PROYECTO

CI-1314 BASES DE DATOS II
II SEMESTRE 2014
Dra. Elzbieta Malinowski
Versión 4

OBJETIVO GENERAL:

Crear una base de datos espacial en tercera forma normal usando datos provenientes de los archivos tipo *shape* y hojas electrónicas o archivos tipo texto. Además, verificar diferentes aspectos que afectan el tiempo de ejecución de las consultas.

OBJETIVOS ESPECÍFICOS:

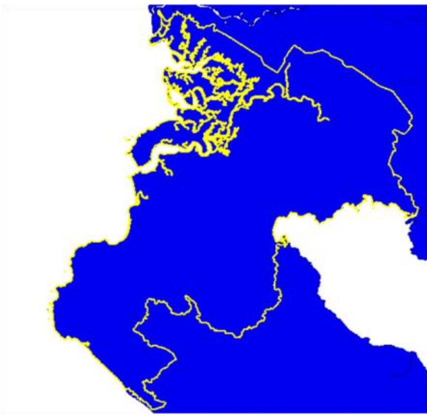
1. Crear una base de datos espacial en tercera forma normal para evitar los problemas de repetición e inconsistencia de datos basándose en datos espaciales presentes en los archivos tipo *shape* y datos convencionales provenientes de diferentes orígenes.
2. Implementar los controles declarativos y dinámicos para mejorar la calidad de datos.
3. Importar los datos espaciales y convencionales realizando las transformaciones necesarias.
4. Realizar las consultas de análisis sobre la bases de datos implementada.
5. Implementar y comparar las estadísticas de ejecución de las consultas expresadas en forma diferente.

DESCRIPCIÓN:

Actualmente existe una gran facilidad de obtener los archivos tipo *shape* para contar con los datos territoriales y poder usarlos en diferentes tipos de aplicaciones. Sin embargo, debido a que estos archivos se basan en la arquitectura híbrida y enfocan más en el despliegue y manipulación de datos espaciales, en muchas ocasiones no se lleva a cabo ningún tipo de control sobre la calidad de datos. Por ejemplo, uno de los archivos *shape* que representa los distritos de Costa Rica con su población en diferentes años tenía 849 elementos (*features*). En este archivo, 45 distritos estaban representados por más que 1 elemento y entre ellos el distrito de Sierpe estaba representado en 61 elementos como se puede ver en la figura 1 en un pequeño extracto del archivo. En estos elementos repetidos, el único atributo diferente es la geometría debido a que cada elemento representa una parte de la geometría de Sierpe. Cuando se pasaron estos datos a SABD espacial y se realizó una consulta simple sumando la población de todos distritos, como lo haría un usuario confiando en la calidad de datos, se obtuvo el valor de población total en año 2000 de casi 7 millones de habitantes.

Además se debe considerar que aunque archivos tipo *shape* tienen (en su mayoría) incluidos datos convencionales, éstos muchas veces son insuficientes para realizar análisis requerido. Por otro lado, existe una variedad de orígenes de datos (hojas electrónicas, archivos planos, entre otros) que contienen datos convencionales útiles, pero carecen de la ubicación especial (por ejemplo, ubicación de cada uno de los

cantones) que podría facilitar la búsqueda de patrones entre los datos que de otra forma son difíciles de encontrar.



PROVINCIA	NCANTON	NDISTRITO	PROV	CANTON	DISTRITO	CODDIST	POBLACION
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124
PUNTARENAS	OSA	SIERPE	6	5	3	60503	4124

Figura 1. Un pequeño extracto de la información repetida en el archivo tipo *shape*

Para evitar las situaciones que llevan a tener datos inconsistentes, se debe realizar un diseño de tablas en tercera forma normal (ignorando el atributo de geometría) integrando todo el conjunto de datos de interés para la aplicación. Además, para fortalecer el control de calidad de datos, durante la implementación de estas bases de datos se deben aprovechar los diferentes controles que los SABD poseen, por ejemplo, controles declarativos definidos durante la creación de tablas como integridad referencial, valores no nulos, valores únicos, revisiones tipo *check* y controles dinámicos como disparadores (*triggers*).

Adicionalmente del aspecto del diseño e implementación de controles para mejorar la calidad de datos, se debe considerar que los datos espaciales son de gran volumen y requieren expresar las consultas en forma que se minimice su tiempo de ejecución. Para esto los SABD incluyen componentes de software llamados optimizadores de consultas que analizan la consulta para seleccionar una estrategia que mejora su tiempo de ejecución. Sin embargo, en muchas ocasiones se requiere la intervención del implementador para lograr la mejoría. Para poder realizarlo, el implementador tiene que ser capaz de entender los aspectos que pueden afectar el desempeño de la consulta, poder expresar la misma consulta en diferentes formas, forzar la ejecución específica, verificar los planes y tiempo de ejecución, entre otros.

ETAPAS DE EJECUCIÓN DEL PROYECTO

El proyecto consiste en dos etapas: (1) diseño de la base de datos espacial enriquecido con datos convencionales, (2) implementación y consultas. Cada etapa incluye la documentación que debe ser entregada en la plataforma Moodle (e impresa, si se solicita), antes de la hora de clases en fechas indicadas abajo. Debe crear la documentación en forma **organizada** de acuerdo a las especificaciones descritas en cada una de las etapas.

Etapa 1 (30%)

Descripción

En esta etapa se entregará a los estudiantes los archivos tipo *shape* y archivos con datos convencionales (por ejemplo, Excel). Cada grupo debe analizar estos archivos y proponer un diseño conceptual usando el modelo ER con extensiones espaciales (con la notación usada en el curso) de acuerdo a la especificación brindada más adelante. Este diseño debe estar en tercera forma normal (suponiendo que la geometría es un campo atómico), o sea, evitar los problemas de repetición de datos como se explicó en la parte de "Descripción". En el caso de considerar la extensión de este proyecto por datos convencionales o inclusión de datos diferentes (con la aprobación de la profesora del curso) se pueden consultar los siguientes sitios: INEC (http://www.inec.go.cr/), Observatorio de Desarrollo de la UCR (http://www.tdc.odd.ucr.ac.cr/), Centro Centroamericano de Población (http://censos.ccp.ucr.ac.cr/) u otros.

Requerimientos mínimos:

1. En cada archivo *shape* considerar los atributos que permiten realizar algún tipo de análisis con respecto al sistema implementado, por ejemplo, excluir datos tipo el número de teléfono.
2. Usar el archivo *shape* referente a las carreteras de Costa Rica (archivo *shape* "redcaminos"). Aplicar el proceso de limpieza y agregación para que cada ruta se presente una solo vez con su geometría compuesta. Para las rutas sin nombre buscar la ruta que tiene alguna relación topológica e introducir un nuevo nombre que refleja esta dependencia de la otra ruta o su continuación.
3. Incluir el archivo de bomberos considerando los datos que tiene para análisis y aplicándole adecuada limpieza.
4. Incorporar los archivos de distrito, cantón y provincia ampliando los datos de los distritos por los que se encuentran en la hoja Excel con el nombre: "Población total por sexo y total de viviendas por ocupación, según provincia, cantón y distrito". Para cada uno de los archivos *shape* añadir el atributo calculado de área
5. Asignar cada estación de bomberos a su respectivo distrito basándose en la relación topológica que existe entre los objetos espaciales
6. Establecer la relación entre cantones y rutas considerando la relación topológica de intersección y calculando la longitud de esta intersección. Además, incluir para los cantones los datos integrados de los archivos Excel que se encuentran en el directorio "Transporte" (seleccionando algún año de interés). Es importante la integración adecuada de esta información.

7. Incluir el archivo *shape* de riesgos de incendio agrupando los riesgos de acuerdo a la clasificación presente en este archivo referente a la velocidad de viento y clase de riesgo..
8. No olvidar presentar adecuadamente todos los tipos de entidades y tipos de relaciones que considerar adecuadas para el sistema en cuestión.

Verificación del diseño (10%)

La verificación del diseño se realizará durante breve presentación donde se debe mostrar la estructura original de datos y el diseño propuesto. Gracias a la participación de otros grupos de proyecto durante la presentación, se espera que los estudiantes reciban una retroalimentación necesaria para mejorar su diseño, si fuese necesario.

Documentación (90%)

La documentación debe contener como mínimo las siguientes partes:

1. (5%) Objetivo de la aplicación: se debe explicar brevemente cuál objetivo podría cumplir el sistema propuesto y que el tipo de análisis se podría realizar.

2. (10%) Descripción de los archivos *shape*, hojas electrónicas, archivos planos, etc. usados en forma de la tabla, donde se indica el nombre del archivo y una breve descripción de los atributos que contiene. Además, debe incluirse la especificación de los atributos que violan la tercera forma normal (3FN), justificando la respuesta. La tabla 1 presenta una forma genérica de esta tabla y la tabla 2 incluye un ejemplo concreto del archivo tipo *shape* para los cantones de Costa Rica.

Tabla 1. Formato genérico de la tabla que describe el origen de datos provenientes de archivos tipo *shape*.

Nombre del archivo *shape*	atr_1	atr_2	...	at_n
Breve descripción de su significado (si conocido)				
Tipo de Dato				
Observaciones sobre valores				
Violación de 3FN				
Razones de la de violación de 3FN				

Tabla 2. Un ejemplo de la tabla que describe el contenido del archivo tipo *shape* de Cantones:

Administrativa	NProvincia	Prov	Ncanton	CodNum	Población	GID	
Breve descripción de su significado (si conocido)	Nombre de la provincia	Código de la provincia	Nombre del cantón	Código del cantón	Población actual	Identificador de la tupla	Geometría
Tipo de Dato	Texto	Número	Texto	Número	Número	Número	Polígono
Observaciones sobre valores	Datos repetidos	1digito, datos repetidos	Datos repetidos	5 dígitos, código repetido.	Se encuentran muchos valores nulos	Valor consecutivo para cada elemento	Representado como WKT. La geometría del mismo cantón se encuentra en varios elementos representando sus partes
Violación de 3FN	Si	Si	Si	Si	No	No	No
Razones de la violación de 3FN	Nombre de la provincia está determinado por el código de la provincia y éste no es la llave primaria	Determina el nombre de la provincia y no es la llave primaria	Nombre del cantón está determinado por el código del cantón y éste no es la llave primaria	Determina nombre del cantón y su población. No determina la geometría			

Nota: se puede representar las dependencias funcionales usando las flechas como se hicieron las prácticas en el curso de Bases de datos I.

