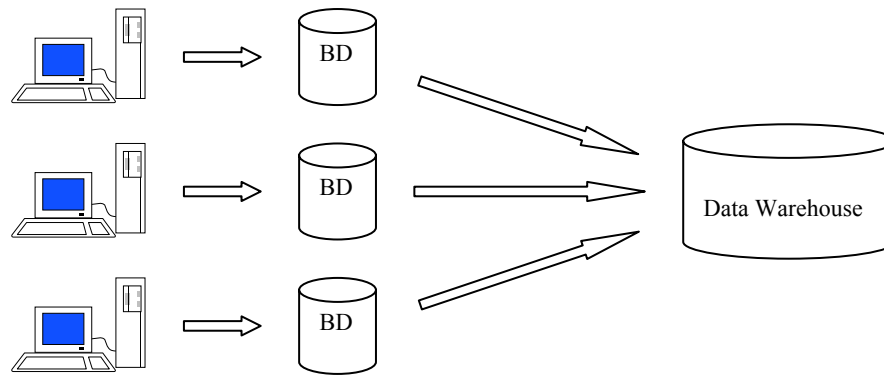


Data Warehouse

Hoy en día, las **bases de datos relacionales** son operativas en un entorno muy concreto que responde a las necesidades para las que se crearon. Estas necesidades suelen involucrar entornos de gestión puros en los que las características principales de las operaciones suelen ser las de la simplicidad en las estructuras y tipos de los datos, **utilización de transacciones cortas**, etc. Para ver un ejemplo supóngase la base de datos operativa que puede estar usando un cajero de una sucursal bancaria en ventanilla. Es cierto que el volumen de datos global de la base de datos puede ser muy alto, pero las operaciones que se manejan en cada una de las transacciones son muy simples (la inserción de un ingreso o de un reintegro en la base de datos probablemente involucre nada más que la inserción en una determinada tabla de una tupla que refleje este hecho). Por tanto, en cada una de las operaciones (de forma general) **se involucran muy pocos datos**; pero es cierto que **el volumen global es enorme** (y dado que se van acumulando diariamente tiende a crecer muy rápidamente). Además, la disponibilidad de la base de datos debe ser total: sería inaceptable que un cliente de dicha sucursal se viese obligado a esperar 15 minutos a que el sistema gestor realizara la transacción que reflejase un reintegro para poder hacerlo.

Por otra parte, las necesidades de información hoy en día han variado. La disponibilidad de gran cantidad de información es de vital importancia para los negocios ya que las decisiones de futuro se suelen tomar sobre la base de dicha información. Supongamos el ejemplo de la sucursal bancaria de antes. Es evidente que si el director de dicha sucursal quiere tomar una decisión sobre si potenciar o no determinado producto financiero y para ello necesita analizar la evolución del índice de morosidad del último año, no va a tener en cuenta para ello, si determinado cliente ha acudido o no esa mañana a reponer una cantidad en su cuenta. **Las necesidades del director son mucho más globales**: necesita conocer la evolución ascendente o descendente de dicho índice sin entrar en detalle. Está claro, por tanto, que los hechos que la base de datos operativa tiene no son lo que el director necesita. Sin embargo la "globalización" de los datos que busca el director se basa claramente en la información reflejada en dicha base de datos, pero organizada de otra manera (en este caso resumida). Este tipo de necesidades para reflejar tendencias, evoluciones, hechos históricos en el negocio y posibilidades futuras son algo que la alta directiva de las instituciones o empresas debe manejar y maneja de una forma habitual y es la causante de que hayan aparecido en el mercado **herramientas del tipo de "ayudas a la toma de decisiones"**.

Los procesos que obtienen la información con la estructura adecuada (resumidos, globalizados, etc.) son sistemas que, a priori, **involucran un alto coste en consumo de recursos** dado el gran volumen de datos sobre el que actúan y el tiempo de procesamiento que conlleva la obtención de las nuevas estructuras. Está claro, por lo dicho antes, que las denominadas "bases de datos operacionales" o "de producción" de la empresa no se pueden ver afectadas en sus tiempos de respuesta por dicho consumo en recursos. Esto, unido al hecho de que la estructura de las bases de datos de producción puede no ser la óptima para la obtención de la información deseada, obliga a la aparición de una nueva estructura en la información: el **Data Warehouse, que consiste en una "réplica masiva de los datos"** disponibles en las bases de datos operacionales de tal forma que su estructura ya no responda a las necesidades del modelo relacional puro, puesto que sobre ella no se van a efectuar las operaciones que se hacen sobre las primeras. Así, ya no existe necesidad de mantener una estructura de la información que esté normalizada, con lo que supone en ahorro de fraccionamiento de la información y las costosas operaciones de recuperación (join entre tablas obtenidas como resultado de la normalización). Por tanto, nos encontramos con una "réplica" conjunta de todas las bases de datos operacionales.



Hasta aquí parece que se ha definido un data warehouse como una simple copia de los datos y que por tanto no aporta nada nuevo. Sin embargo la problemática asociada a la obtención y tratamiento que de sus datos se hace convierten al data warehouse en una nueva estructura con una problemática asociada muy concreta y diferenciada de las bases de datos de producción.

