

SALVADOR RAMOS

Data Warehouse, Data Marts Y Modelos Dimensionales

Un pilar fundamental
para La Toma de Decisiones



SolidQ

Analiza tu Negocio con Excel y Power BI
Aprende de tus datos

VOL. II

ADVERTENCIA LEGAL

Todos los derechos de esta obra están reservados a SolidQ™ Press.

El editor prohíbe cualquier tipo de fijación, reproducción, transformación o distribución de esta obra, ya sea mediante venta, alquiler o cualquier otra forma de cesión o comunicación pública de la misma, total o parcialmente, por cualquier sistema o en cualquier soporte, ya sea por fotocopia, medio mecánico o electrónico, incluido el tratamiento informático de la misma, en cualquier lugar del mundo.

La vulneración de cualquiera de estos derechos podrá ser considerada como una actividad penal tipificada en los artículos 270 y siguientes del Código Penal.

La protección de esta obra se extiende al mundo entero, de acuerdo a las leyes y convenios internacionales.

© SolidQ™ Press, 2016

Título:

Data Warehouse, Data Marts y Modelos Dimensionales.

Un pilar fundamental para la toma de decisiones

Autor: **Salvador Ramos**

ISBN: **978-84-940719-2-8**

Serie:

Analiza tu Negocio con Excel y Power BI.

Aprende de tus datos

SolidQ Global S.A.

Apartado de correos 202

03340 Albufera, Alicante, España

<http://www.solidq.com>

AUTOR



Soy **experto en BI & Analytics**
Director de Formación en SolidQ
SQL Server MVP desde el año 2003

¿En qué te puedo ayudar?

Quiero acompañarte en tu transición hacia el mundo del BI & Analytics y Big Data. Tanto si estás dando un giro a tu carrera profesional, como si buscas una mayor especialización en estas áreas.

Estoy especializado en Bases de datos, Data Warehousing, ETL, técnicas de Visualización y en el uso de tecnologías Microsoft (*SQL Server, Integration Services, Analysis Services, Reporting Services, Performance Point Services, Self-service BI, Excel, Power BI, Microsoft Azure*).

Te invito a que conozcas mi [blog personal](#) (pincha [aquí](#)), a que te [suscribas a él](#) para recibir tu regalo (pincha [aquí](#)) y estés al tanto de todas las novedades (*libros y artículos que publico, conferencias en las que participo, cursos y masters que imparto, material que regalo ...*).

Por confiar en mí, te ofrezco gratis mi libro gratuito “Microsoft Business Intelligence: Vea el cubo medio lleno”. Descárgalo [aquí](#).

Sígueme también en: <http://www.salvador-ramos.com>

Linkedin: <http://www.linkedin.com/in/SalvadorRamos>

Twitter: [@salvador_ramos](#)

SOLIDQ



SolidQ, desde el año 2002, suministra servicios para plataformas Microsoft que le ayudan a diseñar, integrar y optimizar su utilización de datos.

Combina una amplia experiencia técnica y de implementación en el mundo real, con un compromiso firme en la transferencia de conocimiento, dada la combinación única de dotes lectivas y experiencia profesional que nuestros mentores ofrecen. De este modo, no solamente ayudamos a nuestros clientes a solventar sus necesidades tecnológicas, sino que somos capaces de incrementar la capacidad técnica de sus profesionales, dándoles una ventaja competitiva en el mercado. Por eso llamamos **Mentores** a nuestros expertos: por su compromiso en asegurar el éxito de su empresa y de sus equipos profesionales a largo plazo.

Nuestros expertos son profesionales reconocidos en el mercado, con más de 100 premios Microsoft MVP (Most Valuable Professional) obtenidos hasta la fecha. Se trata de autores y ponentes en las conferencias más importantes del sector, con varios centenares de ponencias presentadas en conferencias nacionales e internacionales durante los últimos años.

Nuestra misión es la de **transmitir todo el conocimiento adquirido** resolviendo problemas del mundo real para miles de clientes, escribiendo artículos y libros, publicando whitepapers, creando contenidos educativos y formando a decenas de miles de trabajadores de TI en todo el mundo, para que los proyectos de nuestros clientes obtengan los mayores éxitos. Esta transferencia de conocimiento la realizamos fundamentalmente con dos tipos de servicios:

Consultoría: hazlo bien la primera vez (haz clic [aquí](#))

Mentoring: conoce tu potencial personal y mejora tus decisiones (haz clic [aquí](#))

Formación: la mejor inversión posible es pagar por el conocimiento de otros. Conoce nuestro **Plan Formativo** (haz clic [aquí](#)) y nuestro **Calendario** (haz clic [aquí](#))

Publicaciones: ponemos nuestros conocimientos a su alcance. Acceda a nuestros **blogs** (haz clic [aquí](#)) y a nuestras **publicaciones, la mayoría gratuitas** (haz clic [aquí](#))

MAPA MENTAL



ÍNDICE

ADVERTENCIA LEGAL	0
AUTOR.....	1
SOLIDQ	2
MAPA MENTAL.....	3
Conceptos generales de modelado dimensional.....	5
Data Warehouse y Data Marts	9
Esquema en estrella (<i>star schema</i>) y en copo de nieve (<i>snowflake schema</i>)	11
Modelado Dimensional	14
Trabajando con Hechos	16
Trabajando con Dimensiones	18
SCD (Slowly Changing Dimensions).....	19
Versiones de datos.....	23
Cerrando el ciclo de modelado	29
ENLACES ESENCIALES	34
GRACIAS	35

Conceptos generales de modelado dimensional

Habitualmente estamos acostumbrados a utilizar aplicaciones donde hay varios o muchos usuarios concurrentes, leyendo y escribiendo datos en sus bases de datos.

“Imagínate por ejemplo, un supermercado, donde muchas cajeras están atendiendo simultáneamente a los clientes, los encargados de almacén están introduciendo información de recepción de pedidos, en atención al cliente están creando una tarjeta de fidelización a un cliente y rellenando todos sus datos, en la oficina el departamento de administración está introduciendo facturas y contabilizándolas y los responsables de tienda están consultando información de ventas de ese momento para potenciar las ventas. Hay muchas personas leyendo y escribiendo información en esa base de datos, por ello debe tener un diseño que soporte ese acceso concurrente.”

Estas aplicaciones son sistemas transaccionales o sistemas OLTP (por sus siglas en inglés, OnLine Transaction Processing), y aunque constan de consultas e informes, no están diseñados para responder a las preguntas analíticas de los usuarios de negocio. Almacenan los datos en *Sistemas Gestores de Bases de Datos Relacionales (SGBDR)*, allí dichos datos están altamente normalizados, optimizados para concurrencia de muchos usuarios y de muchas escrituras y lecturas simultáneas.

Veamos en la siguiente imagen un ejemplo de tablas de una base de datos relacional normalizada:

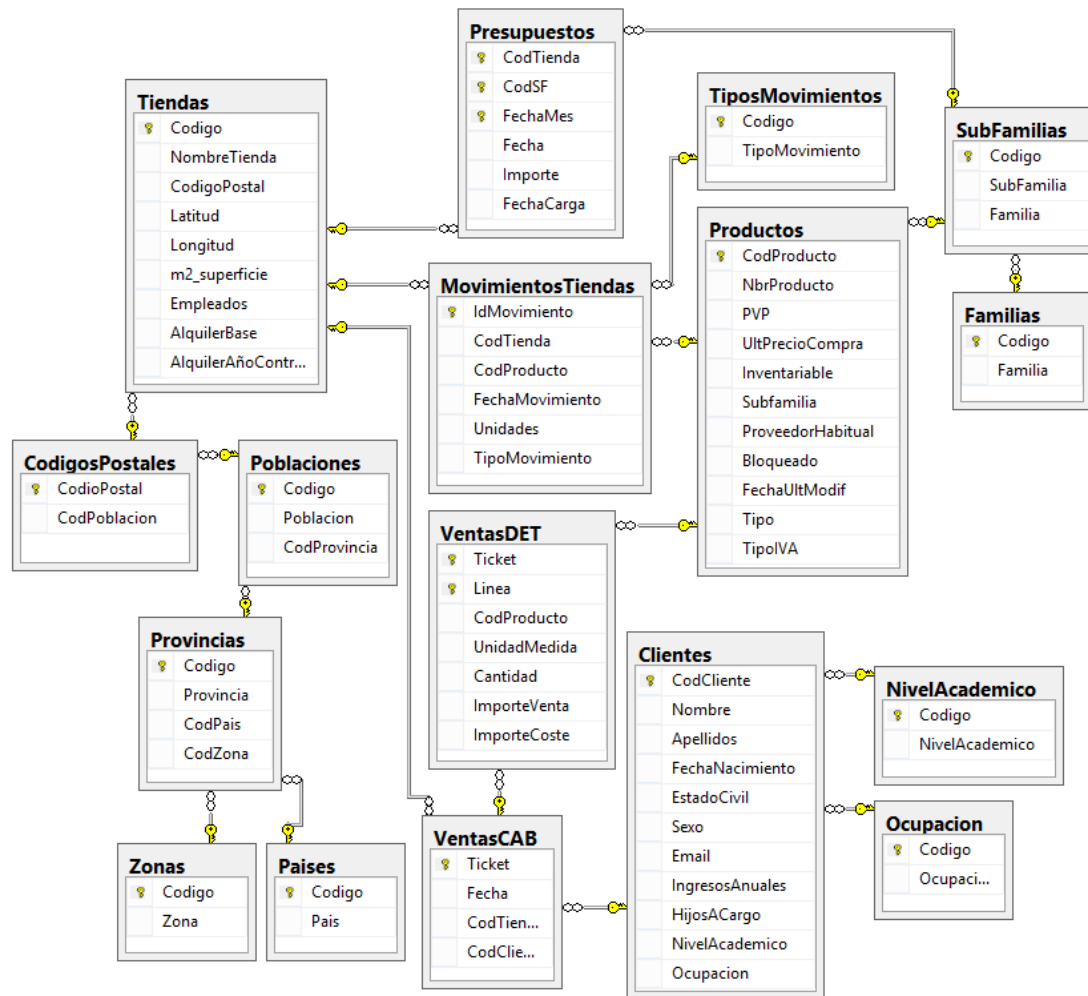


Figura 0-1 Sistema OLTP, Base de Datos Relacional normalizada



Encontrar estos esquemas normalizados es algo habitual y lógico para cualquier persona que haya estudiado sobre Bases de Datos Relacionales y diseño de sistemas transaccionales, pero quien no lo haya hecho, lo primero que hace cuando le muestran y explican un diagrama de este tipo es plantearse una serie de preguntas como:

- *¿Por qué la información del cliente está en 18 tablas y no toda en una sola tabla?*
- *¿Por qué no está toda la información de una factura en una misma tabla? Tengo ir a la tabla 'CabeceraFra' para ver datos del cliente, a 'DetalleFra' para ver unidades y productos vendidos', a 'PieFra' para ver detalles de impuestos, "vaya lío".*
- *Está todo disperso, y me decís que hay que unirlo cada vez que quiero consultarlo. ¿Por qué no está todo unido y nos evitamos estarlo uniendo continuamente para consultarlo?*
- *Lo tenéis todo dividido en procesos de negocio, muy desmenuzado, no entendéis que para los análisis necesitamos obtener una visión global.*
- *Entiendo que un informe de muchas páginas sea lento, Pero ¿Por qué van tan lentos algunos que no sacan más que unas poquitas líneas? Como ocurre con el informe de comparativa de ventas anuales, que tan sólo ocupa 7 líneas (una línea por año), y 4 columnas (ventas año actual, ventas año anterior, diferencia y % diferencia).*

La respuesta está en que estos sistemas transaccionales están enfocados a gestionar un gran número de transacciones concurrentes, hay mucha gente escribiendo simultáneamente, y estos además escriben pequeñas cantidades de filas (dar de alta un cliente, hacer un pedido de 14 líneas, modificar los datos de un proveedor, un apunte contable, etc.) y una poquitas personas consultando datos, además de que la mayoría de las consultas son operativas (¿cuánto llevamos vendido hoy?, ¿se ha enviado ya el pedido XXXXX?, etc.). Por tanto se centran en optimizar estas situaciones. Pero cuando alguien va a hacer un análisis a partir de información de la empresa, su situación ideal sería que nadie estuviese escribiendo y que la estructura estuviese optimizada para consultas (sólo lecturas de datos) y análisis de éstos.

Los sistemas analíticos están enfocados al análisis de grandes cantidades de información, proporcionando respuestas rápidas y complejas.

Veamos una comparativa en la siguiente tabla entre Transaccional y Analítico:

 Transaccional	 Analítico
<i>Orientado a lo operativo (procesos)</i>	<i>Orientado a temas</i>
<i>Predomina la actualización</i>	<i>Predomina la consulta. Datos históricos</i>
<i>Se accede a pocos registros</i>	<i>Procesos masivos, se accede a muchos registros</i>
<i>Datos altamente normalizados</i>	<i>Datos desnormalizados</i>
<i>Estructura relacional</i>	<i>Estructura In-Memory / multidimensional</i>
<i>Tiempos de respuesta muy rápidos (pocos registros)</i>	<i>Tiempos de respuesta rápidos (respuesta datos masivos)</i>
<i>Estructura estática</i>	<i>Estructura dinámica, abundantes cambios</i>

Por tanto la solución pasa por emplear un enfoque totalmente diferente, el **Data Warehousing** y el **Modelado Dimensional**. A continuación iremos estudiando estos conceptos. Con ello conseguimos reducir la carga transacciones (escritura), y utilizar esquemas diseñados especialmente para la consulta y análisis. Pasamos también a un sistema en el que prácticamente todos los accesos son de lectura, realizándose sólo escrituras masivas periódicas (por ejemplo, una vez al día y en horarios nocturnos).

Data Warehouse y Data Marts

Un **Data Warehouse** es una base de datos corporativa en la que se integra información depurada de las diversas fuentes que hay en la organización. Dicha información debe ser homogénea y fiable, se almacena de forma que permita su análisis desde muy diversas perspectivas, y que a su vez dé unos tiempos de respuesta óptimos. Para ello la información se encuentra altamente desnormalizada y modelada de una forma bastante diferente a los sistemas transaccionales, principalmente se utilizan los modelos en estrella (*star schema*) y en copo de nieve (*snowflake schema*) que estudiaremos más adelante.

Un Data Warehouse es mucho más que lo que hemos comentado hasta el momento. Según **Bill Inmon** se caracteriza por ser:

- **Orientado a temas:** los datos están organizados por temas para facilitar el entendimiento por parte de los usuarios, de forma que todos los datos relativos a un mismo elemento de la vida real queden unidos entre sí. Por ejemplo, todos los datos de un cliente pueden estar consolidados en una misma tabla, todos los datos de los productos en otra, y así sucesivamente.
- **Integrado:** los datos se deben integrar en una estructura consistente, debiendo eliminarse las inconsistencias existentes entre los diversos sistemas operacionales. La información se estructura en diversos niveles de detalle para adecuarse a las necesidades de consulta de los usuarios. Algunas de las inconsistencias más comunes que nos solemos encontrar son: en nomenclatura, en unidades de medida, en formatos de fechas, múltiples tablas con información similar (por ejemplo, tener varias aplicaciones en la empresa, cada una de ellas con tablas de clientes).
- **Histórico (variante en el tiempo):** los datos, que pueden ir variando a lo largo del tiempo, deben quedar reflejados de forma que al ser consultados reflejen estos cambios y no se altere la realidad que había en el momento en que se almacenaron, evitando así la problemática que ocurre en los sistemas operacionales, que reflejan solamente el estado de la actividad de negocio presente. Un Data Warehouse debe almacenar los diferentes valores que toma una variable a lo largo del tiempo. Por ejemplo, si un cliente ha vivido en tres ciudades diferentes, debe almacenar el periodo que vivió en cada una de ellas y asociar los hechos (ventas, devoluciones, incidencias, etc.) que se produjeron en cada momento a la ciudad en la que vivía cuando se produjeron, y no asociar todos los hechos históricos a la ciudad en la que vive actualmente. Si un cliente, durante todo el tiempo en que le hemos estado vendiendo ha pasado por tres estados civiles (soltero, casado y divorciado) debemos saber qué estado civil tenía en el momento en que le realizamos cada una de las ventas.
- **No volátil:** la información de un Data Warehouse, una vez introducida, debe ser de sólo lectura, nunca se modifica ni se elimina, y ha de ser permanente y

mantenerse para futuras consultas. Por ejemplo, si en el origen se modifica la cantidad de un producto que entra en el almacén, en el Data Warehouse no podemos hacer directamente una actualización sobre ese registro sin dejar ni el más mínimo rastro de que hubo antes otro valor.

