varbinary(max) de una sola fila y una sola columna.
- **SINGLE_CLOB:** al leer ARCHIVO_DATOS como ASCII, se devuelve el contenido como un conjunto de filas de tipo varchar(max) de una sola fila y una sola columna, mediante la intercalación.

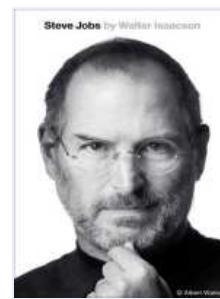
8 views

1 0

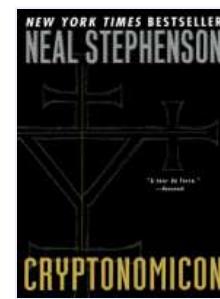
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

318 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

CASO DESARROLLADO N° 5.12

Script que permite volcar el contenido de un archivo de tipo XML en una sola fila de SQL s

```
--1.
DECLARE @RUTA VARCHAR(100) = 'C:\PASAJEROS.XML'
DECLARE @SQL VARCHAR(500)

--2.
SET @SQL = 'SELECT BULKCOLUMN
            FROM OPENROWSET(BULK ''' + @RUTA + ''', SINGLE_CLOB)
            AS DATOS_XML'
EXEC(@SQL)
GO
```

En el punto uno se declaran las variables @RUTA que tienen por misión asignar la ruta fuente a volcar, en este su archivo PASAJERO.XML se encuentra en la unidad C:\, también la variable @SQL que almacenará el volcado convirtiéndose en la variable destino por es declaración siempre debe ser mínimamente VARCHAR o NVARCHAR de una capacidad ex caso se predice un máximo de 500 caracteres pero podría ser MAX.

En el punto dos se asigna la consulta BULKCOLUMN a la variable @SQL proveniente de la OPENROWSET especificando en este caso el volcado BULK, la ruta del archivo fuente y el 1 devuelto; en este ultimo se tiene que usar SINGLE_CLOB ya que tiene que convertir los d texto legible.

Finalmente, se ejecuta la consulta por medio de la función EXEC que tiene por misión e instrucción preparada en Transact SQL.

El resultado de la ejecución se muestra en la siguiente imagen:

```
BulkColumn
1 <pasajeros> <pasajero> <idpasajero>P0004</idpasajero> <email>tutes@hotmail.com</email> </pasajero> <p...
```

Como notará se volcó todo el contenido del archivo externo PASAJERO.XML en una sola mostramos una segunda versión del mismo caso:

```
DECLARE @RUTA VARCHAR(100) = 'C:\PASAJEROS.XML'
DECLARE @SQL NVARCHAR(MAX)

SET @SQL = 'SELECT BULKCOLUMN
            FROM OPENROWSET(BULK ''' + @RUTA + ''', SINGLE_CLOB)
            AS DATOS_XML'

EXEC SP_EXECUTESQL @SQL
--
```

8 views

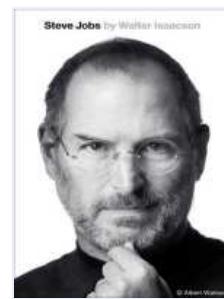
1 like

0 comments

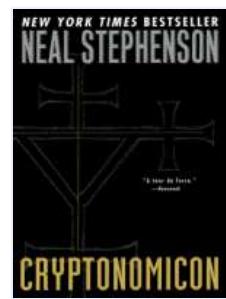
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
 [Save](#)
 [Embed](#)
 [Share](#)
 [Print](#)

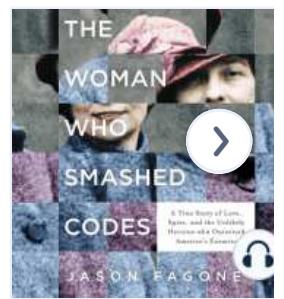
RELATED TITLES



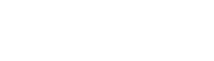
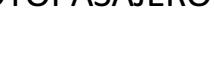
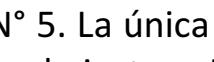
Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics



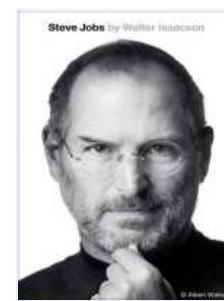
8 views

1 0

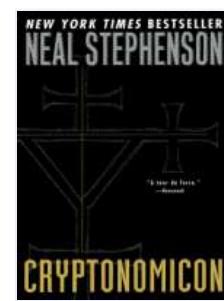
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

320 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Como producto de la inserción de seis registros con una columna para la fotografía se conformidad del motor de base de datos en la siguiente imagen:

```
(1 filas afectadas)
```

La imagen siguiente muestra el resultado de consultar los registros de la tabla **FOTOPASA**.