3. (40%) Propuesta del nuevo esquema a nivel conceptual:
   a. El esquema conceptual original presentado durante la verificación del diseño y mejorado (si es diferente) que incluye los tipos de entidades con atributos, los tipos de relaciones (con atributos, si aplica), tipos de relaciones ISA (si aplica), las llaves candidatas, cardinalidades de mapeo, participaciones, pictogramas para atributos espaciales y relaciones topológicas.
   b. Cualquier restricción adicional no presente en el esquema.

Nota:
SOLO SE PERMITE PRESENTAR LAS SIGUIENTES PARTES DE LA DOCUMENTACIÓN DESPUÉS DE TENER APROBADO EL NUEVO ESQUEMA.

4. (35%) Especificación del esquema lógico con restricciones
   a. Se deben aplicar las reglas de mapeo entre los elementos del modelo ER extendido por datos espaciales y modelo (objeto-) relacional como se vio en clases. Cada tabla debe incluir las restricciones con respecto a valores nulos, valores únicos, rangos, entre otros. Los resultados se deben representar como se muestra en la tabla 3 en forma genérica con un ejemplo específico presentado en la tabla 4.

      Tabla 3. Resultados del mapeo lógico (forma genérica)

Nombre de la Tabla	atr_1	atr_2	...	at_n
Restricciones				
Tipo de Dato				
Longitud				

      Las restricciones pueden ser:
      - PK: llave primaria.
      - U: llave única.
      - U#: llave única compuesta. Se pone el mismo número para los atributos que forman esta llave.
      - FK: llave externa.
      - Cualquier otra, para la cual debe incluir la explicación de leyenda usada.

      Tabla 4. Ejemplo de una parte de la tabla empleado de la base de datos Compañía.

EMPLEADO	**Nombre**	**Apellido**	**NSS**	**Fechan**	**Dirección**	**Sexo**	**Salario**	**NSS Super**	**ND**
Restricciones			PK			M | F	> 30000	FK	ND > 0
Tipo de Dato	char	char	num	date	char	char	num	num	num
Longitud	50	50	9		100	1	8,3	9	2

      La longitud no es en bytes, sino corresponde al formato del tipo de datos. Por ejemplo, para el dato numérico puede ser 8,3 (8 posiciones enteras y 3 decimales).
      Para la geometría especifique su tipo (por ejemplo, punto) en la fila de longitud.

   b. Para cada tabla se debe justificar su tercera forma normal (sin considerar el atributo de geometría) con respecto a la llave primaria.

<u>Fecha de verificación del diseño</u>
29 de setiembre de 2014

<u>Fecha de entrega de la documentación completa de la I etapa:</u>
16 de octubre 2014

Etapa 2 (70%)

Descripción

En esta etapa se implementará el esquema propuesto usando SQL Server 2008 u otro DBMS (con la aprobación de la profesora del curso). Se debe implementar cada una de las tablas con las restricciones especificadas en el diseño lógico. Se deben incluir las acciones específicas para la integridad referencial y ser capaz de justificar su escogencia. Además, debido a que el esquema conceptual es más expresivo que el esquema lógico/físico de la base de datos, se debe asegurar el cumplimiento de las restricciones referentes al tipo de geometría y relaciones topológicas usando las facilidades que ofrece la DBMS (restricciones declarativas) o por medio de disparadores (*triggers*).

Después de crear las tablas se deben introducir los datos de los archivos *shape* y de la(s) hoja(s) electrónica(s) o archivo(s) tipo texto realizando las limpiezas, transformaciones e integración necesarias para mejorar calidad de los datos hasta donde sea posible. Esto incluye, por ejemplo, la agregación de geometrías si el mismo objeto está representado en varias tuplas debido a su geometría compuesta, la eliminación o modificación de valores nulos o desconocidos, corrección de la especificación de la geometría (ver algunos recomendaciones en el Anexo 1), unificación de nombre, entre otros. Cada uno de los archivos *shape* y archivos Excel o texto puede requerir diferentes tipos de transformaciones. Recuerdan representar las relaciones entre dos (o más) tipos de entidades espaciales por medio de las relaciones topológicas.

Para la base de datos implementada se debe desarrollar por lo menos 6 diferentes consultas (o sea, no la misma consulta referente a diferentes objetos espaciales) que permiten analizar los datos espaciales, convencionales y combinación de ambos. Estas consultas deben usar como mínimo las relaciones topológicas, expresiones de *group by* con funciones de agregación, *centroid*, búfer (o c*onvex hull*) y vecino(s) más cercano(s). En esta parte se evaluará la propuesta de presentar diferentes posibilidades de análisis como también la complejidad y eficiencia de código de consultas.

Además, en muchos sistemas de bases de datos reales es necesario considerar diferentes elementos para mejorar el desempeño del sistema tanto referentes directamente a la consulta como a la configuración del sistema. Para los propósitos de este proyecto, solo se considera la parte correspondiente a modificaciones de consultas y parámetros de ejecución para analizar su efecto en los accesos a la memoria, tiempo de ejecución y cambio en planes de ejecución. Nota que no se propone optimizar las consultas debido a la necesidad de un trabajo muy extenso para obtenerlo, sino lo que se quiere es ver los efectos de cambio de parámetros de ejecución al modificar la consulta.

De acuerdo a la aplicación desarrollada, se debe elaborar varias pruebas (especificadas a continuación), donde se utilizan las consultas con diferentes transformaciones. Por ejemplo, se puede evaluar el efecto que tiene ausencia o presencia de un índice sobre algún campo de la consulta. Para esto deben ejecutar la misma consulta con y sin índice. Para cada ejecución de la consulta debe considerar:

- Tiempo de ejecución.
- Lecturas lógicas.

- Lecturas físicas.
- Planes de ejecución.

En el Anexo 2 se encuentra un breve resumen de las opciones existentes en SQL Server 2008 para la verificación de estos parámetros.

Los aspectos de evaluar en las consultas son los siguientes:

1. La relación topológica entre objetos espaciales con y sin índices. Realizar por lo menos tres diferentes pruebas comparativas sin uso de ningún índice espacial (no lo crean o borran si ya lo tienen) y con diferentes variantes para este índice (diferentes resoluciones, por ejemplo, *low*, *medium* y *high*). Si SQL Server no selecciona el índice espacial en su plan de ejecución, fuerza su uso (ver detalles en el Anexo 2). Selecciona las tablas que tienen mayor número de elementos.
2. Consulta que abarca por lo menos dos tablas con datos espaciales usando tres variantes:
    a. *Join* basado en llaves foráneas.
    b. *Join* basado en llaves foráneas y relaciones topológicas.
    c. Join basado solamente en relaciones topológicas.
3. La ampliación de la operación *join* de la consulta 2a forzando el uso de *nested loop join*, *merge* y *hash joins*.
4. Uso de *in* versus *or*.

En toda la implementación se considerará la eficiencia de la implementación (por ejemplo, no recorrer la misma tabla muchas veces, no hacer copias de datos de Excel sino buscar formas más automáticas de transformación), adecuado uso de SQL y el conocimiento sobre el uso del lenguaje de programación de la DBMS seleccionada.

<u>Documentación (30%)</u>

La documentación debe incluir (en el orden especificado de forma organizada - la parte de evaluación):

1. La especificación y justificación de los cambios realizados al esquema conceptual desarrollado en la primera etapa del proyecto (si aplica).
2. Por cada tabla:
    a. El código o pantallazo de la creación de tabla con todas las restricciones indicadas en el esquema lógico.
    b. El código del (de los) disparador(es) (*trigger(s)*).
    c. Breve descripción de las correspondencias entre atributos de la tabla creada, archivos shape y archivos de datos convencionales y de las transformaciones necesarias para la limpieza y corrección de datos.
    d. El código de inserción de datos con las transformaciones especificadas en el punto anterior.

e. Código de creación de índices o especificación de los parámetros usados (entiendo su significado). Los parámetros (especialmente el *bounding box*) tienen que ser adecuados para la extensión espacial de datos usados.
f. Breve referencia a problemas encontrados.

3. Código y los resultados de cada una de las consultas elaboradas para análisis.
4. Una tabla comparativa que debe incluir las consultas y los resultados de tiempo de ejecución, lecturas lógicas, lecturas físicas y los planes de ejecución (para todo es posible usar los pantallazos). Además, deben incluir la **interpretación** (no descripción) de los resultados de cada experimento tomando en cuenta tanto los datos de la tabla comparativa como los planes de ejecución. Se evaluará si la presentación de las conclusiones está expresada en forma clara y concisa.