Adicionalmente estos almacenes contienen metadatos (datos sobre los datos), que aportan un valor adicional, permitiendo tener información sobre su procedencia (sobre todo cuando tenemos múltiples fuentes), la periodicidad con la que han sido introducidos, la fiabilidad que nos ofrecen, etc. Todo ello nos aporta una ayuda adicional, tanto al usuario final como a los técnicos responsables de su mantenimiento, ayudando a estos últimos en aspectos como su auditoría y administración.

Kimball determinó que para él un Data Warehouse no era más que un conjunto de los Data Marts de una organización. Un **Data Mart** es una copia de las transacciones específicamente estructurada para la consulta y el análisis. Defiende por tanto una metodología Bottom-up a la hora de diseñar un almacén de datos.

Según Ralph Kimball, “Un **Data Mart** es un conjunto de datos flexible, idealmente basado en el nivel de granularidad mayor que sea posible, presentado en un modelo dimensional que es capaz de comportarse bien ante cualquier consulta del usuario. En su definición más sencilla, un data mart representa un único proceso de negocio”.

Un **proceso de negocio** es un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. La norma internacional ISO-9001 define un proceso como “una actividad que utiliza recursos, y que se gestiona con el fin de permitir que los elementos de entrada se transformen en resultados”.

La diferencia de un **Data Mart** con respecto a un **Data Warehouse** es solamente en cuanto al alcance. Mientras que un Data Warehouse es un sistema centralizado con datos globales de la empresa y de todos sus procesos operacionales, un Data Mart es un subconjunto temático de datos, orientado a un proceso o un área de negocio específica. Debe tener una estructura óptima desde todas las perspectivas que afecten a los procesos de dicha área. Es más, según *Ralph Kimball*, cada Data Mart debe estar orientado a un proceso determinado dentro de la organización, por ejemplo, a pedidos de clientes, a compras, a inventario de almacén, a envío de materiales, etc.

Si optamos por una solución basada en Data Marts, hay algo muy importante a tener en cuenta, no podemos volver a generar islas de información de las diferentes áreas o procesos de negocio, sino que han de quedar totalmente integradas para poder obtener siempre información coherente de toda organización. Para ello nos apoyamos en el uso de un Bus Dimensional.

Un **Bus Dimensional** es un esquema, habitualmente en forma de tabla, que representa los diversos Data Marts y las diferentes dimensiones definidas en nuestra organización. Como lo habitual es que se vayan creando en diversas fases, también se puede añadir una columna para representar en qué fase se abordará cada Data Mart. A continuación mostramos un ejemplo:

Data Marts	Fecha	Proveedor	Cliente	Producto	Contable	Fase
Ventas	X		X	X		Fase 1
Compras	X	X		X		Fase 1
Distribución	X	X	X	X		Fase 2
Finanzas	X				X	Fase 3
RRHH	X				X	Fase 3

Figura 0-2 Bus Dimensional.

Nota: La existencia de un Data Warehouse no descarta la existencia de Data Marts, ni viceversa. Es decir, puede haber organizaciones que tengan sólo un Data Warehouse, que sólo tengan Data Marts, o que tengan un Data Warehouse y Data Marts.

Esquema en estrella (*star schema*) y en copo de nieve (*snowflake schema*)

A la hora de modelar el Data Mart o Data Warehouse, hay que decidir cuál es el esquema más apropiado para obtener los resultados que queremos conseguir. Habitualmente, y salvo excepciones, se suele modelar la base de datos utilizando el **esquema en estrella (*star schema*)**, en el que hay una única tabla central, la tabla de hechos, que contiene todas las medidas y una tabla adicional por cada una de las perspectivas desde las que queremos analizar dicha información, es decir por cada una de las dimensiones (ver imagen):



Figura 0-3 Esquema en estrella (star schema)

La otra alternativa de modelado es la utilización del **esquema en copo de nieve (snowflake schema)**. Esta es una estructura más compleja que el esquema en estrella. La diferencia es que algunas de las dimensiones no están relacionadas directamente con la tabla de hechos, sino que se relacionan con ella a través de otras dimensiones. En este caso también tenemos una tabla de hechos, situada en el centro, que contiene todas las medidas y una o varias tablas adicionales, con un mayor nivel de normalización (ver imagen):

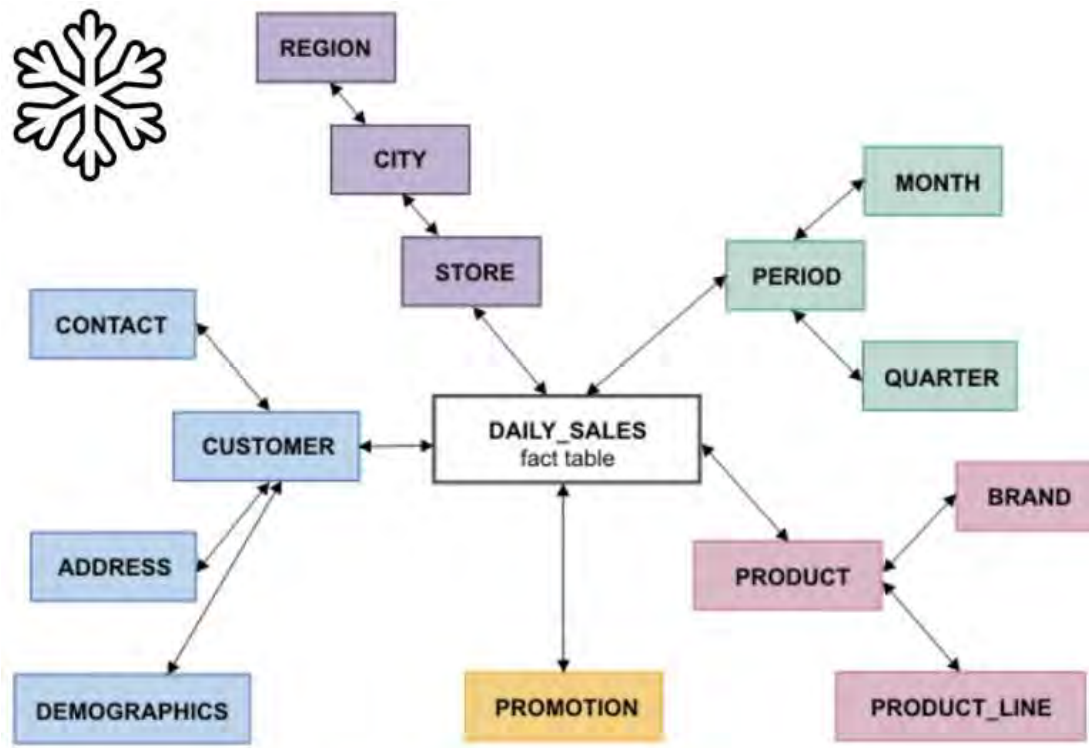


Figura 0-4 Esquema de copo de nieve (snowflake schema)

El modelo en estrella, aunque ocupa más espacio en disco (dato cada vez menos significativo), es más simple de entender por el usuario y ofrece un mejor rendimiento a la hora de ser consultado.

En ocasiones se opta por un modelo híbrido, que tiene parte en estrella y parte en copo de nieve.