La definición más extendida es la de Inmon y Hakathorn en "*Using the Data Warehouse*" (1994) que dice que "**El Data Warehouse es una colección de datos orientados al tema, integrados, no volátiles e historiadados, organizados para el apoyo de un proceso de ayuda a la decisión.**"

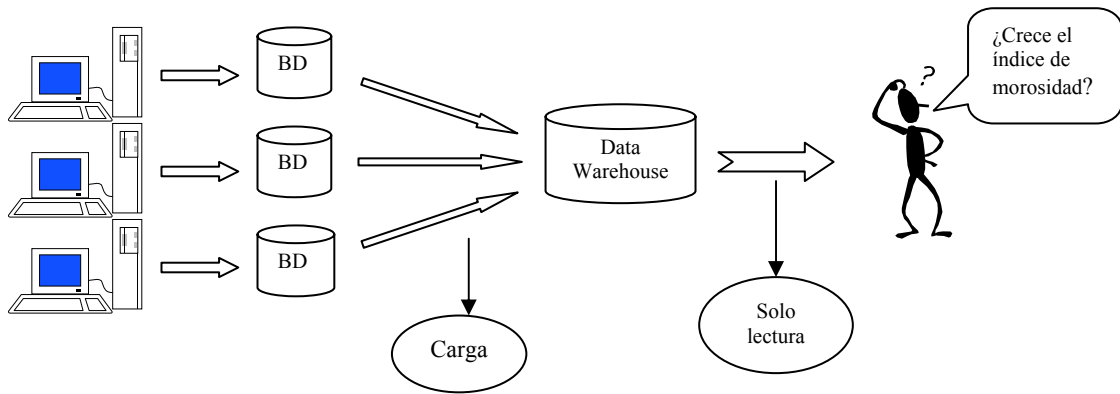
Esta definición incluye el objetivo (ayuda a la decisión) y las principales características (orientados al tema, integrados, no volátiles e historiadados). En base a ella se puede ver claramente que la existencia de un Data Warehouse no conlleva exclusivamente el hecho de que se realice una copia masiva de datos, sino que esa copia tiene un determinado fin y que su propia existencia involucra una dinámica de trabajo diferenciada.

Inicialmente, la primera diferencia estriba en la orientación de los datos y su organización: en el mundo relacional, los datos están orientados a los procesos o transacciones, mientras que en esta estructura, los datos deben estar orientados al tema, entendiendo por tal, los conceptos que tengan relevancia en la organización en la que existe. Por sí solo, este hecho ya implica la existencia de procesos diferentes a los existentes en las bases de datos relacionales. Así, parece evidente que **uno de los procesos característicos de un entorno de DW sea el proceso de "llenado"**, ya que proviene de los datos existentes en las BD de la organización. Este proceso no debe afectar de ninguna manera a los rendimientos de dichos sistemas, con lo que se debe buscar los momentos adecuados en los que realizarlo, lo que **conlleva modificaciones en la "mecánica de trabajo" de la organización.**

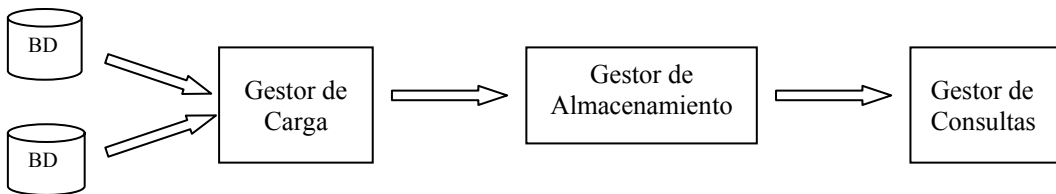
Otra de las características es que, dado que las consultas sobre la base de datos varían respecto de las realizadas en el entorno relacional, los requisitos varían con ellas y, por tanto, obliga a variar el diseño de la base de datos. Este diseño debe orientarse a las nuevas necesidades de tiempo de vida útil mínimo de la base de datos y a otra de las características: que los datos son no volátiles, lo que hace de **la fecha un atributo importantísimo** a la hora de convertir en "historiadados" a dichos datos. El usuario del DW no efectúa operaciones de escritura sobre él, ya que solamente se carga con los procesos ya mencionados.

Por otra parte y debido a la "no volatilidad", es de destacar el rápido crecimiento en volumen de datos, lo que obliga al diseño y mantenimiento de políticas de almacenamiento y de realización de **copias de seguridad que sean eficientes.** Por último, hay que decir que el acceso al DW es diferente al tradicional, y que, otra vez debido a ese gran volumen, se debe dedicar un gran esfuerzo a optimizar el acceso a los datos por parte de las consultas. Pequeños fallos de diseño en las consultas adquieren ahora gran relevancia dado que se notará mucho la falta de optimización.

Todo esto nos lleva a poder representar el flujo de información existente en un DW como un flujo lineal que comienza en las bases de datos tradicionales y que tiene como característica que la escritura o actualización se produce en un punto muy concreto y que el otro extremo **solamente efectúa operaciones de lectura:**



Para llegar a tener estas características, es necesario trabajar con una arquitectura que se basa en tres grandes módulos:



Veamos cuáles son las características de cada uno de estos tres grandes módulos:

Gestor de carga

El módulo que se debe encargar de la gestión de la carga del Data Warehouse debe extraer los datos de las bases de datos operacionales. Sin embargo esto plantea una serie de problemas que son inherentes a dicha carga:

- **El primero de estos problemas es el de la integración de los datos**, dado que una situación normal en estos entornos es que cada una de las bases de datos esté soportada por gestores de diferentes fabricantes. Esto puede provocar que un atributo pueda ser de un determinado tipo en uno de los gestores y de otro tipo en otro. Así podríamos encontrar que un título de un informe (o libro, o revista, etc.) podría tener un tipo CHAR(200) en algún gestor y ser de tipo VARCHAR o VARCHAR2 en otro (dado que puede representar ventajas al no ocupar este segundo tipo toda la longitud si no es usada). Por supuesto, este problema puede aparecer incluso aunque los gestores provengan del mismo fabricante, ya que es posible que, al estar diseñadas e implementadas las diferentes bases de datos por distintos equipos/departamentos y en diferentes fechas, es posible que nos encontremos con que, por ejemplo, la longitud de un atributo que contenga el apellido de los clientes en una de las bases de datos es de 20 caracteres mientras que en otra, viendo que en el 95% de los casos la longitud no pasa de 15 caracteres, se haya optado por truncar los apellidos largos y se haya elegido esa longitud. Es evidente que, por trivial que sea, dicha integración supone un trabajo extra.
- El segundo de los problemas que conlleva esta arquitectura es **elegir el momento en que se produce la carga del Data Warehouse**. Lo importante en este tema se centra en que esta extracción se debe realizar en un momento en que **todas las bases de datos estén en un estado “estable”** de tal forma que se minimicen **las posibles incoherencias**, que por otra parte, de producirse, **deben detectarse y corregirse en este punto**, ya que es el único en el que se debe permitir la actualización.
- Asociado a lo anteriormente expuesto está el hecho de que para realizar **la carga no se puede parar la operativa diaria** de la institución ya que, normalmente, es fundamental para su buen funcionamiento. Esto nos lleva a que hay que diseñar y preparar los procedimientos necesarios para minimizar el tiempo destinado a ello (normalmente se dispondrá de ventanas de tiempo bastante cortas y que pueden obligar a cargas parciales). Todo ello lleva a que, **en general, se realiza una carga sobre una zona temporal** sobre la que sí está permitido hacer transformaciones asociadas a la integración y comprobación de

coherencia anteriormente mencionadas. En cualquier caso, el paso a la zona temporal se debe realizar sin ningún tipo de estructura para evitar retrasos indebidos, pero cuando se realice la carga al DW ya deben estar estructurados de acuerdo a su diseño final.

- Adquiere especial importancia también, la **existencia de un buen diccionario de datos** o “metadatos” ya que es absolutamente necesario conocer, de la estructura final del DW, todos los detalles posibles. Esto quiere decir que un diccionario de datos de estas características no puede ser un mero registro de las características principales de los atributos, sino que **debe contemplar todos los detalles tales como el atributo del que proviene, de qué base de datos, qué transformación ha sufrido, por qué es necesario para el DW, etc.**
- Por último hay que tener en cuenta detalles de más bajo nivel que pueden afectar al proceso de carga debido a las características del gestor usado: así, una mejora en la carga la constituye el hecho de **eliminar completamente los índices** de la base de datos para pasar a proceder con la **carga** y, finalmente **volver a generar los índices** (ya que de mantenerse los índices, lo normal es que por cada tupla que se inserta en una tabla el gestor debe proceder a reordenar todos los índices).

Gestor de almacenamiento

Este módulo es el encargado de proceder al almacenamiento y organización de los datos. Aquí se remarca la diferencia con el modelo relacional dado que este último **surge en base a una serie de necesidades de gestión muy concretas** y responde aceptablemente a dichos problemas. En el DW las necesidades varían, los accesos que se producen no tienen nada que ver con el modelo relacional (como se ha comentado anteriormente, no hay actualizaciones que no provengan de la carga). Así, por ejemplo, **no parecen tener mucho sentido** dos de las grandes preocupaciones en el modelo relacional: **la normalización** (que se ocupa, sobre todo, de garantizar que no se van a producir anomalías en las actualizaciones y en este modelo no existen) y **la integridad referencial** (que se supone garantizada por los procesos de integración realizados durante la carga).

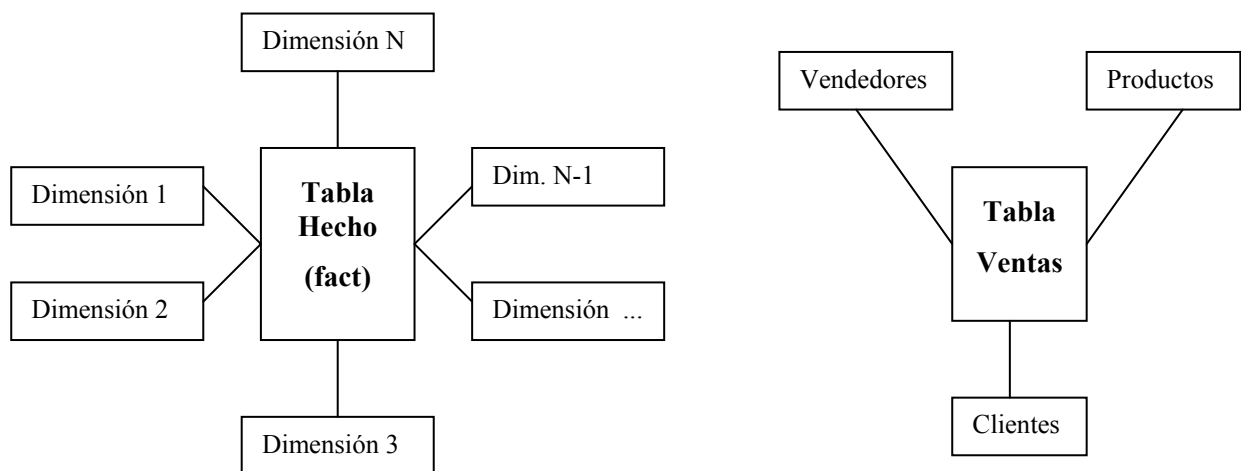
Parece evidente que una de las características del gestor de almacenamiento es la necesidad, no solamente de almacenar los **datos al detalle (datos “raw”)** sino que, dado el tipo de consultas que se van a realizar, de almacenar un “resumen” de dichos datos por diferentes conceptos (comúnmente denominados **datos agregados o globalizados**). La necesidad de almacenar tal volumen de datos hace que el diseño tenga en cuenta que **los datos detallados más actuales aparezcan almacenados en dispositivos rápidos** mientras que **los menos recientes, normalmente, se trasladen a dispositivos más baratos y lentos**, dejando los agregados en los primeros. Los datos agregados deben figurar también en el diccionario de datos, con su significado, su método de obtención, etc., de tal forma que se tenga un control exhaustivo sobre ellos para poder hacer un uso racional y eficiente.