Ejemplo de la posible presentación de los resultados (una parte):

Tabla Provincias

<table>
<tr><td>Creación de la tabla</td><td colspan="3">

```
CREATE TABLE [dbo].[ElkaProvincia](
        [Codigo] [bigint] NOT NULL,
        [Nombre] [nchar](255) NULL UNIQUE,
        [Geometria] [geometry] NULL,
 CONSTRAINT [PK_ElkaProvincia] PRIMARY KEY CLUSTERED [Codigo] ASC)
```

</td></tr>
<tr><td>Disparador(es)</td><td colspan="3">

```
CREATE TRIGGER CorrigaGeom]
  ON  ElkaProvincia
  AFTER INSERT, UPDATE
AS
BEGIN
  DECLARE @GeomCheck Geometry,
     @Nombre Nchar(10);

  SELECT @Nombre = Nombre, @GeomCheck = Geometria
  FROM INSERTED
  IF @GeomCheck.STIsValid()=0
  SET @GeomCheck = @GeomCheck.MakeValid();

  UPDATE DatosEspa
  SET Geometria = @GeomCheck
  WHERE Nombre = @Nombre
END
```

</td></tr>
<tr><td rowspan="3">Descripción de las correspondencias y transformaciones</td><td>Nombre de la tabla
Provincia</td><td>Archivo shape:
provincias2008crtm05</td><td>Descripción</td></tr>
<tr><td>Nombre</td><td>Provincia</td><td>Nombre de la provincia; se mantiene</td></tr>
<tr><td>Codigo</td><td>Cod_Prov</td><td>Código de la provincia</td></tr>
</table>

	CantidadViviendas	Viviendas_1, Viviendas_2, Viviendas_3	Se suma la cantidad de diferentes tipos de vivienda para formar una sola cantidad
	Geometria	Sin nombre	Se aplica las transformaciones para asegurar que las geometrías sean válidas.
Código de la inserción con transformaciones, limpieza y correcciones de geometrías	INSERT INTO Provincias (Nombre, Codigo, CantidadViviendas, Geometria) SELECT …		
Código de la creación de índices (incluyendo la verificación de *bounding box*) o la especificación de parámetros usados	CREATE INDEX …		
Problemas encontrados			

Consultas analíticas

- Consulta
- Breve explicación de la consulta
- Resultados (o su parte si más que 10 tuplas)

Comparación de ejecución de las consultas

Consultas	**Lecturas físicas**	**Lecturas lógicas**	**Tiempo de ejecución**
Consulta1			
Consulta1 modificada			
Plan de ejecución consulta1			
Plan de ejecución consulta1 modificada			
Interpretación			
Consulta2			
Consulta2 modificada			

Plan de ejecución consulta 2	
Plan de ejecución consulta2 modificada	
Interpretación	

Implementación y defensa (70%)

Durante la defensa los estudiantes deben preparar la presentación y ser capaces de responder a las preguntas del profesor del curso para la verificación de la implementación especificada en la documentación. Cada estudiante debe ser capaz de responder a las preguntas referentes a todas las etapas de desarrollo e implementación del proyecto. Además, cada grupo del proyecto evaluará la participación de los integrantes y distribuirá el porcentaje de participación de cada uno de ellos. Este porcentaje se considerará para la asignación de la nota de implementación.

Fecha de entrega de la documentación
17 de noviembre de 2014

Fecha de la defensa
20 de noviembre de 2014

SIEMBRA UNA ACCIÓN Y COSECHARÁS UN HÁBITO.
SIEMBRA UN HÁBITO Y CONSECHARÁS UN CARÁCTER.
SIMEBRA UN CARÁCTER Y COSECHARÁS UN DESTINO

William James.

ANEXO 1

BREVE INTRODUCIÓN A LAS CARACTERÍSTICAS ESPACIALES DE SQL SERVER 2008 LIMITADO A ASPECTOS IMPORTANTES PARA LA II ETAPA DEL PROYECTO (Basado en el libro Alastair Airchison "Beginning Spatial with SQL Server 2008", Apress)

Elaborado por Dra. Elzbieta Malinowski

TIPOS DE DATOS ESPACIALES

SQL Server 2008 permite manejar 2 tipos de datos espaciales:

1. Geography: datos tipo vector con coordinadas geodésicas.
2. Geometry: datos tipo vector con coordinadas planas (locales o proyectadas).

Estos dos tipos de datos se diferencian en varios aspectos como se puede ver en la siguiente tabla [1, pg. 34]

Property	geometry Datatype	geography Datatype
Shape of the earth	Flat	Round
Coordinate system	Projected (or natural planar)	Geographic
Coordinate values	Cartesian (x and y)	Latitude and longitude
Unit of measurement	Same as coordinate values	Defined in `sys.spatial_reference_systems`
Spatial reference identifier	Not enforced	Enforced
Default SRID	0	4326 (WGS 84)
Size limitations	None	No object may occupy more than one hemisphere
Ring orientation	Not significant	Significant

Es importante notar la restricción que actualmente tiene SQL Server permitiendo el uso de los datos espaciales que pertenecen solamente al mismo hemisferio; o sea, no se puede tener extensiones que abarcan los dos hemisferios.

MÉTODOS ASOCIADOS A LOS DATOS ESPACIALES

Existe una gran variedad de métodos y operaciones asociados a los datos espaciales de los dos tipos (geography y geometry). Los detalles de cada uno se pueden encontrar en los "SQL Server Books Online". Algunos de los ejemplos de los métodos que pueden ser útiles en el desarrollo del proyecto [1]:

Method	Description	geometry	geography
Describing a Geometry			
`STGeometryType()`	Returns the name of the type of geometry (e.g., Point, LineString)	•	•
`InstanceOf()`	Tests whether an instance is of a particular geometry type	•	•
`STDimension()`	Returns the number of dimensions an instance occupies	•	•
`STIsSimple()`	Determines whether an instance is simple	•	
`STIsClosed()`	Determines whether an instance is closed	•	
`STIsRing()`	Determines whether an instance is a ring	•	
`STNumPoints()`	Returns the number of points in an instance	•	•
`STIsEmpty()`	Determines whether a geometry is empty (i.e., contains no points)	•	•
Returning the Coordinates Values of a Point			
`STX`	Returns the x coordinate of a `geometry` Point instance	•	
`STY`	Returns the y coordinate of a `geometry` Point instance	•	
`Lat`	Returns the latitude of a `geography` Point instance		•
`Long`	Returns the longitude of a `geography` Point instance		•
`M`	Returns the m (measure) coordinate of a Point instance	•	•
`Z`	Returns the z (elevation) coordinate of a Point instance	•	•

Returning Individual Points from a Geometry

`STPointN()`	Returns the *n*th point in the definition of an instance	•	•
`STStartPoint()`	Returns the first point in the definition of an instance	•	•
`STEndPoint()`	Returns the last point in the definition of an instance	•	•
`STCentroid()`	Returns the geometric center point of a Polygon instance	•	
`EnvelopeCenter()`	Returns the envelope center point of a `geography` instance		•
`STPointOnSurface()`	Returns an arbitrary point from the interior of an instance	•	
`STLength()`	Returns the length of all lines in an instance	•	•
`STArea()`	Returns the area contained by an instance	•	•
`STSrid`	Sets or retrieves the SRID of the spatial reference system in which an instance is defined	•	•

Handling Rings of a Polygon

`STExteriorRing()`	Returns the exterior ring of a Polygon instance	•	
`STNumInteriorRing()`	Returns the number of interior rings in a Polygon instance	•	
`STInteriorRingN()`	Returns the specified interior ring of a Polygon instance	•	
`NumRings()`	Returns the number of rings in a `geography` Polygon instance		•
`RingN()`	Returns a specific ring from a `geography` Polygon instance		•

Describing the Extent of a Geometry			
`STBoundary()`	Returns the boundary of an instance	•	
`STEnvelope()`	Returns the envelope (bounding box) of a geometry instance	•	
`EnvelopeAngle()`	Returns the angle between the center of a geography instance and the most outlying point		•
Working with Multielement Geometries			
`STNumGeometries()`	Returns the number of geometries in an instance	•	•
`STGeometryN()`	Returns a specific geometry from a multielement geometry	•	•

Modifying Spatial Objects

`MakeValid()`	Modifies a geometry to ensure it is valid	•	
`STUnion()`	Defines an instance from the union of two geometries	•	•
`STIntersection()`	Defines an instance from the intersection of two geometries	•	•
`STDifference()`	Defines an instance from the points representing the difference between two geometries	•	•
`STSymDifference()`	Defines an instance from the symmetric difference of two geometries	•	•
`Reduce()`	Defines a simplified instance of a Polygon or LineString	•	•
`STBuffer()`	Defines a Polygon or MultiPolygon representing the buffer zone around an object	•	•
`BufferWithTolerance()`	Defines a Polygon or MultiPolygon representing the buffer zone around an object with a given tolerance	•	•
`STConvexHull()`	Creates a Polygon representing the convex hull of a geometry	•	

Testing Relationships Between Spatial Instances

`STEquals()`	Determines whether two instances are composed of the same point set	•	•
`STDistance()`	Calculates the shortest distance between two instances	•	•
`STIntersects()`	Determines whether two instances intersect	•	•
`Filter()`	Tests whether two instances intersect based on a spatial index	•	•
`STDisjoint()`	Determines whether two instances are disjoint	•	•
`STTouches()`	Tests whether two instances touch	•	
`STOverlaps()`	Determines whether two instances overlap	•	
`STCrosses()`	Determines whether one instance crosses another	•	
`STWithin()`	Determines whether one instance is within another	•	
`STContains()`	Determines whether one instance contains another	•	
`STRelate()`	Tests whether two instances exhibit a given relationship specified using the DE-9IM model	•	

INSERCIÓN DE DATOS ESPACIALES

Los datos espaciales se pueden insertar de varias formas diferentes, por ejemplo:

1. Especificando la geometría en el formato de WKT incluyendo sus coordenadas (algo parecido a lo que hicimos en Oracle en el laboratorio 1) con la invocación del método adecuado, por ejemplo,

```
INSERT INTO Rutas(Nombre, Compania, Geometria)
VALUES(  'Grande',  'Buses Unidos', geometry::STGeomFromText('LINESTRING(
            6 2,8 8,18 8,24 12,15 16,11 18,6 17,2 12 )', 0));
```

2. Creando la geometría en WKB (ver el proceso en [1]) con la invocación del método adecuado.
3. Usando la representación GML (ver el proceso en [1]) con la invocación del método adecuado.
4. Usando el software Shape2Sql.exe (http://www.sharpgis.net/page/SQL-Server-2008-Spatial-Tools.aspx)

Se recomienda el uso de la última herramienta para el desarrollo del proyecto ya que en forma fácil permita tener datos de los archivos tipo *shape* en las tablas de SQL Server 2008 facilitando la manipulación y transformaciones requeridas para el desarrollo exitoso del proyecto. Este software al ejecutarse primero solicita indicar los parámetros de la conexión con el servidor SQL Server 2008 y el nombre de la base de datos donde se requiere crear la tabla e insertar los datos como se puede ver en la figura 1.