Veamos un ejemplo de un esquema en el que hemos utilizado un modelo híbrido:

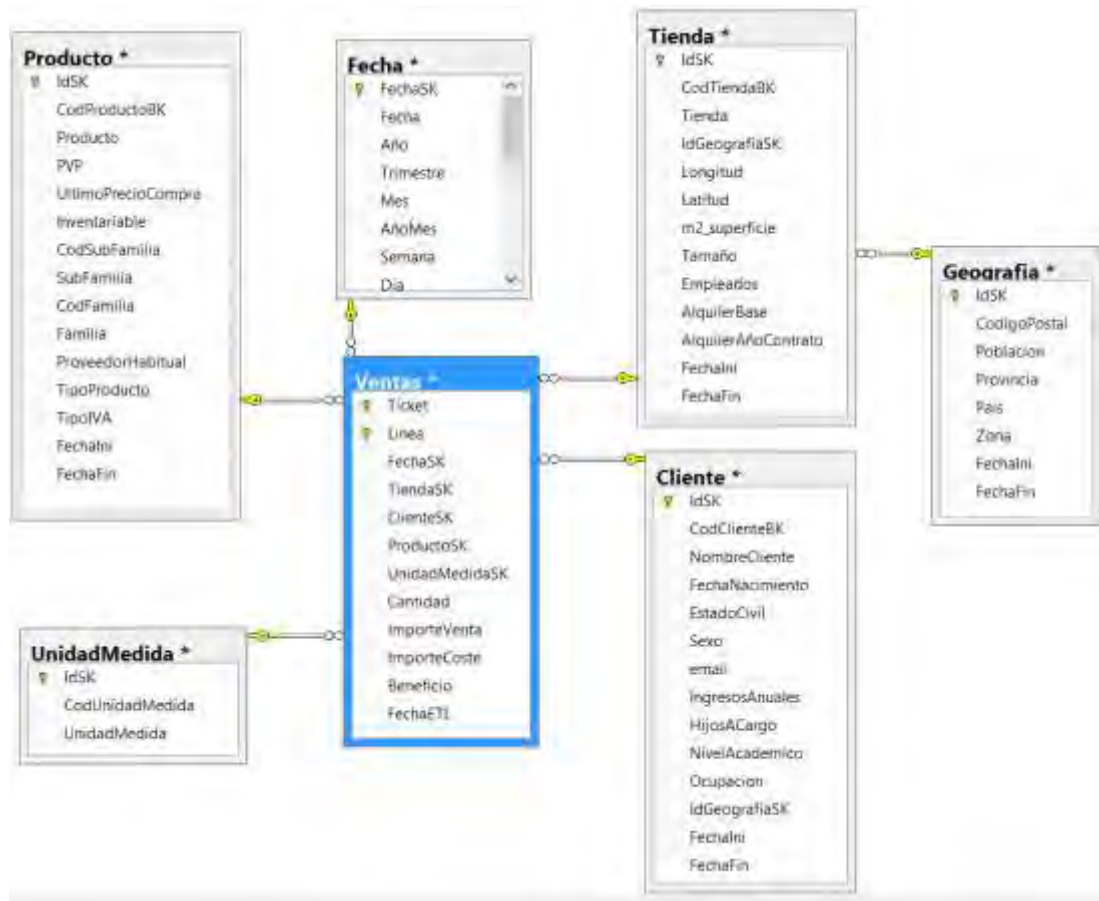


Figura 0-5 Esquema híbrido

Este modelo que aparece en la imagen anterior, es un modelo que utilizamos frecuentemente en [SolidQ, en nuestros cursos y Masters](#), para mostrar diversos análisis Comerciales y de Ventas, concretamente pertenece al caso práctico “Ventas en cadena Tiendas 24H”. En él hemos modelado la dimensión Geografía en copo de nieve (*snowflake*) con el fin de poder reutilizarla desde diversas dimensiones (tiendas, proveedores, clientes, empleados, etc.) que habitualmente tienen datos geográficos.

Modelado Dimensional

El **Modelado Dimensional** es utilizado hoy en día en la mayoría de las soluciones de BI. Es una mezcla correcta de normalización y desnormalización, comúnmente llamada *Normalización Dimensional*. Se utiliza tanto para el diseño de Data Marts como de Data Warehouses.

Básicamente hay dos tipos de tablas:

- Tablas de Dimensión (*Dimension Tables*)
- Tablas de Hechos (*Fact Tables*)

En los siguientes apartados nos centraremos en estudiarlas a fondo.

Trabajando con Hechos

Los **Hechos** están compuestos por los detalles del proceso de negocio a analizar, contienen datos numéricos y medidas (métricas) de Negocio a analizar. Contienen también elementos (claves externas) para contextualizar dichas medidas, como por ejemplo el producto, la fecha, el cliente, la cuenta contable, etc.

Las **Tablas de Hechos (Fact Tables)**, son tablas que representan dicho proceso de negocio, por ejemplo, las ventas, las compras, las incidencias recibidas, los pagos, los apuntes contables, los clics sobre nuestro sitio web, etc. Veamos con más detalle los elementos que las componen:

- **Clave principal:** identifica de forma única cada fila. Al igual que en los sistemas transaccionales toda tabla debe tener una clave principal, en una tabla de hechos puede tenerla o no, y esto tiene sus pros y sus contras, pero ambas posturas son defendibles.
- **Claves externas (Foreign Keys):** apuntan hacia las claves principales (claves subrogadas) de cada una de las dimensiones que tienen relación con dicha tabla de hechos.
- **Medidas (Measures):** representan columnas que contienen datos cuantificables, numéricos, que se pueden agregar. Por ejemplo, cantidad, importe, precio, margen, número de operaciones, etc.
- **Metadatos y linaje:** nos permite obtener información adicional sobre la fila, como por ejemplo, que día se incorporó al Data Warehouse, de qué origen proviene (si tenemos varias fuentes), etc. No es necesario para el usuario de negocio, pero es interesante analizar en cada tabla de hechos qué nos aporta y si merece pena introducir algunas columnas de este tipo.

PK¿?Si	FK1	...	FKn	M1	...	Mn
1	1200		30	10		100257,3
2	1001		30	2		2105,05
3	12		28	5		
4	17		12	14		
5	7800		5	25		
6	14103		51	78		
...
100.000.000	254		28	65		

Figura 0-1 Estructura de una Tabla de Hechos.

Las tablas de hechos, habitualmente son muy estrechas, tienen pocas columnas, además éstas son en su mayoría numéricas y de una longitud corta, de muy pocos bytes. Aunque sí que suelen ser muy largas, tienen un gran número de filas.

ProductoFK	FechaVentaFK	FechaEnvioFK	FechaCambioFK	TiendaSK	ComercioSK	Almacen	Cantidad	PrecioUnitario	Descuento	PrecioCosto	Impuestos	OrigenFila	FechaOper
219	20050701	20050713	20050708	678	265	SO43859	8	5.70	0	0.0000	2.736	0	20050702
223	20050701	20050713	20050708	678	265	SO43859	2	0.1865	0	0.0000	0.0296	0	20050702
220	20050701	20050713	20050708	678	265	SO43859	4	20.1885	0	0.0000	0.0000	0	20050702
326	20050701	20050713	20050708	132	265	SO43860	1	419.4588	0	413.1483	33.5957	0	20050702
319	20050701	20050713	20050708	117	265	SO43860	1	874.794	0	864.7083	69.3855	0	20050702
300	20050701	20050713	20050708	442	268	SO43861	1	809.76	0	809.0928	64.7808	0	20050702
296	20050701	20050713	20050708	442	268	SO43861	1	714.7043	0	617.0281	57.1763	0	20050702
394	20050701	20050713	20050708	442	268	SO43861	2	714.7043	0	617.0281	114.3527	0	20050702
223	20050701	20050713	20050708	442	268	SO43861	4	0.1865	0	0.0000	0.0000	0	20050702

Figura 0-2 Tabla de Hechos de Ventas

En la figura anterior se muestra la tabla de hechos de ventas, en la que podemos apreciar que tiene una serie de claves externas a las dimensiones: Producto, Fecha (hay varias fechas), Tienda y Comercial. Adicionalmente tiene el número de venta; luego una serie de medidas: Cantidad, Precio Unitario, Descuento, PrecioCosto e Impuestos. Y por último un par de columnas de metadatos: OrigenFila (identifica el sistema de origen desde el que se obtuvo dicha fila) y FechaOper (fecha en la que la fila entró en la tabla de hechos).

Es importante a la hora de diseñar una tabla de hechos, tener en cuenta el nivel de **granularidad** que va a tener, es decir, el nivel de detalle más atómico que vamos a encontrar de los datos. No es lo mismo tener una fila por cada venta, que una fila donde se indiquen las ventas del día para cada artículo y tienda. A mayor granularidad, mayor será el número de filas de nuestra tabla de hechos, y dado que el espacio en disco y rendimiento no se ven notablemente afectados en los sistemas actuales, debemos llegar siempre al máximo nivel de granularidad que resulte útil a los usuarios.

Uno de los errores más frecuentes y que debemos evitar en nuestros diseños es añadir dimensiones en una tabla de hechos antes de definir su granularidad.