IDPASAJERO	FOTO
P0001	0xFFD8FFE000104A46494600010202012C012C0000FFE10E0D4578696600004D4D002A000000...
P0002	0xFFD8FFE000104A46494600010202012C012C0000FFE10ECD4578696600004D4D002A000000...
P0003	0xFFD8FFE000104A46494600010202012C012C0000FFE10ECE4578696600004D4D002A000000...
P0004	0xFFD8FFE000104A46494600010202012C012C0000FFE10EDA4578696600004D4D002A000000...
P0005	0xFFD8FFE000104A46494600010202012C012C0000FFE10F3B4578696600004D4D002A000000...
P0006	0xFFD8FFE000104A46494600010202012C012C0000FFE10D9F4578696600004D4D002A000000...

Como vera no se muestra la fotografía ya que su tipo de datos es varbinary, para nuestro caso una aplicación que permitirá mostrar dichas imágenes el software es SQL IMAGE VIEWER.

Para poder descargar la versión de prueba entre a la siguiente dirección URL <http://www.yohz.com/downloads.htm> desde aquí deberá seleccionar: SQL Image Viewer

The screenshot shows a web browser window with the following details:

- Address Bar:** http://www.yohz.com/downloads.htm
- Title Bar:** Yohz Software - Downloads
- Content Area:**
 - YOHZ SOFTWARE** logo
 - Downloads** section
 - Database tools** icon (represented by three green cylinders)
 - Navigation Links:** Home, Products, Support, Contact, About, News, Forum, Log In, Sign Up

8 views

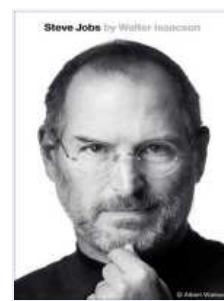
1 like

0 comments

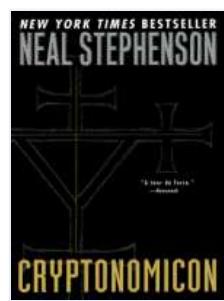
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



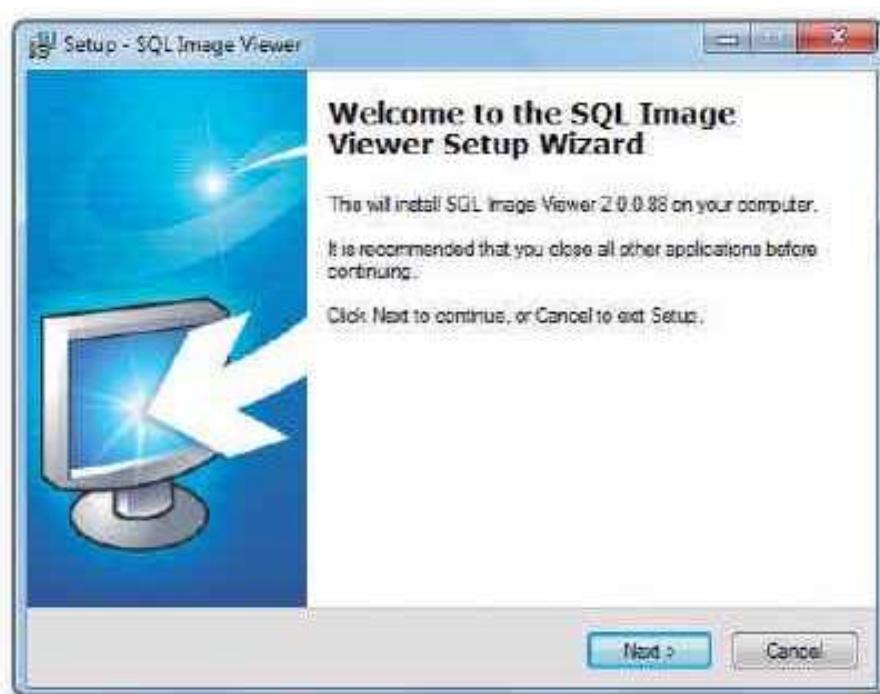
Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in Computer Science