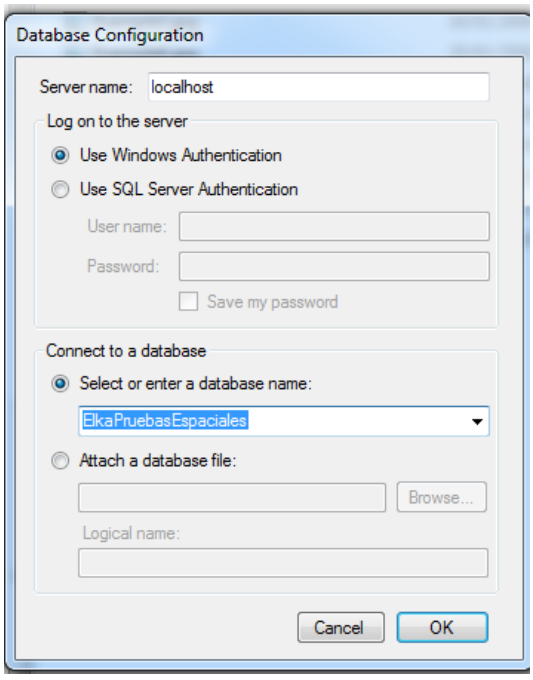


Figura 1. Establecimiento de parámetros de configuración para la herramienta Shape2Sql.

Posteriormente se debe especificar la ubicación y el nombre del archivo tipo *shape* como también las propiedades de los datos espaciales (su tipo). Además, se puede seleccionar de cargar solo los atributos de interés en lugar de todos los atributos incluidos en el archivo tipo *shape* como se puede observar en la figura 2.

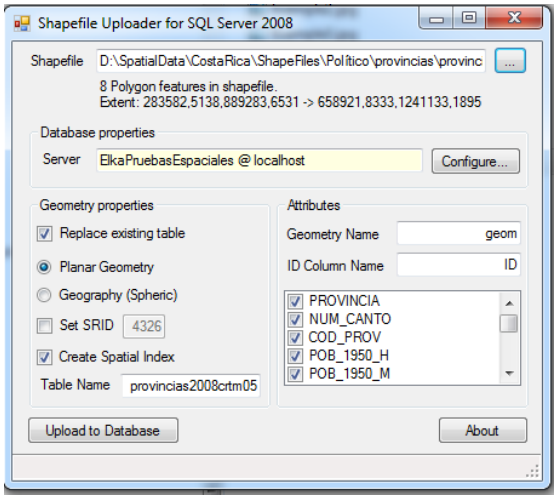


Figura 2. La definición de los parámetros para transformar el archivo shape a formato de SQL Server.

CORRECCIÓN DE DATOS ESPACIALES

Desafortunadamente no toda la información presente en los archivos tipo shape está formada correctamente y debido a que SQL Server 2008 primero revisa las geometrías antes de poder trabajar sobre ellas (índices, consultas), en algunos casas es necesario "corregirlas". Está "corrección" no se trata de transformaciones que tienen que hacer para que los archivos tipo shape queden en tercera forma normal (por ejemplo, crear geometrías compuestas para evitar la repetición de la información).

La descripción detallada como "corregir" estas geometrías, la pueden encontrar en http://beginningspatial.com/fixing_invalid_geography_data. Algunos ejemplos realizados sobre el archivo tipo shape que contiene la información de provincias:

1. Consultar sobre la validez de las geometrías:

```
SELECT Nombre, geometria.STIsValid()
FROM ElkaProvincia
```

	Nombre	(No column name)
1	SAN JOSE	1
2	ALAJUELA	1
3	CARTAGO	0
4	HEREDIA	1
5	GUANACASTE	0
6	PUNTARENAS	0
7	LIMON	0

2. Corregir las geometrías

```
UPDATE ElkaProvincia
SET Geometria = Geometria.MakeValid()
```

	Nombre	(No column name)
1	SAN JOSE	1
2	ALAJUELA	1
3	CARTAGO	1
4	HEREDIA	1
5	GUANACASTE	1
6	PUNTARENAS	1
7	LIMON	1

3. Corregir la orientación del ring

```
UPDATE ElkaProvincia
SET geometria = geometria.STUnion(geometria.STStartPoint())
```

4. Corregir la precisión (los picos) de las coordenadas.

```
UPDATE ElkaProvincia
SET geometria = geometria.STBuffer(0.00001).STBuffer(-0.00001)
```

CONSULTAS E ÍNDICES ESPACIALES

Algunas consultas sobre objetos espaciales en SQL Server 2008 se pueden realizar usando el modelo de dos pasos aplicando el primer filtro sobre los índices espaciales (si existen) y posteriormente el filtro secundario para las geometrías candidatas seleccionadas en el primer paso. El índice usado por SQL Server se basa en el concepto de *quadtree* (visto en clases) y usa las estructuras de árboles B+ para su implementación. En "SQL Server Books Online" se puede encontrar una explicación más detallada sobre la estructura de este índice y los parámetros usados para su definición. Uno de los parámetros solicitados en *bounding box* que representa el marco espacial sobre el cual se realizará el proceso de división en celdas (*teselletion process*) tesselación.

Los índices espaciales se pueden crear usando el comando CREATE INDEX (ver "SQL Server Books Online" para los detalles de sintaxis) o la interface gráfica de Server Management Studio como se presentan en las figuras 3 y 4.

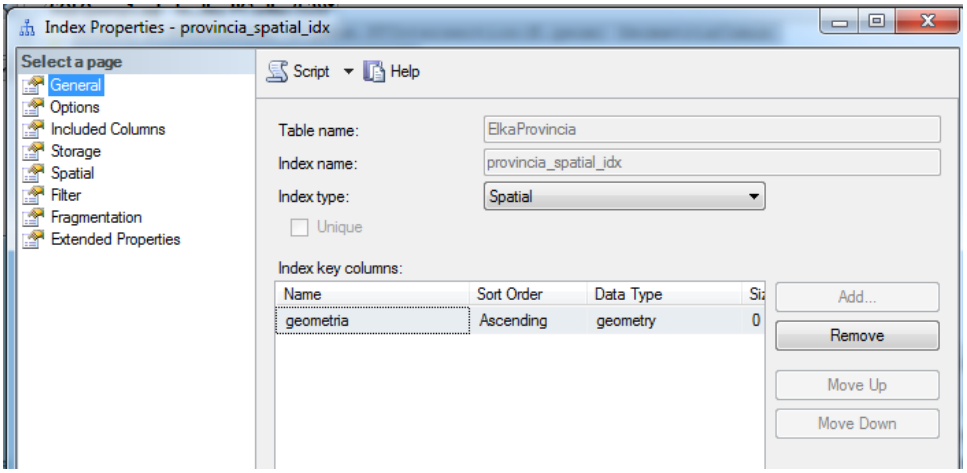


Figura 3. Creación del índice espacial.

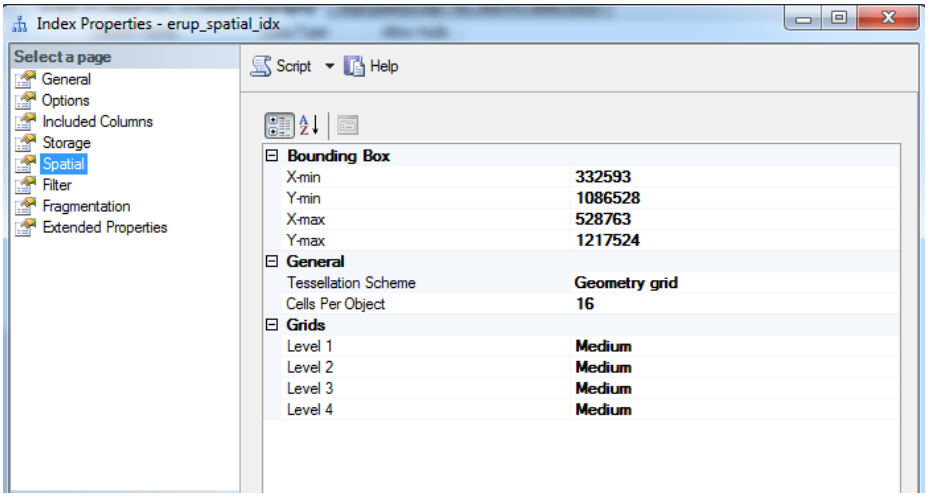


Figura 4. Especificación de parámetros para el índice espacial.

Las consultas que son candidatos a la ejecución por medio de dos filtros deben tener especificada en la cláusula WHERE algunos de las siguientes métodos: Filter, STContains, STDistance, STEquals, STIntersects, STOverlaps, ST Touches, STWithin, por ejemplo, WHERE obj1.STWithin(obj2) = 1.