La **agregación** es el proceso por el cual se resumen los datos a partir de las filas de detalle de máxima granularidad. Hoy en día disponemos de sistemas OLAP y sistemas In-Memory que se encargan de agregar dichos datos y ofrecerlos al usuario con una gran rapidez y eficacia.

Trabajando con Dimensiones

Las **Dimensiones** nos permiten contextualizar los hechos, agregando diferentes perspectivas de análisis a ellos. Si una agregación de una medida nos devuelve el valor 17.538 unidades, por sí sólo no nos dice nada, en cambio, si le agregamos las perspectivas tiempo, tienda y cliente, podríamos decir que “hemos vendido 17.538 unidades en el mes de marzo de 2012, en la tienda de Murcia al cliente Juan López García”.

Las **Tablas de Dimensiones** son las que almacenan la información de las dimensiones. Una dimensión contiene una serie de atributos o características, por las cuales podemos agrupar, rebanar o filtrar la información. A veces estos atributos están organizados en **jerarquías** que permiten analizar los datos de forma agrupada, dicha agrupación se realiza mediante relaciones uno a muchos (1:N). Por ejemplo, en una dimensión Fecha es fácil que encontremos una jerarquía formada por los atributos Año, Mes y Día, otra por Año, Semana y Día; en una dimensión Producto podemos encontrarnos una jerarquía formada por los atributos Categoría, Subcategoría y Producto. Como se ha podido comprobar en los ejemplos, se puede dar el caso de que exista más de una jerarquía para una misma dimensión.

Las tablas de dimensiones, por lo general son muy anchas (contienen muchos atributos, y éstos pueden tener bastantes caracteres cada uno) y cortas (suelen tener pocas filas). Por ejemplo en una dimensión Producto, podemos encontrarnos con que tiene varias decenas de atributos, y que éstos están desnormalizados. No es extraño encontrarnos aquí valores ya en texto, como claves a otras tablas, por ejemplo categoría ('alimentación', 'textil', etc.), subcategoría ('congelados', 'frescos', 'bebidas', etc.), colores ('rojo', 'verde', 'amarillo', 'azul', etc.), tamaño ('pequeño', 'mediano', 'grande'), etc.

ProductoSK	ProductoBK	Categoría	Subcategoría	Producto	Peso	Tamaño	Color	Coste	FechaInicio	FechaFin
350	BK-M82B-44	Bicicleta	Bicicleta de montaña	Montaña: 100, negra, 44	21.13	44	Black	1898.0944	2005-07-01	2006-06-30
606	BK-M82B-44	Bicicleta	Bicicleta de paseo	Montaña: 100, negra, 44	21.13	44	Black	1898.0944	2007-06-30	NULL
352	BK-M63S-38	Bicicleta	Bicicleta de montaña	Montaña: 200, plateada, 38	23.35	38	Silver	1117.8559	2006-07-01	2007-06-30
561	BK-M63S-38	Bicicleta	Bicicleta de paseo	Montaña: 200, plateada, 38	23.35	38	Silver	1117.8559	2007-06-30	NULL
353	BK-R79Y-42	Bicicleta	Bicicleta de montaña	Montaña: 200, plateada, 38	23.35	38	Silver	1265.8195	2007-06-30	NULL

Figura 0-1 Tabla Dimensión Producto (star schema)

En una tabla de dimensiones, habitualmente no es posible utilizar la clave de negocio (*business key*) como clave principal (*primary key*), e incluso en el caso de que sea posible no es aconsejable. Una clave de negocio como clave principal no es aconsejable en muchos casos por temas de rendimiento, ya que desde este punto de vista es recomendable utilizar números enteros de pocos bytes (en el caso concreto de SQL Server es muy recomendable usar el tipo de datos INT que ocupa 4 bytes). Si en el sistema transaccional tenemos, por ejemplo, una clave principal que es un char(10) siempre será más óptimo utilizar un tipo de datos numérico de menos bytes como clave principal en nuestra tabla de dimensiones. Adicionalmente hay

casos en los que ya no es tan sólo un tema de rendimiento, sino que simplemente no es viable utilizar como clave principal de la tabla de dimensiones la clave principal de la tabla del transaccional. Por ejemplo, si tenemos varios orígenes de datos que queremos consolidar en una misma tabla de dimensiones, y cada uno utiliza un tipo de datos o longitud diferente, o simplemente para un mismo elemento en cada tabla de origen tiene un valor diferente. Por otro lado cuando queremos almacenar el historial de cambios, cumpliendo las características básicas que debe cumplir un Data Warehouse según *Inmon*, necesitamos tener, en muchos casos, varias filas con las diferentes versiones de los atributos y el periodo durante el que han estado vigentes, lo que implica que en la tabla de dimensiones habrá duplicidades en la clave de negocio, lo que impide que ésta pueda ser la clave principal.

Por tanto, lo habitual es que optemos por tener una clave principal diferente, esta clave es lo que se conoce con el nombre de clave subrogada.

Una **Clave subrogada** (*subrogate key*) es un identificador único que es asignado a cada fila de la tabla de dimensiones, en definitiva, será su clave principal. Esta clave no tiene ningún sentido a nivel de negocio, pero la necesitamos para identificar de forma única cada una de las filas. Son siempre de tipo numérico, y habitualmente también son auto-incrementales. En el caso de SQL Server recomendamos que sean de tipo INT con la propiedad *identity* activada (es una recomendación genérica, a la que siempre habrá excepciones).

Una **Clave de negocio** (*business key*) es una clave que actúa como *primary key* en nuestro origen de datos, y es con la que el usuario está familiarizado, pero no puede ser clave principal en nuestra tabla de dimensiones porque se podrían producir duplicidades, como veremos más adelante al explicar el concepto de *Slowly Changing Dimensions*.

SCD (Slowly Changing Dimensions)

Como hemos comentado anteriormente, la información de los sistemas transaccionales puede ser modificada, aunque éstos sólo guardan la última versión. Por el contrario en un Data Warehouse, debemos reflejar ese historial de cambios para mostrar la verdad que había en el momento en que se produjeron los hechos.

Veamos un ejemplo. Si en nuestro sistema transaccional asociamos cada venta al comercial que la realiza, y éste a su vez depende de un director de zona. En la tabla de ventas queda reflejado el comercial que realiza la venta, y en la tabla del empleado se almacena el director de zona del que depende, ya que tenemos los datos normalizados. ¿Qué ocurre si un comercial, por cualquier motivo, bien personal o bien laboral, le cambian la zona asignada?, ¿Y si además la nueva zona depende de otro director de zona?, ¿Y qué ocurre si sacamos un informe de ventas

de ese nuevo director de zona? Pues que se le han trasladado a él todas las ventas que ha hecho este comercial durante toda su vida laboral en la empresa. Esto no es real, e imaginamos que su antiguo jefe de zona no estará en absoluto de acuerdo con estos informes de ventas, además de que no son ciertos. Cuando diseñamos un Data Warehouse debemos evitar esta problemática que tenemos en muchos sistemas transaccionales, donde sólo tenemos la versión actual de los datos. Para ello hay una serie de técnicas que nos permiten ir detectando los cambios que ocurren en el transaccional y dejándolos reflejados. Volviendo con el ejemplo anterior, en la tabla de dimensiones se deberían tener dos filas (o versiones) del empleado, una en la que se indica cuál es su jefe de zona antiguo, y durante qué periodo ha sido su jefe de zona, y otra que indica cuál es su jefe actual y desde cuándo. Adicionalmente, cada una de las ventas debe estar apuntando a la versión correcta del comercial, es decir, las ventas deben apuntar a la versión del comercial correspondiente al momento en que se produjeron, quedando así reflejado el jefe de zona y la zona que realmente tenía asignados en el momento de cada venta.

Por el contrario, hay otros casos en los que no necesito reflejar el historial, por ejemplo, si corrijo el nombre de dicho comercial porque lo tenía mal escrito, no quiero tener dos versiones de él, una con el nombre mal escrito y otra con él bien escrito, sino que quiero que se sobrescriba y siempre aparezca la versión actual que es donde está escrito correctamente.