El modelo más extendido para representar los datos en este gestor de almacenamiento es el **modelo en estrella (“star”)** que se basa en el hecho de que no todas las entidades (entendiendo como tales los conceptos de interés para la institución) tienen igual número de ocurrencias ni presentan igual curva de crecimiento en el volumen de dichos datos. Asumiendo que el modelo está soportado por un modelo relacional (que no tiene por qué ocurrir a pesar de que, hasta ahora, es lo más usual y probablemente lo más eficiente y práctico), se observa claramente que, en una situación normal, las **tablas que presentan un mayor crecimiento** en cuanto al volumen de datos son aquellas que **provienen de las relaciones del modelo Entidad/Relación**, mientras que las tablas que provienen de las entidades sí que presentan una variación en el tiempo pero que, en proporción a las primeras, se puede decir que “son estables”. Por lo tanto, aparece una **“tabla”, denominada “tabla hecho” (o tabla ‘fact’ usando el término anglosajón)** que es el centro del modelo dado que es el concepto bajo el cual se desea estudiar el DW. Así, en el caso de unos grandes almacenes que quisieran estudiar las ventas producidas, la tabla “hecho” reflejaría las transacciones realizadas, que probablemente en un modelo Entidad/Relación aparecerían en la tabla generada en base a la relación existente entre las entidades cliente, producto y empleado (por ejemplo). Como se intuye, ninguna de estas tres

entidades debe de presentar una variación considerable en cuanto al volumen de los datos que soporta. Es evidente que tanto el número de empleados y productos presentan altas y bajas y que un objetivo de los grandes almacenes será tener más clientes, pero en ninguna de esas tablas se insertarán tantos datos como en la de ventas. Como se puede suponer, el crecimiento de las tres tablas de entidades se podría considerar "nulo" en comparación con el crecimiento del número de ocurrencias (tuplas en el modelo relacional) de la tabla proveniente de la asociación entre ellas.

Por otra parte, es más que probable que dichos grandes almacenes deseen estudiar sus ventas en relación con otros parámetros (por ejemplo en función de los productos o de las características de sus clientes), por lo que aparecen las denominadas **tablas "dimensión" que son aquellas tablas que contienen las tuplas de los conceptos que se desean relacionar con el hecho**. Estas tablas suelen provenir de las entidades que aparecían involucradas en la relación que da lugar a la tabla "hecho" en el modelo Entidad/Relación.

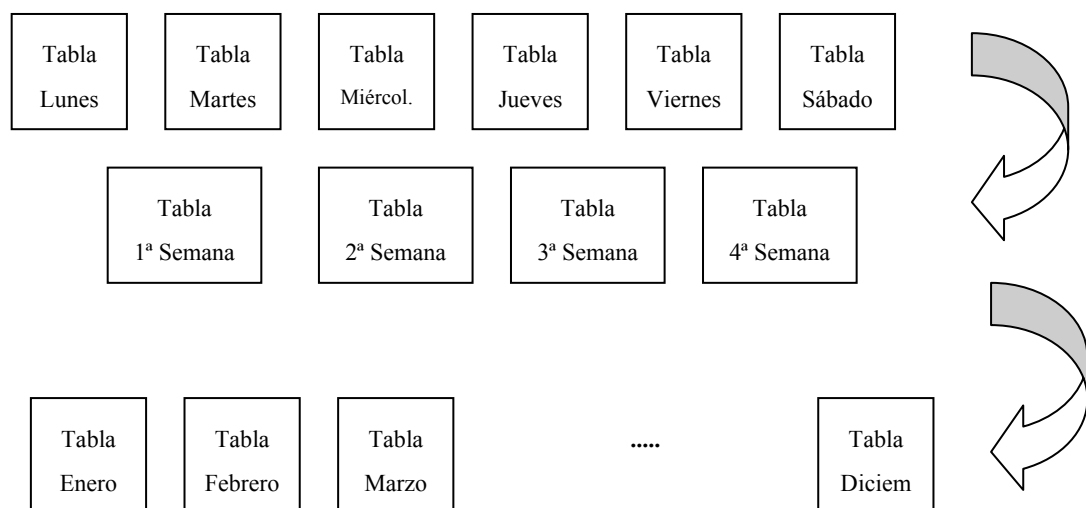
Con estas consideraciones, el aspecto que presenta un diagrama con las tablas descritas es el de una estrella (star), dando nombre al modelo:



Además del aspecto estructural, el gestor de almacenamiento debe tener en cuenta que **en un DW es absolutamente necesario particionar los datos**. Es un problema fundamental del diseño definir la política de partición, pero, dado el volumen de datos que se maneja, en ningún caso se habrá de poner en duda su necesidad. El objetivo de esta partición es la de conseguir mayor eficiencia en las consultas al no tener que manejar todos los datos, por lo que dicha política se **aplica exclusivamente a la tabla "hecho"** y no a las tablas "dimensión". Es importantísimo no confundir esta partición horizontal con la existencia de "datamarts" (que se verá más adelante) o con el hecho de que todo el DW esté soportado por un gestor que permita una base de datos distribuida. Hay que hacer notar que **el DW es una base de datos "centralizada"** desde el punto de vista de que la visión del modelo es global y como tal se considera, independientemente de que en su implementación se utilicen estructuras distribuidas. El criterio que se elija para realizar la partición horizontal es importantísimo y afectará grandemente al diseño y eficiencia del sistema completo. Como **criterio general** es aconsejable la utilización de los atributos que determinan **la fecha** para realizar dicha partición. El motivo de esto es que la elección de cualquier otro factor que pueda llegar a variar en el tiempo (aunque a priori no parezca que vaya a hacerlo) puede provocar errores. Así, si se utiliza como criterio una partición horizontal por departamentos, es posible que una reestructuración no planificada de la empresa lleve al desastre a la organización del DW: si se suprime un departamento, todos los datos almacenados de dicho departamento (y hablamos de un volumen de datos enorme) estarán indebidamente clasificados y, mucho más grave, los datos agregados no serán fiables. Por tanto, **el único criterio que es invariable es el tiempo (el mes de febrero será "mes de febrero" este año, el que viene y el siguiente ...)**. Aún así, se debe tener especial cuidado con el criterio de "tiempo", dado que es muy posible, volviendo al caso de unos grandes almacenes, que el volumen de ventas no tenga una distribución uniforme a lo largo de todo el año y así,

probablemente en época de rebajas se produzcan más del doble de ventas que en otro momento. Incluso teniendo en cuenta una distribución uniforme, dadas las características de la utilización del DW, podría ocurrir que las consultas requeridas presenten una división temporal que no es la tradicional (ventas antes del mediodía, por promociones, horarios comerciales, etc.).

Relacionado con el tema de las particiones y su almacenamiento surge la problemática asociada a la **política de granularidad** elegida, esto es, ¿hasta cuándo y qué datos se almacenan al detalle? Estadísticamente se demuestra que **durante los primeros seis meses de vida de un DW se accede al 90% de los datos almacenados**, pero a medida que se van incorporando nuevos datos de forma masiva se tiene que, **al cabo de un año, se accede solamente al 50% y al cabo de dos años al 10% de media**. Esto quiere decir que la utilización de **datos antiguos** es escasa y que probablemente se puede obviar el detalle para **dejar almacenados solamente los datos agregados**. Con estos datos parece bastante adecuado escoger una política que, en vez de almacenar todos los datos al detalle, opte por hacer un **"rollup" de los datos, almacenando en detalle los más actuales y acumulando los antiguos**. Así, en el caso de las ventas se podría tener una tabla para las ventas de cada día de la semana y una vez que se termina la semana, acumular los datos de toda la semana en una tabla, almacenando una para cada semana del mes y, una vez terminado el mes, agregar los datos en una tabla para cada mes:



Evidentemente, este tipo de políticas debe trabajar con un cierto margen de seguridad para tener la certeza de que no se pierden datos que van a ser consultados en detalle (por ejemplo se podría almacenar en detalle dos de los plazos requeridos: almacenar, en el ejemplo, dos semanas para pasar a la tabla de semanas la más antigua).

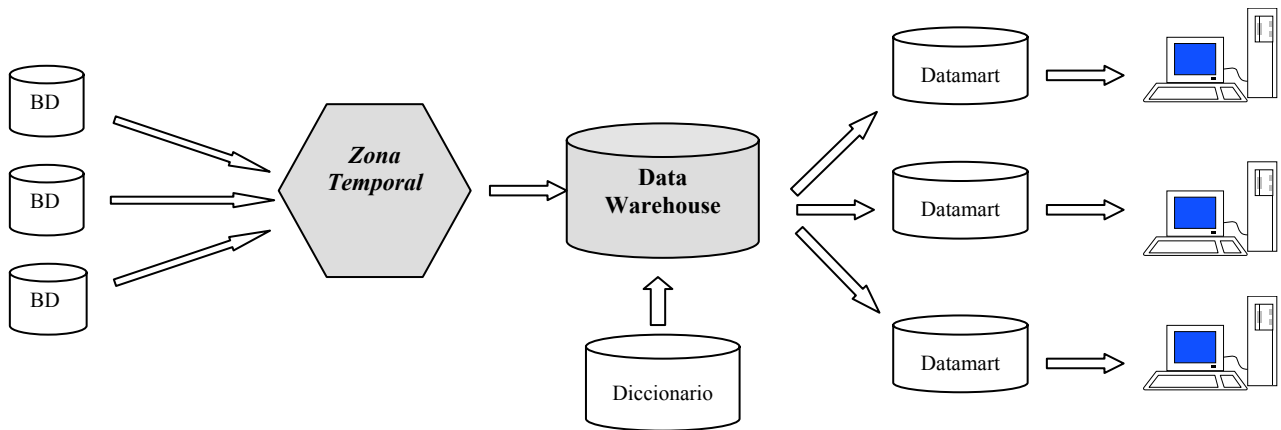
Gestor de consultas

El gestor de consultas es una parte muy importante de cara al usuario. Hoy en día, cualquier gestor relacional dispone de un optimizador que interviene antes de la ejecución de las consultas, de tal manera que las transforma y ejecuta los procesos necesarios para que éstas se procesen de la manera más eficiente. **El gestor de consultas es al DW lo mismo que el optimizador al modelo relacional**, pero mucho más complicado. Este gestor no solamente debe tender a optimizar estas consultas en su ejecución, sino que para hacerlo **debe capturar los diferentes perfiles de consultas** que presentan los usuarios, de tal forma que pueda capturar y dirigir las consultas a los agregados ya existentes y que reflejen la información solicitada. Para esto vuelve a adquirir importancia vital la existencia de un diccionario de datos activo y avanzado, que permita a este gestor la obtención de información acerca de dichos agregados.