CAP. 5: XML CON SQL SERVER

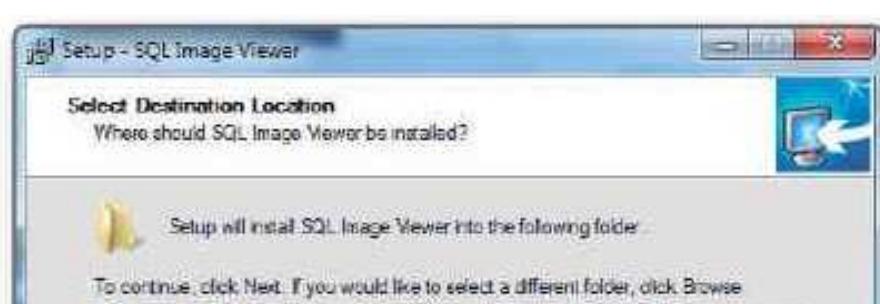
Veamos el proceso de instalación del SQL Image Viewer:



Sólo presione Next para iniciar el proceso de instalación.



Acepte la licencia de la aplicación, recuerde que sólo es una versión de prueba.



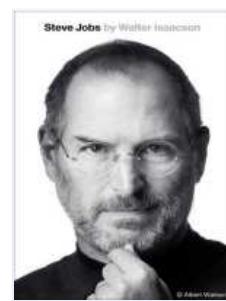
8 views

1 0

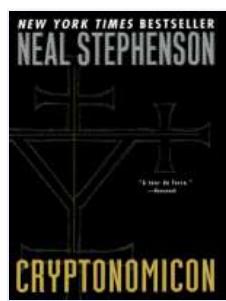
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



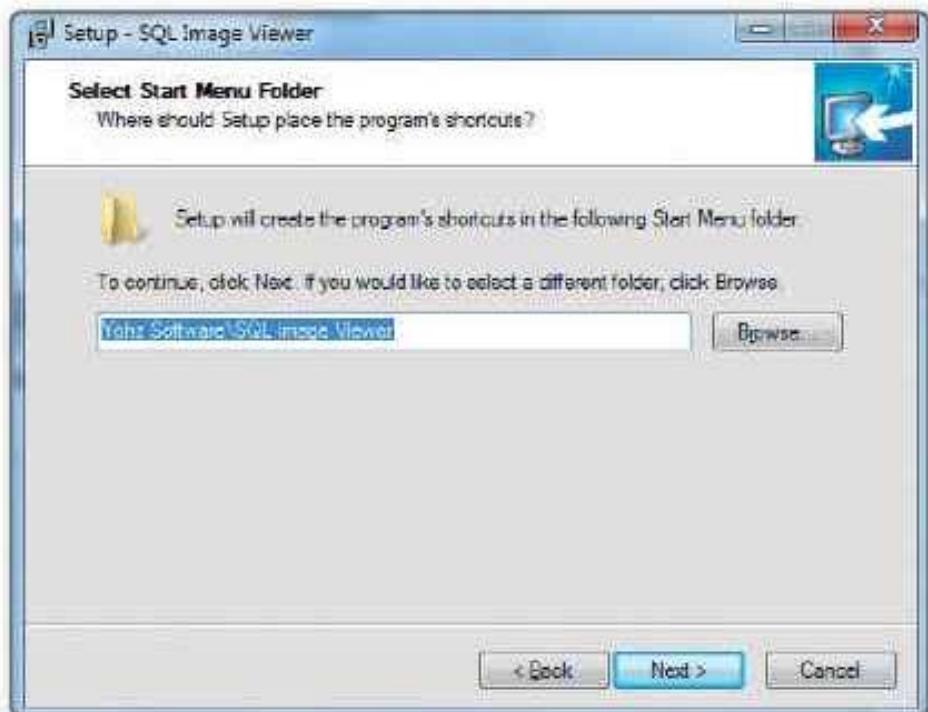
Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and the Violent History of Mathematics

322 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Dírecte donde se grabarán los archivos fuentes de la aplicación.