Vistos estos ejemplos, pasemos a describir los diferentes tipos de SCD más habituales:

- **SCD tipo 1, Sobrescritura:** la nueva información sobrescribe a la antigua, no se guardan históricos y sólo se tiene la versión actual. Dicha sobre escritura se produce cuando se detecta algún error en los valores para corregirlo y mejorar la calidad del dato. Desde el punto de vista analítico sólo interesa la versión actual.
- **SCD tipo 2, historial de cambios:** refleja toda la información histórica. Por cada cambio que se produzca, se crea una nueva fila en la tabla de dimensiones con la fecha de inicio y una nueva clave subrogada, y se marca la fecha de fin de la versión anterior. Cada hecho que entra, debe comprobar a qué versión de la fila en la tabla de dimensiones se debe asociar (qué clave subrogada debe almacenar) en función de la fecha en la que se produzca.

Como hemos visto en los ejemplos, es habitual, que incluso en una misma tabla haya atributos de *tipo 1* y de *tipo 2*, y deberemos dar el tratamiento adecuando a cada caso en nuestros procesos ETL.

El término *Slowly Changing Dimension*, SCD por sus siglas en inglés, suele aparecer traducido como “dimensiones lentamente cambiantes”, aunque en la ayuda de SQL

Server (*Books Online*) aparece como “dimensiones de variación lenta”, téngalo en cuenta el lector si busca información en español sobre este término.

Recomiendo al lector que consulte la bibliografía de *Ralph Kimball*, especialmente el libro *“The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (second edition)”* donde se exponen ampliamente estos conceptos y otros relacionados con el modelado dimensional, así como una amplia variedad de casos prácticos

Por último quiero concluir indicando los elementos que van a formar una tabla de dimensiones:

- **Clave subrogada:** Es la clave principal de la tabla de dimensiones. Nos permite identificar de forma única cada fila, suele ser un entero auto-incremental. Es totalmente transparente al usuario de negocio, no la usará en ningún momento, ni tan siquiera tendrá conocimiento de su existencia.
- **Clave de negocio:** Es la clave con la que trabaja habitualmente el usuario, pero no puede ser la clave principal porque se pueden producir duplicidades.
- **Atributos de la dimensión:** serán cada una de las características que necesitemos almacenar. Lo habitual es que haya varias decenas de ellos, incluso que en algunos casos superen el centenar.
- **Fecha de Inicio y Fecha de Fin:** Servirán para conocer el periodo de vigencia de cada una de las versiones de los atributos.

SubrogateKey	BusinessKey	Atrib1	Atrib2	...	AtribN	Fecha Ini	Fecha Fin
1	C1838					01/01/2010	05/04/2011
2	B23949					01/01/2010	Null
3	A989889					02/01/2010	Null
4	A3945					20/01/2010	Null
5	C1838					05/04/2011	Null

Figura 0-2 Estructura de una Tabla de Dimensión

Básicamente, una tabla de un sistema OLTP tiene una clave de negocio que suele ser además clave principal, y una serie de atributos. Cuando tenemos una tabla de dimensiones, la clave de negocio deja de ser la clave principal, aparece una nueva clave principal, que es la clave subrogada, y se agregan dos columnas (FechaInicio y FechaFin) para gestionar los periodos de vigencia de cada versión. Además es habitual que tenga un mayor número de atributos, que son la recopilación de los existentes en las diferentes fuentes de datos, y algunos adicionales que se calculan en los procesos ETL.

Volvamos a mostrar la figura de la tabla de la dimensión Producto donde se aprecia todo lo visto para tres productos. Vemos que los productos BK-M83B-44 y BK-M68S-38, han tenido dos versiones, dado que se cambiaron de categoría, así como las fechas en las que ha estado vigente cada una de ellas, mientras que el producto BK-R79Y-42 sólo tiene una versión. También podemos saber cuál es la versión actual de cada uno de ellos, ésta es la que la columna FechaFin vale NULL.

ProductoSK	ProductoBK	Categoria	Subcategoria	Producto	Peso	Tamaño	Color	Coste	FechaInicio	FechaFin
350	BK-M82B-44	Bicicleta	Bicicleta de montaña	Montaña: 100, negra, 44	21.13	44	Black	1898.0944	2005-07-01	2006-06-30
606	BK-M82B-44	Bicicleta	Bicicleta de paseo	Montaña: 100, negra, 44	21.13	44	Black	1898.0944	2007-06-30	NULL
352	BK-M68S-38	Bicicleta	Bicicleta de montaña	Montaña: 200, plateada, 38	23.35	38	Silver	1117.8559	2006-07-01	2007-06-30
561	BK-M68S-38	Bicicleta	Bicicleta de paseo	Montaña: 200, plateada, 38	23.35	38	Silver	1117.8559	2007-06-30	NULL
353	BK-R79Y-42	Bicicleta	Bicicleta de montaña	Montaña: 200, plateada, 38	23.35	38	Silver	1265.6195	2007-06-30	NULL

Figura 0-3 Tabla Dimensión Producto

A continuación abordaremos diversas casuísticas de “versionado de datos” y volveremos sobre las estructuras de las tablas de hechos y de dimensiones, así como sobre el concepto de *SCD (Slowly Changing Dimensions)*.

Versiones de datos

Los orígenes de datos que solemos utilizar tienen una problemática que en la mayoría de los casos pasa desapercibida al usuario y al analista de negocio, pero que tiene una vital importancia a la hora de conocer la verdad de lo ocurrido históricamente en nuestro negocio:

“La mayoría de los sistemas tratan datos históricos con respecto a la realidad actual, no con respecto a la realidad que había en el momento en que se produjeron”

En este tema explicaremos con detalle este importante aspecto y ofreceremos diversas soluciones de diseño del modelo y de los procesos ETL (procesos de extracción, transformación y carga de datos) que darán solución a este problema, que consideramos de gran importancia y de necesaria solución para tener una información de calidad.

Veamos un ejemplo para entender lo anterior:

Tenemos datos de ventas almacenados en nuestro sistema de origen, entre otros y por simplificar (esto es más complejo en la realidad, aquí sólo contamos lo mínimo para entender la problemática) tenemos los datos de clientes almacenados en una tabla y los datos de ventas en otra, tal y como se muestra en la siguiente imagen, que representa la información tal y como estaba el 30/3/2008:

Ventas			Cliente		
Fecha	Importe	Cliente	Codigo	Nombre	Estado Civil
01/01/2003	750	2	1	Pepe	Casado
15/01/2004	150	1	2	Juan	Soltero
03/07/2006	200	1			
15/12/2007	75	2			

Si con esta información sacamos nuestro informe de ventas por estado civil, nos da el siguiente resultado:

<i>Informe ventas por Estado Civil 30/3/2008</i>			
Estado Civil	Importe		
Casado	350		
Soltero	825		

Va transcurriendo el tiempo, seguimos haciendo ventas a estos clientes. El 2/4/2008 viene Juan a nuestra tienda y, además de venderle productos nuevamente, nos comunica que se ha casado, y nosotros actualizamos su ficha de cliente:

Ventas			Cliente		
Fecha	Importe	Cliente	Codigo	Nombre	Estado Civil
01/01/2003	750	2	1	Pepe	Casado
15/01/2004	150	1	2	Juan	Casado
03/07/2006	200	1			
15/12/2007	75	2			
02/04/2008	56	2			

Sigue transcurriendo el tiempo, seguimos vendiendo a Juan y Pepe, y a 5/5/2014, así quedan nuestras tablas de Ventas y Clientes:

Ventas			Cliente		
Fecha	Importe	Cliente	Codigo	Nombre	Estado Civil
01/01/2003	750	2	1	Pepe	Casado
15/01/2004	150	1	2	Juan	Casado
03/07/2006	200	1			
15/12/2007	75	2			
02/04/2008	56	2			
07/05/2009	114	2			
01/05/2010	350	1			
04/06/2011	195	2			
01/09/2012	375	1			
05/09/2013	415	2			
04/03/2014	290	2			

Volvemos sacar un informe de ventas por estado civil, cuyo resultado es el siguiente:

<i>Informe ventas por Estado Civil 5/5/2014</i>			
Estado Civil	Importe		
Casado	2970		
Soltero	0		

¿Es correcto este informe?, ¿Qué ha pasado con las ventas que hicimos a Juan mientras estaba soltero?