Este módulo debería, a su vez, ser capaz de **determinar la utilización** que se está haciendo de **los agregados**, de tal forma que fuera capaz de **crear nuevos** o, ante la falta de utilización de alguno en concreto, **darlo de baja**.

Por otra parte, aparece, con el estudio del perfil de las consultas, el **problema asociado al interés por los datos "locales"**. Volvamos al caso de la entidad bancaria: a cada uno de los directores le interesan solamente los datos asociados con su sucursal, de tal forma que los otros datos lo único que le suponen es un retraso considerable en la obtención de la información. Sin embargo, por otra parte, a directivos de más alto nivel le interesan datos más globales; así a un responsable de zona le interesarán los datos globalizados de todas las sucursales de su zona, etc. Para solucionar este problema de eficiencia se diseñan los denominados **"datamarts"** que son **conjuntos de datos que se corresponden con un determinado perfil de consulta** y que podrían ser considerados como "pequeños Data Warehouse" de cada uno de los grupos. Hay que hacer notar otra vez que esta división o creación de nuevos conjuntos de datos, nada tiene que ver con la partición horizontal propugnada en el almacenamiento o con el hecho de que estos datos puedan estar almacenados de forma distribuida (que podrían estarlo o no).

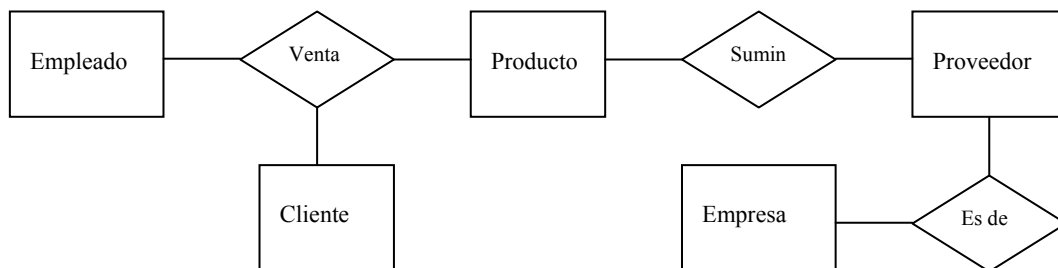
Con todo lo dicho, el aspecto que presenta un sistema de Data Warehouse bien podría ser:



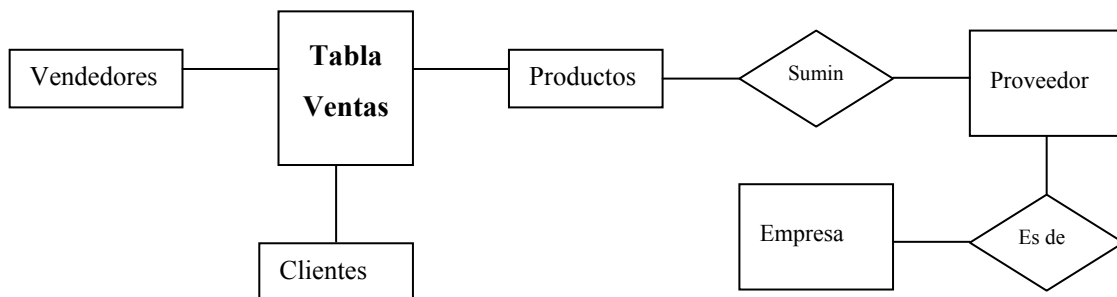
Consideraciones de diseño

El diseño de un DW debe estar orientado a optimizar las consultas relacionadas con los aspectos del negocio que se desean estudiar. Tal y como se planteó anteriormente, esto conduce a una estructura en estrella en la que el centro es la tabla "fact" o "hecho" que representa al factor principal por el que se desea analizar la base de datos. Alrededor de esta tabla aparecen las tablas "dimensión", que representan los diferentes aspectos relacionados con el principal y que influyen en el estudio.

Dado que lo habitual no es diseñar el DW a partir de cero, sino que **se suelen comenzar desde el diseño de las bases de datos operacionales de las que se dispone**, el aspecto final del diagrama no es realmente el de **una estrella** sino que **presenta "flecos" (lo que da el nombre de "starflake" o "snowflake" al diagrama resultante) que pueden "descentrar" a la tabla de hechos**. Así, si la tabla de hechos se refiriese a las ventas y una de las tablas de dimensión fuese la de productos, y otra la de empleados que realizan la venta, es posible que se arrastrase al diagrama parte del modelo que proviene del Entidad/Relación inicial por lo que podría presentar un "desequilibrio" hacia los proveedores de dichos productos:



Daríamos como resultado al pasarlo al diseño del DW:



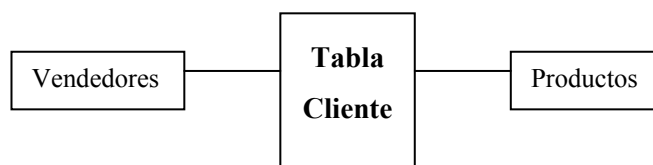
que, como se puede ver, no aparece centrado, sino que presenta "flecos" por arrastrar la parte del Entidad/Relación correspondiente a los proveedores, a pesar de que no van a ser, a priori, aspectos por los cuales se van a estudiar las ventas.