DESPLIEGUE DE LOS RESULTADOS

SQL Server cuenta con las facilidades de despliegue de los resultados espaciales en el propio Server Management Studio. Cuando se solicita la información espacial, aparece una pestaña nueva donde se presenta el mapa resultante. Este mapa tiene funciones muy limitantes de zoom y etiquetas, aunque pueden ser suficientes para la verificación de los resultados de las consultas desarrolladas en el proyecto. Por ejemplo, se puede solicitar la información usando solo una tabla de provincias (consulta 1 con los resultados en las figuras 5 y 6) o aplicar algunas relaciones topológicas (la intersección de las provincias y zonas de peligro de erupciones) (consulta 2 con los resultados en las figuras 7 y 8):

Consulta 1: SELECT * FROM provincias

Results | Spatial results | Messages

	PROVINCIA	NUM_CANTO	COD_PROV	POB_1950_H	POB_1950_M	POB_1963_H	POB_1963_M	POB_1973_H	POB_1973_M
1	NA	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2	SAN JOSE	20	1	135100	146700	235600	252100	337200	358000
3	ALAJUELA	15	2	74200	74600	121100	119600	164600	161400
4	CARTAGO	8	3	50800	49900	79200	76200	103800	100900
5	HEREDIA	10	4	25300	26400	42000	43100	66500	67300
6	GUANACASTE	11	5	45100	43100	72700	69900	91400	87300
7	PUNTARENAS	11	6	47400	40800	82300	74200	113600	104600
8	LIMON	6	7	21900	19500	36000	32300	61400	53800

Figura 5. Resultados tabulares de la consulta.

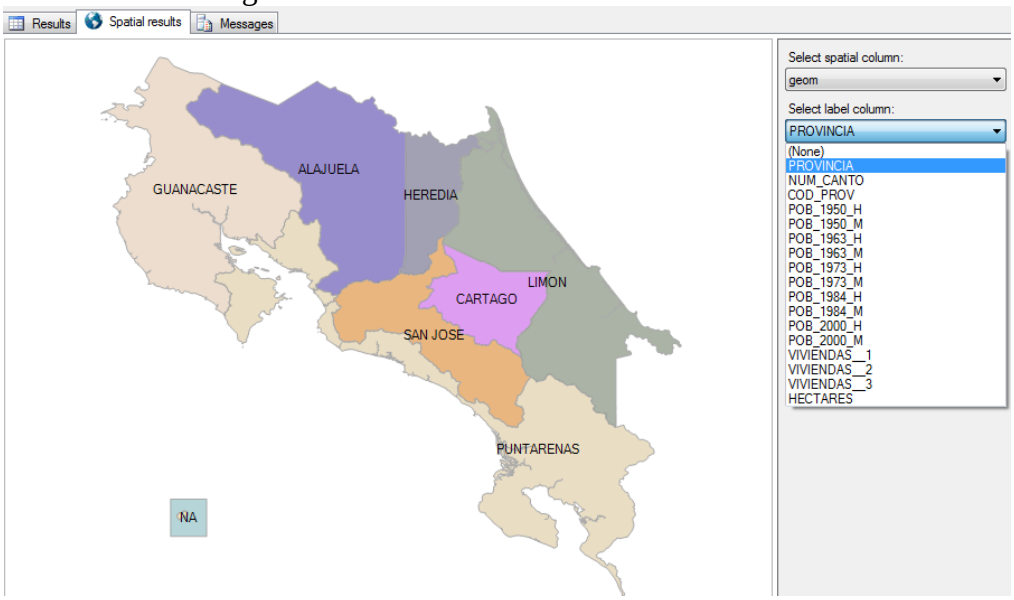


Figura 6. Resultados espaciales presentes en el mapa.

Consulta 2:
```
SELECT P.Provincia, P.geom.STIntersection(E.geom) GeometriaComun
FROM Provincias P, Erupciones E
WHERE P.geom.STIntersects(E.geom) = 1
```

Results | Spatial results | Messages

	nombre	GeometriaComun
1	ALAJUELA	0x000000000104870600000C68CC91E043618418E0EBE5D5D...
2	GUANACASTE	0x000000000104BE06000000000078581F184100000087E6CF...
3	ALAJUELA	0x000000000104640000000000090933A1941000000C393B3...
4	GUANACASTE	0x000000000104 9F000000000000C0E0B81841000000EA3B6...
5	PUNTARENAS	0x000000000104330000023F187648DE9184117A6B5A24F6...
6	ALAJUELA	0x000000000104 D000000000000007CBF1C4100000040FCD...
7	HEREDIA	0x000000000104 E0000002897642 1925C1D41CB1D536B09E...
8	SAN JOSE	0x000000000104C40200000000004E82461E410000005F4094...
9	CARTAGO	0x000000000104D201000024F8B76891511E41F513B2AF8994...
10	HEREDIA	0x000000000104F8000000000000BB86C11D4100000001E1C...

Figura 7. Resultados tabulares de la consulta.

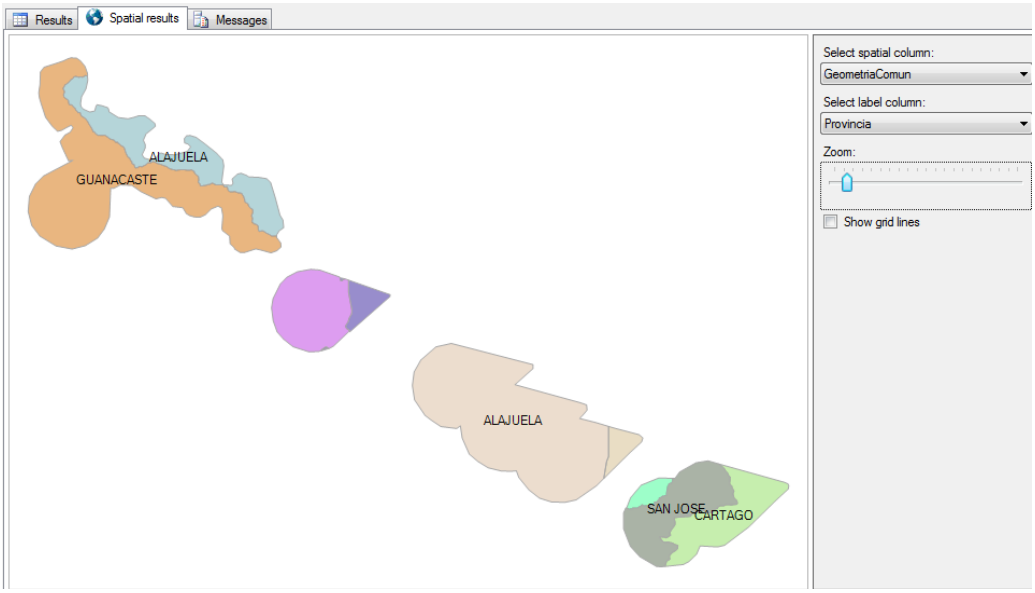


Figura 8. Resultados espaciales presentes en el mapa.

Como otra alternativa se puede usar otra herramienta llamada SQL Spatial que también aparece en el sitio http://www.sharpgis.net/page/SQL-Server-2008-Spatial-Tools.aspx. Esta herramienta permite expresar las consultas SQL y obtener su despliegue en mapa incluyendo toda información convencional solicitada (ver la figura 9).

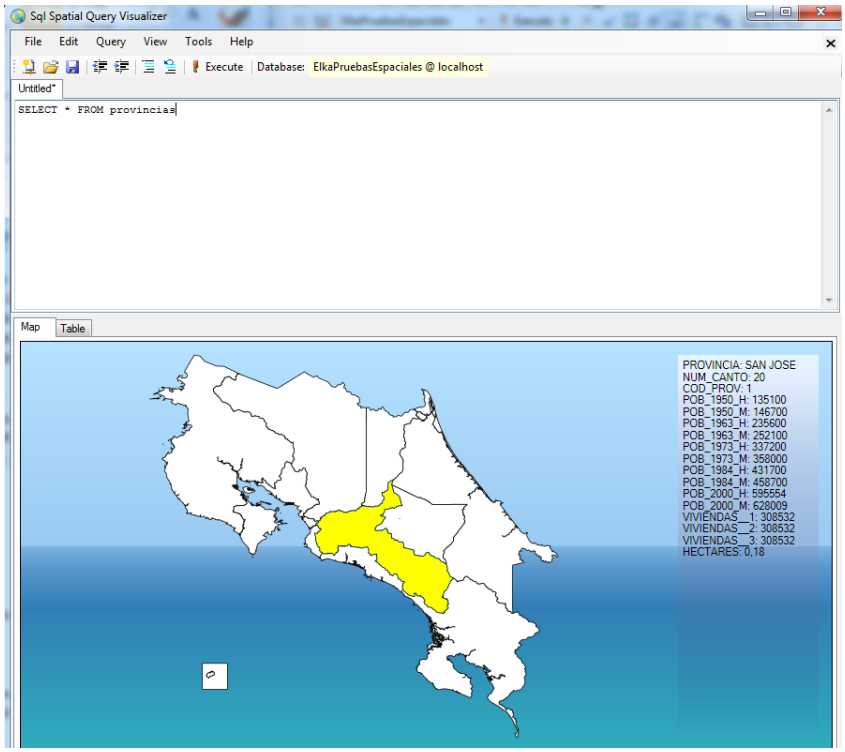


Figura 9. Resultados espaciales presentes en la herramienta Sql Spatial.

En caso necesario, esta herramienta también permite la exportación de los resultados al archivo tipo shape (ver la figura 10):

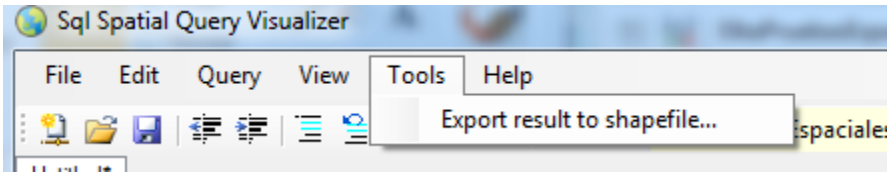


Figura 10. La opción de exportar tablas a archivo shape usando Sql Spatial.