Que desde el 2/4/2008 en que actualizamos los datos de Juan cambiando su estado civil de 'Soltero' a 'Casado' se han falseado todos los datos históricos de las ventas de Juan, apareciendo en todas ellas como 'Casado'. Es decir, estamos tratando datos históricos teniendo en cuenta la realidad actual, no la que había cuando se produjeron los hechos.

A continuación vamos a aplicar a este mismo caso las técnicas estudiadas referentes a SCD (*Slowly Changing Dimensions*) para entender qué estructura de almacenamiento de datos deberíamos tener y cómo sería el proceso ETL que nos permitiese mantener los datos históricos sin que se vean afectados por cambios que se producen posteriormente y que no les deberían afectar.

Ahora veremos la información como estaba, teniendo en cuenta una nueva estructura (con más columnas en cada tabla de dimensiones) y procesos ETL que permitan mantener el historial tal y como estaba cuando se produjeron los hechos.

Información a 30/3/2008:

Ventas			Cliente					
Fecha	Importe	Cliente	SK	CodigoBK	Nombre	Estado Civil	Fecha Ini	Fecha Fin
01/01/2003	750	1129	1129	1	Pepe	Casado	15/01/2004	Null
15/01/2004	150	1130	1130	2	Juan	Soltero	01/01/2003	Null
03/07/2006	200	1130						
15/12/2007	75	1129						

Si con esta información sacamos nuestro informe de ventas por estado civil, nos da el siguiente resultado:

<i>Informe ventas por Estado Civil 30/3/2008</i>			
Estado Civil	Importe		
Casado	350		
Soltero	825		

Va transcurriendo el tiempo, seguimos haciendo ventas a estos clientes. El 2/4/2008 viene 'Juan' a nuestra tienda y, además de venderle productos nuevamente, nos comunica que se ha casado y nosotros actualizamos su ficha de cliente. Pero en esta ocasión cuando por la noche se ejecuta el proceso ETL que actualiza los datos del cliente en el Data Mart, se detecta el cambio y se genera una nueva versión de la ficha de Juan, y además, las ventas de ese día ya se asignan a la nueva versión:

Ventas			Cliente						
Fecha	Importe	Cliente	SK	CodigoBK	Nombre	Estado Civil	Fecha Ini	Fecha Fin	
01/01/2003	750	1129	1129	1	Pepe	Casado	15/01/2004	Null	
15/01/2004	150	1130	1130	2	Juan	Soltero	01/01/2003	02/04/2008	
03/07/2006	200	1130	
15/12/2007	75	1129	1254	2	Juan	Casado	02/04/2008	Null	
02/04/2008	56	1254							

Sigue transcurriendo el tiempo, seguimos vendiendo a Juan y Pepe, y a 5/5/2014, así quedan nuestras tablas de Ventas y Clientes:

Ventas			Cliente						
Fecha	Importe	Cliente	SK	CodigoBK	Nombre	Estado Civil	Fecha Ini	Fecha Fin	
01/01/2003	750	1129	1129	1	Pepe	Casado	15/01/2004	Null	
15/01/2004	150	1130	1130	2	Juan	Soltero	01/01/2003	02/04/2008	
03/07/2006	200	1130	
15/12/2007	75	1129	1254	2	Juan	Casado	02/04/2008	Null	
02/04/2008	56	1254							
07/05/2009	114	1254							
01/05/2010	350	1129							
04/06/2011	195	1254							
01/09/2012	375	1129							
05/09/2013	415	1254							
04/03/2014	290	1254							

Para las ventas que se han producido entre el 01/01/2003 y el 01/04/2008 el proceso ETL les asignó el código 1130 (esto quedó grabado en su momento y no se modifica con el paso del tiempo), y para las ventas desde el 02/04/2008 en adelante les asigna el código 1254. Por tanto, si consultamos por el cliente 'Juan', se suman todas, pero si consultamos por el estado civil cada una va al grupo que le corresponde.

En resumen, en la tabla de dimensiones aplicamos las técnicas SCD estudiadas y por cada cambio en un atributo del que queramos mantener su historial de cambios, generamos en el proceso ETL una nueva versión del registro, asignando una nueva *clave subrogada*, y controlamos el rango de fechas de vigencia de cada versión. Adicionalmente en el proceso ETL que carga los hechos, asignamos, en función de la fecha del hecho, la *clave subrogada* de la fila de la dimensión con la versión que corresponde a ese rango.

Volvemos sacar un informe de ventas por estado civil, cuyo resultado es el siguiente:

<i>Informe ventas por Estado Civil 5/5/2014</i>			
Estado Civil	Importe		
Casado	1420		
Soltero	1550		

Ahora si han quedado las ventas de Juan asignadas correctamente al estado civil en el que se encontraba en el momento en que le realizamos cada una de las ventas.

Cerrando el ciclo de modelado

Llegados a este punto, es el momento de recopilar y revisar todo lo estudiado, y quedarnos con una imagen clara y concisa de los pasos que debemos seguir desde que nos planteamos la necesidad de solucionar el poder analizar un área de negocio, hasta que lo dejamos plasmado en un modelo dimensional para posteriormente llevar a cabo dicha solución con las herramientas de que dispongamos, en nuestro caso, las de la plataforma de Microsoft BI.

Estos son los pasos que debemos seguir:

1. **Seleccionar el proceso.** Para ello puede utilizar, tanto lo explicado en este tema, como en el tema anterior, en el apartado *Toma de requerimientos*.
2. **Definir la granularidad.** Como hemos visto anteriormente la granularidad es el máximo nivel de detalle que vamos a tener en cada tabla de hechos.
3. **Escoger las dimensiones.** Siempre lo haremos una vez definida la granularidad. Y en ellas definiremos las diversas jerarquías en base a los atributos disponibles.
4. **Identificar las medidas.** Por cada una de ellas especificaremos de qué columna del origen procede, y cuando no se obtenga directamente, el cálculo a realizar para su obtención.

Habrà que crear la documentación necesaria en base a las notas de las reuniones mantenidas y de nuestros conocimientos y experiencia en el diseño de este tipo de soluciones.

Estos pasos los debemos seguir por cada proceso o área de negocio. Si definimos más de uno, es conveniente que adicionalmente mantenemos de forma esquematizada las relaciones entre las tablas de hechos y las dimensiones mediante un **Bus Dimensional**, que incluiremos en la documentación tal y como ya estudiamos en su momento.

En toda esa documentación tendremos disponible lo que se denomina **modelo lógico**.

El **Modelo Lógico** es, en este tipo de proyectos, un documento que recopila el modelo dimensional que da solución a nuestros problemas de negocio mediante texto, tablas y gráficos. Básicamente recopila, descripciones detalladas, tablas, atributos de las tablas, relaciones entre dichas tablas, jerarquías de las que constará y ejemplos. Bien se puede hacer con cualquier procesador de textos, por ejemplo, con Microsoft Word, bien se pueden utilizar herramientas especializadas para ello. El

propósito principal del *modelo lógico* es ofrecer un mapa que describa de forma gráfica los elementos que utilizaremos para el almacenamiento de la información y cómo están estructurados.

En este caso estamos orientando toda la documentación de forma que pueda ser realizada por un usuario de negocio, sin necesidad de ciertos conocimientos tecnológicos propios del área de TI. Si usted se encuentra desarrollando una solución de BI con un equipo mixto, formado por personal del área de negocio y por técnicos del departamento de TI, déjese asesorar por ellos y que esta parte la lleven a cabo los técnicos con la herramienta que consideren más apropiada. Si, en cambio, se encuentra sólo desarrollando una solución de BI Personal, le recomendamos que opte por utilizar Microsoft Word y seguir todas las pautas estudiadas.