Asigne un nombre adecuado a la aplicación, así será como Windows lo reconocerá a la ap



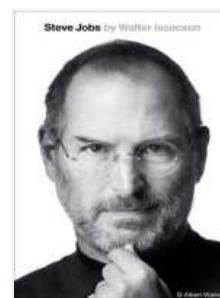
8 views

1 0

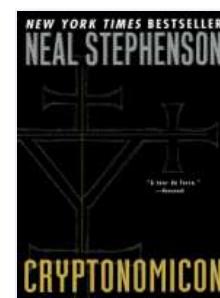
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon

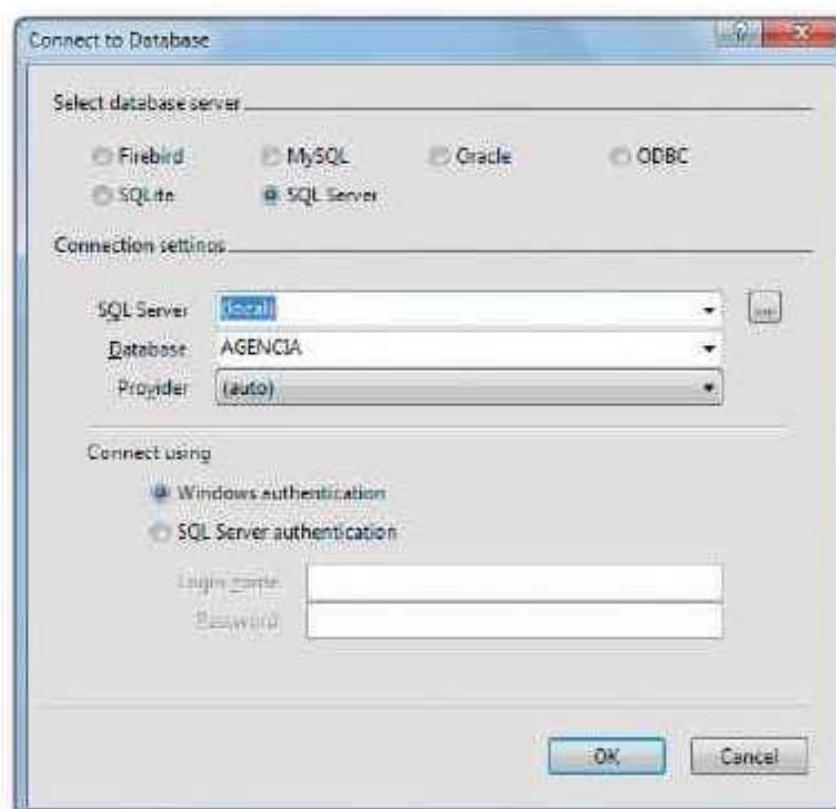


The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America



CAP. 5: XML CON S

Una vez finalizada la instalación se procederá a ejecutar la misma con solo activar el **Setup Image Viewer.exe** y presionar **Finish**.



Como verá esta aplicación presenta múltiples alternativas, si usted maneja imágenes en otra base de datos también podrá visualizar las imágenes por medio de esta aplicación.

En la ventana Connect to Database debe seleccionar **SQL Server**, en SQL Server seleccionar **Local** como el servidor, el Database **AGENCIA** y no debe olvidarse de usar la autenticación según la configuración ir a la sección **Connect using**. En este caso usaremos el Windows authentication.



Luego se mostrará la ventana principal de la aplicación y como notará presenta un SQL editor donde usted podrá colocar el script que desea y podrá ejecutarlo como si se encontrase dentro de SQL Server. En el lado derecho de la aplicación se muestran todas las tablas que tiene la base de datos.



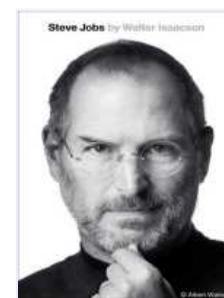
8 views

1 0

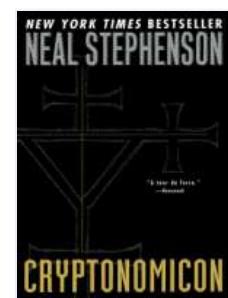
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)
[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hatred, and Violent Genius

324 PROGRAMACIÓN TRANSACT CON SQL SERVER 2012

Empezaremos por consultar la tabla FOTOPASAJERO que como recordará se implementó para almacenar fotografías de los pasajeros. En esta parte no debe añadir la sentencia GO ya que es reconocida como instrucción y generará un error.

El script se puede ejecutar seleccionándolo y presionando F5 o en su defecto presionar el botón **EXECUTE**.