BIBLIOGRAFÍA

1. Alastair Airchison "Beginning Spatial with SQL Server 2008", Apress, 2009

ANEXO 2

PROCESAMIENTO Y OPTIMIZACIÓN DE CONSULTAS EN SQL SERVER

Elaborado por Dra. Elzbieta Malinowski

PROCESO DE COMPILACIÓN

El procesamiento de consultas en SQL Server parecido a otros DBMSs incluye varias etapas que se pueden agrupar en dos bloques de compilación y optimización. La compilación y optimización son dos actividades separadas y difieren en su contenido. La compilación en SQL Server se realiza en varias fases (los nombres de las fases son propias a SQL Server):

- *Parsing*: transforma la consulta en la forma entendible por el compilador haciendo también la revisión de sintaxis. En esta etapa no se detecta la validez de los nombres de la tablas o columnas.
- *Normalization*: determina las características de los objetos referenciados en la consulta y revisa los aspectos semánticos.
- *Compilation*: se crea "el árbol de secuencia" que incluye la secuencia de todas las operaciones requeridas para la ejecución de la consulta. Durante este proceso también se carga todos los metadatos y realiza las conversiones entre datos, si son requeridas. Para las sentencias de DML se crea a "Query graph".

PLAN Y ESTADÍSTICAS DE EJECUCIÓN

SQL Server ofrece varias alternativas de ver plan de ejecución y estadísticas de ejecución de la consulta.

- *STATISTICS IO*: devuelve información estadística relacionada con la cantidad de actividad de disco generada por una consulta. La salida de datos incluye el número de páginas leídas de la caché (lecturas lógicas), el número de páginas leídas del disco (lecturas físicas), el número de páginas llevadas a la caché por la consulta (lecturas anticipadas) y el número de exámenes de índice o tabla realizados (recuento de exploraciones)
- *STATISTICS TIME*: muestra el número de milisegundos necesarios para analizar, compilar y ejecutar cada instrucción. Los tiempos se han separado en dos partes, el tiempo necesario para analizar y compilar la consulta, y el tiempo necesario para ejecutar la consulta.
- *STATISTICS PROFILE*: muestra la información detallada de una instrucción.
- *STATISTICS XML*: genera información detallada sobre cómo se ejecutaron las instrucciones en un documento XML definido.

Estas opciones se pueden activar por medio de las sentencias como SET STATISTICS IO ON, SET STATISTICS TIME ON, etc. antes de ejecutar la consulta editándolas en la pantalla de consulta o estableciéndolas como parámetros en el menú *Query* → *Option* → *Advanced*.
Además, en SQL Server existen 3 maneras de recuperar un plan de ejecución para una consulta:

- **Plan gráfico** (ver figura 1): esta es una opción bastante atractiva visualmente, donde cada operación lógica o física se representa por un icono formando una estructura de árbol. Las flechas muestran el

flujo de datos entre las operaciones. Los planes de ejecución gráficos deben leerse de abajo arriba y de derecha a izquierda. Si se coloca el puntero sobre uno de los iconos, es posible visualizar información detallada de la operación, incluyendo la descripción y datos estadísticos como el número de filas generado por cada operación, el tamaño promedio de cada fila y el costo de su ejecución. Estas características ofrecen la solución más conveniente para el examen directo y constituyen, sin duda, el enfoque usado con más frecuencia para mostrar y analizar los planes de ejecución. Este plan de ejecución se puede ver seleccionado en Query Editor, el botón de *Display Estimated Execution Plan* y *Include Client Statistics.*

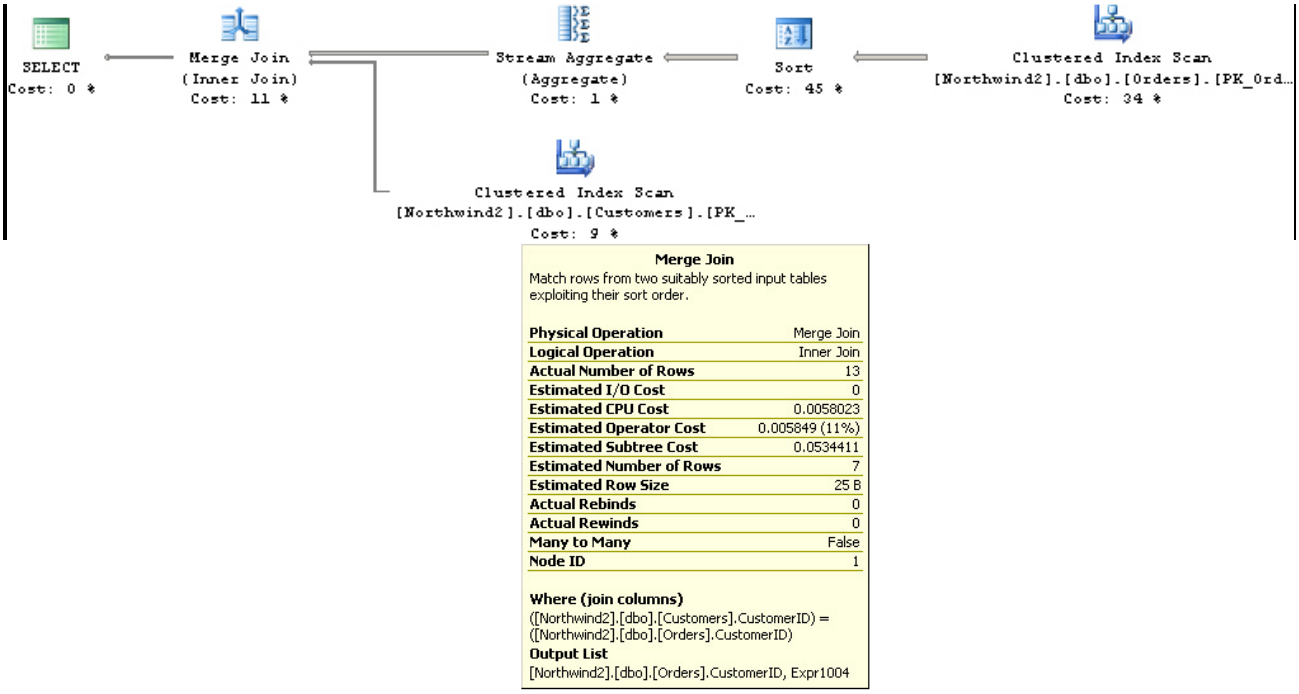


Figura 1. Plan gráfico

- **Plan basado en texto** (ver figura 2): en esta opción cada operación se representa como una línea separada; se utiliza la identación y barras verticales para mostrar la relación entre elementos padres e hijos. No se utilizan explícitamente flechas, pero los datos siempre fluyen de los elementos hijos a los padres. Esta opción se puede obtener por medio del comando SET SHOWPLAN_TEXT ON o SET STATISTICS PROFILE ON.

```
|--Merge Join(Inner Join, MERGE:([O].[CustomerID])=([C].[CustomerID]), RESIDUAL:(...))
     |--Stream Aggregate(GROUP BY:([O].[CustomerID])
        DEFINE:([Expr1004]=MAX([O].[Freight])))
     |    |--Sort(ORDER BY:([O].[CustomerID] ASC))
     |         |--Clustered Index Scan(OBJECT:([Orders].[PK_Orders] AS [O]))
     |--Clustered Index Scan(OBJECT:([Customers].[PK_Customers] AS [C]),
        WHERE:([C].[Country]=[@Country]) ORDERED FORWARD)
```

Figura 2. Plan basado en texto

- **Plan basado en XML** (ver figura 3): esta opción es nueva en SQL Server 2005, reúne muchas de las mejores características de los planes de texto y gráficos. Brinda la facilidad de anidar elementos XML, lo que hace mucho más natural el árbol del plan de consulta que solamente texto puro. A pesar que este plan de ejecución puede no ser el formato más sencillo para leer, esta opción permite escribir procedimientos y utilidades para analizar planes de ejecución, buscar signos de problemas de rendimiento y planes poco óptimos. Se puede obtener este plan de ejecución por medio del comando SHOWPLANXML.

```
<StmtSimple StatementText=
     "SELECT O.[CustomerId], MAX(O.[Freight]) as MaxFreight
     FROM [Customers] C JOIN [Orders] O
          ON C.[CustomerId] = O.[CustomerId
     WHERE C.[Country] = @Country
     GROUP BY O.[CustomerId]
     OPTION (OPTIMIZE FOR (@Country = N'UK'))"...>
  <StatementSetOptions QUOTED_IDENTIFIER="false" ARITHABORT="true"
                       CONCAT_NULL_YIELDS_NULL="false" ANSI_NULLS="false"
                       ANSI_PADDING="false" ANSI_WARNINGS="false"
                       NUMERIC_ROUNDABORT="false" />
  <QueryPlan DegreeOfParallelism="0" MemoryGrant="64" CachedPlanSize="15"
             CompileTime="20" CompileCPU="20" CompileMemory="280">
   <RelOp NodeId="1" PhysicalOp="Merge Join" LogicalOp="Inner Join"...>
    <Merge ManyToMany="0">
      <RelOp NodeId="2" PhysicalOp="Stream Aggregate" LogicalOp="Aggregate"...>
        <StreamAggregate>
          <RelOp NodeId="3" PhysicalOp="Sort" LogicalOp="Sort"...>
            <MemoryFractions Input="1" Output="1" />
            <Sort Distinct="0">
              <RelOp NodeId="4" PhysicalOp="Clustered Index Scan"
                                LogicalOp="Clustered Index Scan"...>
                <IndexScan Ordered="0" ForcedIndex="0" NoExpandHint="0">
                  <Object Database="[Northwind2]" Schema="[dbo]" Table="[Orders]"
                          Index="[PK_Orders]" Alias="[O]" />
                </IndexScan>
              </RelOp>
            </Sort>
          </RelOp>
        </StreamAggregate>
      </RelOp>
      <RelOp NodeId="8" PhysicalOp="Clustered Index Scan"
                        LogicalOp="Clustered Index Scan"...>
        <IndexScan Ordered="1" ScanDirection="FORWARD" ForcedIndex="0"
                   NoExpandHint="0">
          <Object Database="[Northwind2]" Schema="[dbo]" Table="[Customers]"
                  Index="[PK_Customers]" Alias="[C]" />
            <Predicate>
              <ScalarOperator ScalarString="[Northwind2].[dbo].[Customers].[Country]
                                            as [C].[Country]=[@Country]">
              </ScalarOperator>
            </Predicate>
         </IndexScan>
       </RelOp>
     </Merge>
    </RelOp>
    <ParameterList>
      <ColumnReference Column="@Country" ParameterCompiledValue="N'UK'"
                       ParameterRuntimeValue="N'USA'" />
    </ParameterList>
   </QueryPlan>
</StmtSimple>
```

Figura 3. Plan basado en XML

Para analizar el plan de ejecución es necesario entender claramente los operadores que se utilizan al momento de conformar este plan. Son muchos operadores y existen muchas maneras de combinarlos, por lo que es necesario consultar *Books online* para conocer los detalles.