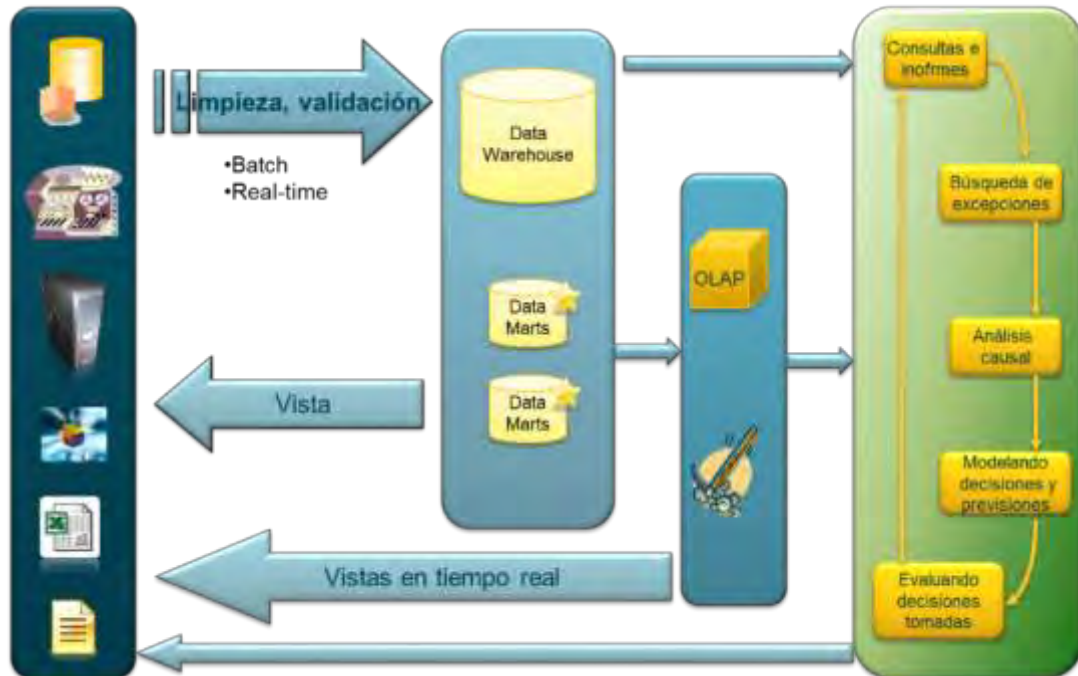
Si se encuentra realizando alguno de los [cursos o Masters ofertados por SolidQ](#), podrá utilizar los diferentes documentos y materiales que forman parte de ellos, donde puede encontrar ejemplos de documentos de Visión y Alcance, de mapeo de datos y demás, así como pedir consejo al profesor tutor asignado para que, en función de las necesidades de documentación que tenga, le aconseje las mejores alternativas.

Una vez que disponemos del modelo lógico terminado, el siguiente paso es crear el modelo físico.

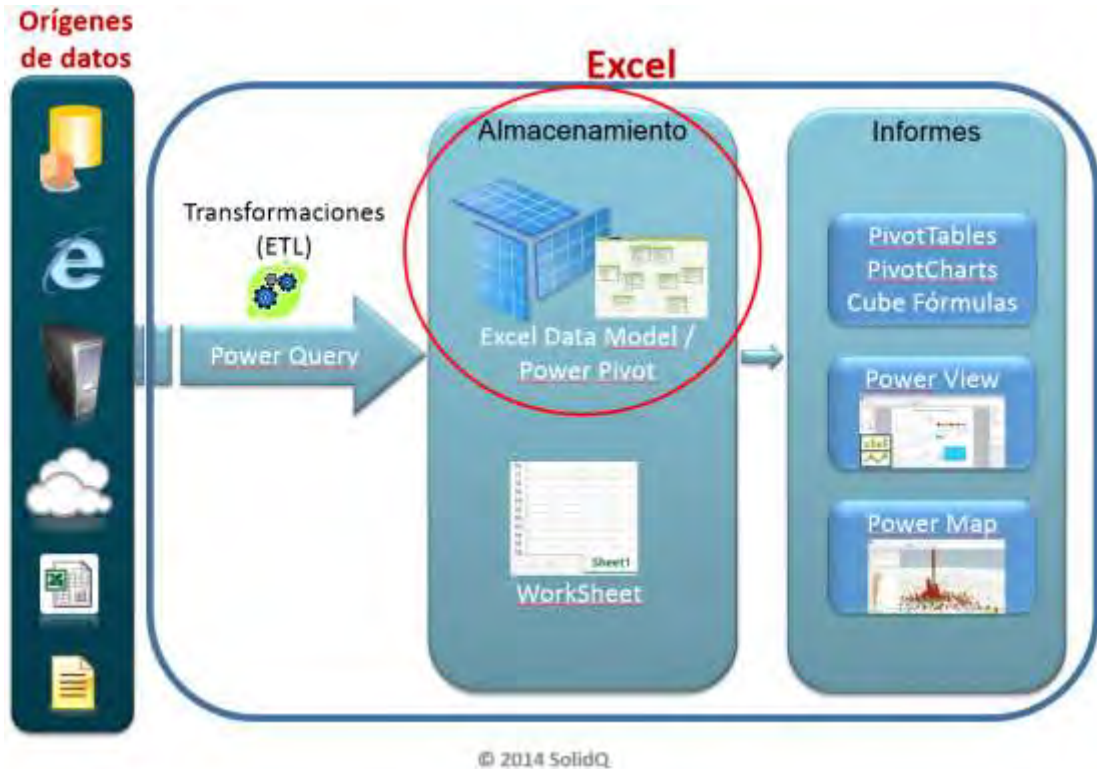
El **Modelo Físico** es la creación de las bases de datos, con sus correspondientes elementos utilizando la herramienta elegida para almacenar la información y hacerlo de la forma más apropiada.

Si vamos al caso de la tecnología utilizada en base a la plataforma Microsoft, que es la que estamos estudiando, tenemos dos alternativas para la creación de los modelos físicos:

- Si optamos por una solución de **BI corporativa**, basada en el almacenamiento en SQL Server, recomendamos crear, por un lado el almacenamiento en el motor relacional de *SQL Server* de nuestros Data Marts y/o Data Warehouse. Y a partir de ahí construir la capa analítica utilizando *SQL Server Analysis Services (SSAS)* y optando por una instancia *Multidimensional* (con cubos OLAP) o por una instancia *Tabular* (con modelos “In-Memory”). Recuerde la imagen de arquitectura de BI Corporativo estudiada en el tema anterior:



- Si optamos por una solución de **BI personal** basada en Excel 2013 y los complementos de *Power BI* (*Power Pivot* y *Power Query*), crearemos con ellos el modelo físico. Recuerde la imagen de arquitectura de BI Personal estudiada en el tema anterior:



En los próximos capítulos nos adentraremos en todos los detalles tecnológicos para la creación de modelos de BI Personal, los estudiaremos en detalle, utilizando y estudiando a fondo las herramientas que aparecen en la imagen anterior. Aunque no dejaremos de lado las soluciones de BI Corporativo, ni el BI Colaborativo, que retomaremos de nuevo como parte del último tema del libro.

Si está realizando alguno de los [cursos o Masters de SolidQ](#), esa sensación habrá desaparecido, o al menos disminuido enormemente 😊, ya que en ellos se realizan una serie de “laboratorios guiados paso a paso” para consolidar lo estudiado sobre diversos casos prácticos reales.

Si no es su caso, le recomiendo que se forme con nuestro curso:

Analiza tu Negocio con Excel y Power BI.

Aprende de tus datos

[Quiero obtener información detallada y registrarme](#)

ENLACES ESENCIALES

Visita el [blog de Salvador Ramos](#)

Mi libro “Microsoft BI: Vea el cubo medio lleno” ([descárgalo gratis](#))

Linkedin [Salvador Ramos](#)

Twitter [Salvador Ramos](#)

Visita [los blogs de SolidQ](#)

Libros y publicaciones de SolidQ, la mayoría gratuitos ([descárgalos aquí](#))

Accede a los [Cursos de SolidQ](#)

Accede al [Calendario de cursos de SolidQ](#) (continuamente actualizado)

Accede a los [Servicios de SolidQ](#)

Twitter de [SolidQ](#), y también nuestro [canal en español](#)

GRACIAS

Antes de que te vayas, quiero decirte “Gracias por leer mi libro”.

Sé que hay otros muchos libros en la materia, pero apostaste por este. Ahora sólo quiero pedirte un “pequeño” favor:

Dedica unos minutos y deja un testimonio en Amazon, lo necesito para seguir mejorando y aportando contenido que sea de tu interés.

Y, por último, si crees que merece la pena compartir este libro, ¿podrías tomarte unos segundos y mostrárselo a tus seguidores en las Redes Sociales? El boca a boca es crucial para hacer llegar estos conocimientos al mayor número posible de personas interesadas en la materia, les serás de gran ayuda. Si tienes un momento, te estaré muy agradecido.

¡Gracias!