The screenshot shows the SQL Image Viewer application interface. At the top, there's a toolbar with icons for Database, Tools, Help, and various file operations like Open, Save, and Export. Below the toolbar, the database is set to 'MS SQL (local)' and the database name is 'AGENCIA'. The main window has two tabs: 'SQL editor' (which is active) and 'Results'. In the 'Results' tab, there is a table titled 'Data (6 rows)' with columns 'IDPASAJERO' and 'FOTO'. The data is as follows:

IDPASAJERO	FOTO
1	(BINARY)
2	(BINARY)
3	(BINARY)
4	(BINARY)
5	(BINARY)
6	(BINARY)

Below the table, the status bar indicates 'Ready (unnamed)' and 'SQL Server 2008 Developer Edition RTM | Mitores-PC\mitores | Unicode | 0:00:00 | Rows: 0 | Ln 1, Col 1'.

Como verá las imágenes de los pasajeros se podrán visualizar tal como se preparó el script, es decir, el orden de las imágenes responde al orden implementado en la sentencia SELECT.

The screenshot shows the SQL Image Viewer application interface again. The 'SQL editor' tab is active, displaying the following SQL query:

```

1: SELECT * FROM FOTOPASAJERO
2: WHERE IDPASAJERO='P0001'
3:

```

To the right of the editor, a tooltip provides information about binary data: 'Tables containing binary data (double click to retrieve row samples)' and lists 'dbo.FOTOPASAJERO' and 'dbo.sysdiagrams'. The status bar at the bottom indicates 'Ready (unnamed)' and 'SQL Server 2008 Developer Edition RTM | Mitores-PC\mitores | Unicode | 0:00:00 | Rows: 1 | Ln 2, Col 25'.

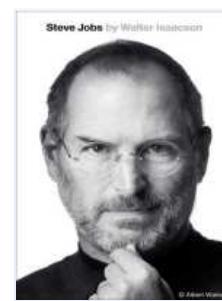
8 views

1 0

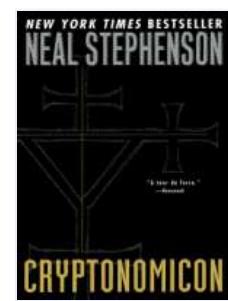
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)[Full description](#)[Save](#) [Embed](#) [Share](#) [Print](#)

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Coders in All of Computing

CAP. 5: XML CON S

La imagen siguiente muestra el resultado de la ejecución de la instrucción:

The screenshot shows the SQL Image Viewer application interface. At the top, it displays the database connection information: "MS SQL (local)" and the current database: "AGENCIA". Below the toolbar, there are two tabs: "SQL editor" and "Results". The "Results" tab is active, showing a table named "Images" with one row. The table has two columns: "IDPASAJERO" and "FOTO". The data row shows "P0004" in the first column and "(BINARY)" in the second. Above the table, there is a preview area showing a thumbnail of a woman's face. The status bar at the bottom indicates the session is "Ready" and provides other system details.

IDPASAJERO	FOTO
P0004	(BINARY)

8 views

1 0

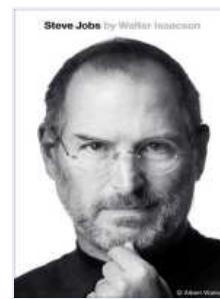
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

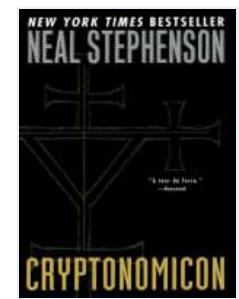
[Full description](#)

   
Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who
Smashed
Codes: A Tru

8 views

1 0

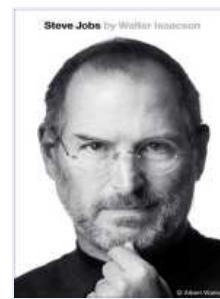
eMacro - Programación Transact SQL Server 2012-4.pdf

Uploaded by [anon_61286589](#)

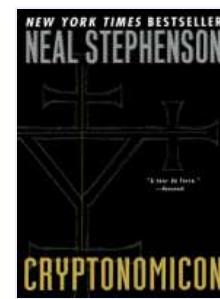
[Full description](#)

   
Save Embed Share Print

RELATED TITLES



Steve Jobs



Cryptonomicon



The Woman Who Smashed Codes: A True Story of Love, Hate, and the Violent History of One of the Greatest Codebreakers in America

Impreso en los Talleres Gráficos de



Surquillo
 719-9700