COSTO ESTIMADO DE EJECUCIÓN

Con frecuencia las personas asumen que la información sobre el costo estimado de ejecución es un buen indicador de cuánto tiempo tardará la consulta en ejecutarse y que esta estimación les permite distinguir

los planes buenos de los que no lo son. Esto no siempre es verdad, ya que se trata de una estimación inexacta y podría estar equivocada; en ocasiones los planes con costos estimados más altos pueden ser mucho más efectivos en términos de CPU, E/S y del tiempo de ejecución, a pesar de que sus estimaciones sean más altas. Entonces se debe incluir las opciones de STATISTICS I/O y STATISTICS TIME en el análisis, para entender cuál es realmente el costo de la ejecución en términos de E/S y tiempo de la CPU.

ANÁLISIS DE PLAN DE EJECUCIÓN

Ejecuta la siguiente consulta en la BD AdventureWorks que calcula la cantidad total de pedidos realizados por cada cliente de Adventure Works y verifique su plan de ejecución usando la representación gráfica (ver figura 4).

```
SELECT c.CustomerID, SUM(LineTotal)
FROM Sales.SalesOrderDetail od
JOIN Sales.SalesOrderHeader oh
ON od.SalesOrderID=oh.SalesOrderID
JOIN Sales.Customer c ON oh.CustomerID=c.CustomerID
GROUP BY c.CustomerID
```

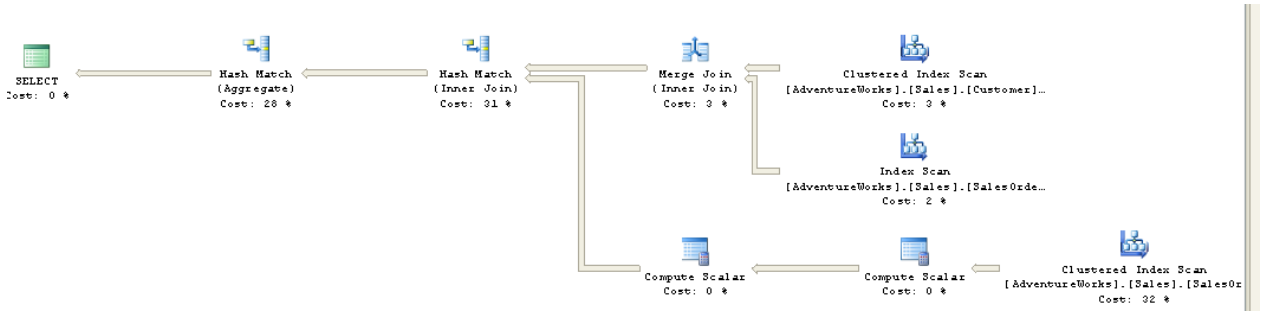


Figura 4. El plan de ejecución de la consulta especificada arriba.

Se puede analizar la secuencia de pasos realizados de la siguiente manera:

1. El motor de la base de datos realiza una operación Clustered Index Scan en la tabla Sales.Customer y devuelve la columna CustomerID para todas las filas de esa tabla.
2. A continuación, realiza un Index Scan en uno de los índices de la tabla Sales.SalesOrderHeader. Éste es un índice de la columna CustomerID, pero también incluye implícitamente la columna SalesOrderID por las particularidades del almacenamiento físico de la tabla. El examen devuelve los valores de ambas columnas.
3. El resultado de ambos exámenes se une en la columna CustomerID mediante el operador físico Merge Join.
4. A continuación, el motor de la base de datos realiza un examen del índice agrupado en la tabla Sales.SalesOrderDetail, recuperando los valores de cuatro columnas (SalesOrderID, OrderQty, UnitPrice y UnitPriceDiscount) de todas filas en esta tabla. Esta operación devuelve 123.317 filas

como se puede observar en las propiedades de número estimado de filas como tambien estimado tamaño de tupla y de datos en general.

5. Las filas producidas por el examen de índice agrupado se transmiten al primer operador Compute Scalar de manera que el valor de la columna calculada LineTotal pueda calcularse para cada fila, según las columnas OrderQty, UnitPrice y UnitPriceDiscount implicadas en la fórmula (verifique las propiedades de esta columna).
6. El segundo operador Compute Scalar aplica la función ISNULL al resultado del cálculo anterior, tal como lo requiera la fórmula de la columna calculada. Esto completa el cálculo de la columna LineTotal y lo devuelve, junto con la columna SalesOrderID, al operador siguiente (verifique las propiedades de esta columna para ver el uso de la función ISNULL).
7. El resultado del operador Merge Join del paso 3 se une con el resultado del operador Compute Scalar del paso 6, mediante el operador físico Hash Match.
8. A continuación, se aplica otro operador Hash Match para agrupar las filas devueltas de Merge Join por el valor de la columna CustomerID y el agregado calculado SUM de la columna LineTotal.
9. El último nodo, SELECT, no es un operador físico ni lógico, sino más bien un marcador de posición que representa los resultados de consulta generales y el costo.

Los porcentajes que se muestran bajo cada uno de los operadores representan el costo de cada operador individual en relación al costo general estimado de todo el plan de ejecución. Puede ver que la mayor parte del costo total de todo el plan de ejecución se asocia con los tres operadores siguientes: el Clustered Index Scan de la tabla Sales.SalesOrderDetail y los dos operadores Hash Match.

OPTIMIZACIÓN DE CONSULTAS

La segunda parte del procesamiento de consultas es su optimización. El objetivo de este proceso consiste en producir un plan de ejecución para la consulta específica o para un procedimiento almacenado que mejore su tiempo de ejecución. El plan producido después de la optimización especifica los pasos que deben tomarse para ejecutar la consulta e incluye la información variada sobre por ejemplo, el acceso a tabla, el uso de los índices, el tipo de operación para *join*, *group by* y *sort*, entre otros. La optimización se basa en costos donde se busca el plan de ejecución más baratos entre varias posibilidades. Para esto se necesita saber diferentes estadísticas de la base de datos. Figura 5 representa una esquema simplificada del proceso de optimización de consultas.

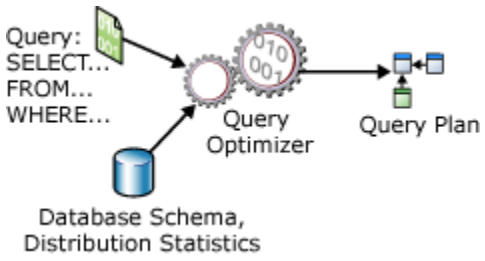


Figura 5. Una representación simplificada del proceso de optimización de consultas (SQL Server 2005 Books Online).

El optimizador contiene varias reglas de transformación que permiten hacer las permutaciones entre índices y las estrategias de *join* con el fin de buscar una opción barata, eso es la que usa menos recursos y se ejecuta más rápido. Debido a que el número de permutaciones puede ser alto, el optimizador de SQL Server no analiza todas las permutaciones posibles; en cambio solo selecciona éstos que producen soluciones baratas de acuerdo a *threshold* establecido en la configuración y límite de tiempo que se establece para búsqueda de las soluciones alternativas.

La optimización de consultas se realiza en varias etapas:

- **Análisis de la consulta**: el propósito es limitar el número de registros que deben ser escaneados. Se considera la cláusula WHERE y las condiciones (excepto de *join*) que contiene.
- **Selección de índices**. Para seleccionar los índices es necesario evaluarlos. El proceso de evaluación de basa en
    - Revisión de la existencia del índice: para la condición especificada en WHERE o HAVING se revisa la existencia de índices para su potencial uso. Se analiza los índices para cada cláusula de WHERE por separado y para los índices compuestos de varios atributos.
    - Estadísticas de índices: para poder seleccionar los índices se usa los criterios numéricos contenidos en las tablas del sistema SQL Server. Esta información puede ser desactualizada si los valores en las tablas se cambian frecuentemente. Para actualizarlos se debe ejecutar el comando de UPDATE STATISTICS para un índice específico o para las estadísticas creadas por medio de CREATE STATISTICS. Las últimas se usan en general para las columnas no-indexadas. La información estadística incluye mucha información. Para los índices uno de los datos importantes es el número de valores distintos para un atributo específico que permite calcular el factor de selectividad. Entre varios índices disponibles se selecciona éste que asegura la recuperación de menos registros.
    - El costo de índice: para asegurar que el uso de índice mejora la ejecución de la consulta es necesario calcular el número de páginas que se necesita leer con y sin uso de los índices y seleccionar la opción que minimice este número.
- **Selección de la operación de join**: se selecciona la estrategia de join que ofrece menor costo de ejecución basándose en varios factores, como el número de páginas leídas, la cantidad de memora requerida, entre otros. Se puede seleccionar una de las tres estrategias que ofrece SQL Server: *nested loop join*, *merge join* o *hash join* con sus variaciones. *Merge join* y *hash join* solo se pueden usar para la operación de *equijoin*. Note que la estrategia de seleccionar la operación de join puede llegar a ser muy compleja si participan muchas tablas.
- **Otras estrategias de procesamiento de operaciones**
    - GROUP BY: en versiones anteriores a 7.0 esta operación requería primero ordenamiento y después agrupación. De esta forma el resultado siempre se daba en forma ordenada. Actualmente, SQL Server puede usar una opción adicional de hash para organizar los grupos. De esta forma el resultado no se da en forma ordenada.
    - DISTINCT: para poder seleccionar los valores distintos en muchas ocasiones se aplica previamente el ordenamiento. De esta forma se puede obtener los valores ordenados sin especificar la cláusula ORDER BY. Si el optimizador encuentra las dos cláusulas (DISTINCT y ORDER BY) ignora la segunda.