Entre los aspectos a tener en cuenta al afrontar el diseño de un DW hay que tener especial cuidado al:

- **Identificar las tablas de hechos**, ya que es posible tener más de una. Por cada aspecto del negocio que interese estudiar debe aparecer una tabla de hechos.
- **Identificar las tablas de dimensión** (esto es, decidir cuáles son los parámetros por los que interesa realizar el estudio).
- **Comprobar que ninguna de las tablas de hechos oculta tablas de dimensiones**. Al heredar la estructura de las bases de datos operacionales, esto ocurre muy a menudo al encontrarnos que no se han eliminado atributos que ya no interesan.

- **Comprobar que ninguna de las tablas de dimensión oculta una tabla de hechos.** Esto conduciría a la tabla a un crecimiento anormal muy por encima de los límites aceptables para este tipo de tablas (por otra parte, este síntoma ayuda a identificar el error cometido en el diseño).

Las tablas de dimensión no presentan una participación importante a efectos de alterar el rendimiento del sistema por cuanto, en general, el peor de los casos nos lleva a que nos encontremos con tablas de más que no se utilizan, pero que, dado el escaso crecimiento, no afectan al rendimiento. Por otra parte, las tablas de hechos sí son fundamentales y, como se ha planteado anteriormente, provienen, en general, de las tablas de las relaciones del modelo Entidad/Relación. A pesar de esto, sí es importante tener en cuenta que **puede haber tablas que unas veces participen en el diseño del DW como tablas de dimensión y otras veces como tablas de hechos.** Así es posible que en el ejemplo anterior, el primer diseño interesante sea el ya planteado, con la tabla de ventas como tabla de hechos, pero también es posible que interese un estudio pormenorizado de los clientes y su naturaleza de acuerdo con los productos, etc. ...



En general, una tabla será de hechos si representa a alguno de los aspectos de negocio que se desean estudiar, mientras que lo será de dimensión si representa a uno de los factores por los cuales se desean estudiar los anteriores aspectos.

Otro punto importante a tener en cuenta a la hora de acometer el diseño de un DW es el de los **atributos incluidos en las tablas.** Estos atributos deberían cumplir una serie de características como son:

- **No deben aparecer atributos que no interesen.** Esto representa un serio problema de compromiso a la hora de elegir los atributos que se deben incluir en las tablas de hechos, ya que no se deben incluir atributos "por si son necesarios algún día" porque se incrementa el tamaño de la tabla de una manera espectacular con su lógica repercusión en la eficiencia. Sin embargo, por otro lado, la falta de atributos en estas tablas pueden llevarnos a disponer de un DW cuyo análisis no es válido al no aparecer toda la información significativa.
- Se deben elegir los **tamaños adecuados para cada uno de los atributos.** Una de las costumbres en el modelo relacional es la de incrementar el tamaño necesario de un atributo "por si se necesitase" (por ejemplo, no se considera importante si a un nombre se le reserva un tamaño de 25 o de 30 caracteres). En este caso, dado el volumen de datos que trata, el **ahorro** de unos pocos bytes en cada una de las ocurrencias nos lleva a un gran ahorro en el tamaño global del DW.
- En contra de lo habitual, la discusión sobre la elección de "claves inteligentes/no inteligentes" en la tabla de hechos, se decanta, en este caso, por la segunda opción. **Se deben incluir claves artificiales que no guarden ninguna relación con su significado semántico.** Es cierto que esto conlleva la **obligación de realizar operaciones de "join" con las tablas de dimensión para poder interpretar dicho significado, pero evita la incoherencia que se produciría si el contenido de las claves variase algún día** y, por una parte hubiera que actualizarlo en todo el contenido de la tabla de hecho, y por otra, se perdiera información al encontrarse parte de ella ya almacenada en forma de resultados agregados/resumidos y su actualización resultase imposible.
- Como **caso especial** de atributo, existe una variante fundamental respecto del modelo relacional: el tratamiento que de la fecha se hace. **La importancia que adquiere en un DW la fecha es enorme,** ya que los análisis de los datos se suelen referir a intervalos temporales muy definidos. Por tanto, lo primero es que la fecha debe estar almacenada como tal en la