- **Creación del plan de ejecución de la consulta**: basándose en el previo análisis y creación de planos alternativos de ejecución, se crea el plan de ejecución. Este plan se pasa a memoria caché para su ejecución. Para cada nueva consulta SQL Server revisa el texto de esta consulta para verificar si existe en la memoria caché su plan de ejecución. En caso afirmativo, lo usa. Se puede forzar la creación de nuevo plan de ejecución con la opción de recompilar que se debe aplicar especialmente para los procedimientos almacenados con parámetros que varían considerablemente o cuando las BD cambie significativamente.

Al principio de esta práctica se ejecutó la siguiente consulta y se obtuve el plan de ejecución como se muestra en la figura 6.

```
SELECT c.CustomerID, SUM(LineTotal)
FROM Sales.SalesOrderDetail od
JOIN Sales.SalesOrderHeader oh
ON od.SalesOrderID=oh.SalesOrderID
JOIN Sales.Customer c ON oh.CustomerID=c.CustomerID
GROUP BY c.CustomerID
```

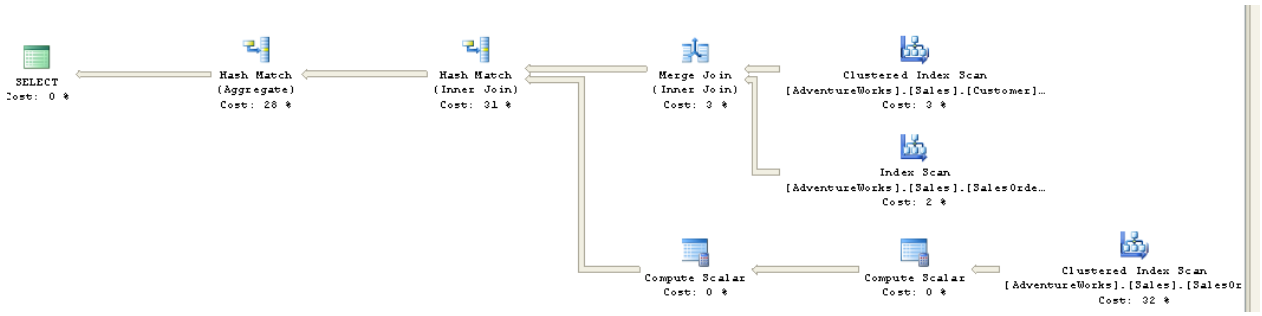


Figura 6. El plan de ejecución gráfico.

Nota que la única columna que devuelve la tabla Sales.Customer es CustomerID y esta columna se incluye también como una llave externa en Sales.SalesOrderHeaderTable. Por esta razón, se puede eliminar completamente la tabla Customer de la consulta sin cambiar el resultado producido como se muestra en la siguiente consulta (nota como se simplificó la consulta):

```
SELECT oh.CustomerID, SUM(LineTotal)
FROM Sales.SalesOrderDetail od JOIN Sales.SalesOrderHeader oh
ON od.SalesOrderID=oh.SalesOrderID
GROUP BY oh.CustomerID
```

Esta consulta tiene el plan de ejecución ilustrado en la figura 7:

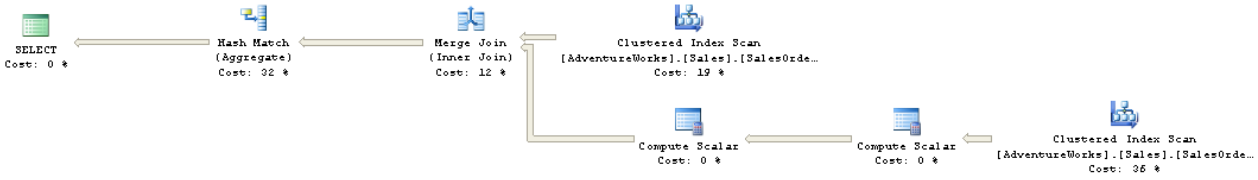


Figura 7. El plan de ejecución de la consulta modificada.

Como se puede ver, ocurrieron los siguientes cambios:

- Se eliminó Clustered Index Scan en la tabla Customer
- Se eliminó Merge Join entre Customer y SalesOrderHeader
- El join Hash Match se sustituyó por Merge Join, que es más eficaz.

El rendimiento de la consulta se puede mejorar. Se puede observar que el Clustered Index Scan de la tabla SalesOrderDetail se ha convertido en el operador más caro de este plan de ejecución. Puesto que sólo se necesita la columna de SalesOrderID para llevar a cabo la consulta, se puede crear un índice no agrupado que contenga sólo esa columna, reemplazando el examen de toda la tabla por uno del índice no agrupado, mucho más pequeño. La definición del índice:

```
CREATE INDEX IDX_OrderDetail_OrderID_TotalLine
ON Sales.SalesOrderDetail (SalesOrderID) INCLUDE (LineTotal)
```

El índice creado es un ejemplo del llamado "índice de cobertura". Se trata de un índice no agrupado que contiene todas las columnas necesarias para realizar la consulta (SalesOrderID y LineTotal), eliminando la necesidad de examinar toda la tabla mediante los operadores Table Scan o Clustered Index Scan. El índice es una copia más pequeña de la tabla, que contiene un subconjunto de columnas de la tabla. Sólo esas columnas necesarias para contestar la consulta (o consultas) se incluyen en el índice, en otras palabras, el índice contiene sólo lo que necesita para "cubrir" la consulta.

Después de crear el índice y ejecutar la consulta el plan de ejecución cambió al mostrado en la figura 8, mejorando la ejecución.

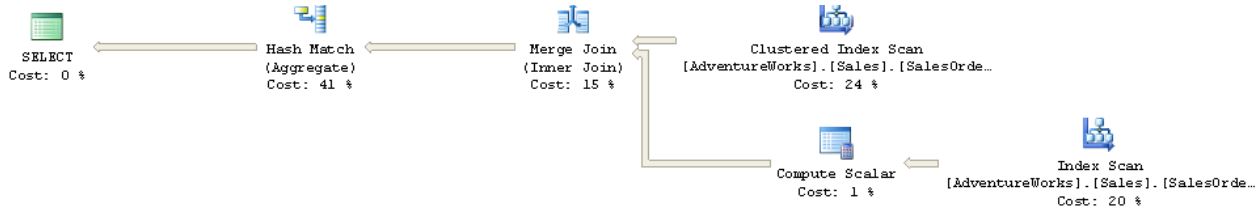


Figura 8. Plan de ejeución después de incluir el índice de cobertura.

Para modificar los planes de ejecución y buscar el "mejor" se puede usar sugerencias (hints) dentro de la consulta de SQL. En SQL Server las sugerencias se incluye o en la cláusula FROM (cunaod se quiere referir a un acceso específico para la tabla) o al final de la consulta. Algunos de las sugerencias útiles:

- WITH (INDEX(<Index_Name>)) se utiliza inmediatamente después de la cláusula FROM indicando un índice específico que se debe usar para la tabla.
- WITH (INDEX(0)) obliga usar el índice de segmento (si existe) en modo de escanear, mientras INDEX(1) en modo de buscar. Si índice de segmento no existe, solo index(0) se puede usar para realice el escaneo de la tabla.
- LOOP y HASH se usa solo en las consultas donde aparece la cláusula JOIN en FROM y permite indicar el tipo de join que se quiere usar para dos tablas. Si hay varias tablas con las operaciones de join, se pueden usar diferentes opciones para cada par de ellas.
- OPTION(LOOP JOIN) o OPTION(HASH JOIN) se pone al final de la consulta e indica el tipo de join usado para todas las tablas,
- OPTION(FORCE ORDER) parecido al parámetro visto antes donde se oblige al compilador realizar los joins en el orden en el cual aparecen las tablas en la cláusula FROM.

Se puede poner varias sugerencias separadas con comam por ejemplo, OPTION(LOOP JOIN, FORCE ORDER)

Ejemplos:

```
SELECT *
FROM Sales.Customer AS c
INNER JOIN Sales.CustomerAddress AS ca ON c.CustomerID = ca.CustomerID
WHERE TerritoryID = 5
OPTION (MERGE JOIN);

SELECT *
FROM Sales.Customer AS c
INNER MERGE JOIN Sales.CustomerAddress AS ca ON c.CustomerID = ca.CustomerID
WHERE TerritoryID = 5;

create index FK_ProductSubcategory
ON Production.Product(ProductSubcategoryID);

SELECT p.Name, s.Name
FROM Production.ProductSubcategory AS s
INNER JOIN  Production.Product AS p WITH (INDEX(FK_ProductSubcategory))
ON s.ProductSubcategoryID = p.ProductSubcategoryID
OPTION (LOOP JOIN, FORCE ORDER);
```

Algunos comandos útiles

EXEC sp_helpindex 'nombre de la table' permite ver todos los índices y su tipo.
CHECKPOINT: graba al disco todos los cambios hechos por tranasacciones.
DBCC DROPCLEANBUFFERS limpia el cache de los datos
DBCC FREEPROCCACHE limpia el cache de planes de ejecución
COMMIT confirmación de ejecución (se guardan los resultados en la base de datos)