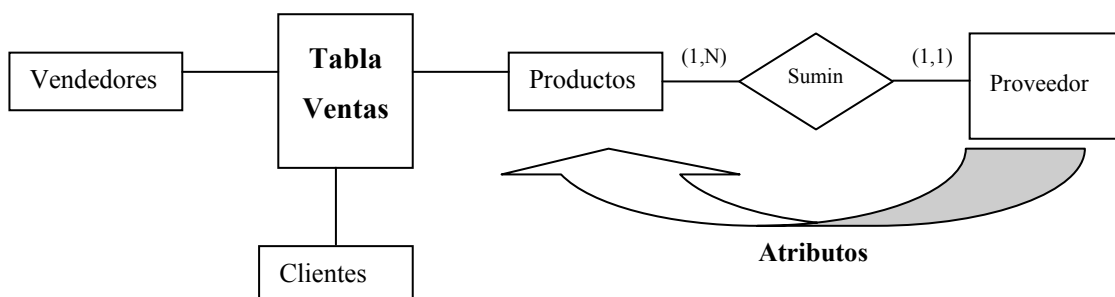
tabla de hechos y **no con un código artificial**. Un aspecto fundamental es el de cómo se almacena dicha fecha. Las diferentes opciones presentan ventajas e inconvenientes: almacenar la **fecha completa evita la necesidad de cálculos** pero ocupa más espacio; almacenar únicamente el **"offset temporal" producido desde la fecha por la cual se ha realizado la partición** ahorra espacio, pero afecta al rendimiento al necesitar de **cálculos adicionales** a la hora de mostrar una fecha completa; y por último, **almacenar intervalos** es una posición intermedia que sin afectar gravemente al rendimiento, tampoco ocupa tanto espacio. Lo habitual, sin embargo, es proceder al almacenaje de la fecha completa.

Como última consideración en cuanto al diseño de las **tablas de hechos** está el problema asociado a la partición. Hay que volver a destacar que éstas tablas **siempre se particionan**, presentando únicamente como duda el criterio a seguir para dicha partición. Tal y como ya se ha planteado, el criterio que mejor se adapta a las necesidades de "no variación" de un DW es el de particionar por fecha. Sin embargo esto puede plantear problemas con los diferentes tamaños generados, dado que es posible que el número de ocurrencias no presente una distribución homogénea a lo largo del tiempo (por ejemplo, las ventas en época de rebaja son mayores que en otras épocas). Por esto, es común la **utilización de políticas acumulativas** como la expuesta anteriormente (agregar datos a lo largo del tiempo, teniendo, por ejemplo, una tabla para cada día de la presente semana, una tabla para las cuatro últimas semanas y una tabla para cada uno de los doce meses del año, de tal manera que en cada plazo se agreguen los datos a las tablas correspondientes).

En caso de que los datos que se manejen hagan que la partición por fechas no sea válida (por ejemplo: el manejo de información de la bolsa, que es demasiado dependiente del momento) **se pueden usar otros posibles criterios** como las particiones de igual tamaño (cuando una partición excede un límite se acude a almacenar datos en la siguiente); particiones correspondientes a distribuciones geográficas del negocio (una partición por sucursal o departamento), etc. El **problema básico** de usar otros criterios es el de la **localización de información correspondiente a otros momentos** ("¿dónde está la información de hace tres meses?").

En cuanto a las **tablas de dimensión**, presentan como características fundamentales que no tienen tamaños muy grandes (siempre en proporción al tamaño de la tabla de hechos), y que no van a incrementar ostensiblemente dicho tamaño a lo largo del tiempo (si lo hacen, probablemente habremos cometido algún error de diseño y ocultan alguna tabla de hechos).

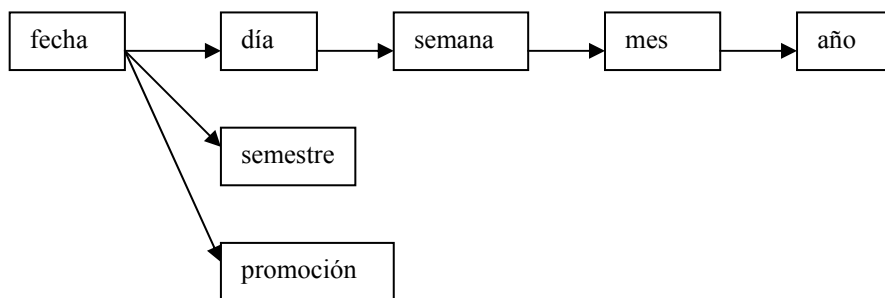
El contenido de estas tablas se importa casi directamente del modelo de datos del Entidad/Relación pero con la particularización de que si aparecen "flecos" en el modelo, se debe tender a **"desnormalizar"** las tablas que presenten relaciones de **cardinalidad 1:N**, de tal forma que se incorpore toda la información que se necesite en la tabla correspondiente:



Es evidente que, en este ejemplo, la "desnormalización" de los datos procedentes de los proveedores, pasando dichos datos con la clave, a la tabla de productos, mejora sensiblemente la eficiencia de las consultas dado que evita hacer los correspondientes join con la tabla de proveedores.

Otro factor a tener en cuenta es el hecho de intentar **evitar que las tablas de dimensión presenten intersecciones no vacías con las tablas de hechos** (posible dado que puede haber una entidad que participe como tabla de dimensión en un "star" pero que sea el aspecto de negocio a estudiar en otro). **El gestor, al ejecutar las consultas, podría elegir la tabla de hechos para hacer el join en vez de la tabla de dimensión**, lo que daría lugar a un tiempo de proceso enorme.

Por otra parte, los análisis y estudios que se realizan de los datos de un DW **incluyen consultas del estilo de "¿Cómo se vieron afectadas las ventas durante esta promoción?" o "¿Se ha vendido más por la mañana o por la tarde?" o "¿Se ve afectado el volumen de ventas por el día de la semana?"**. Este tipo de consultas incluye, en un entorno relacional tradicional, operaciones típicas en los atributos que hacen referencias a fechas con cálculos de "que día de la semana ha sido una determinada fecha", etc., que, de aplicarse a cada una de las ocurrencias de un DW, afectarían gravemente a la eficiencia. Por tanto, probablemente convenga **realizar un diseño "a medida" para cada uno de los casos, que contemple la estructura de la fecha más conveniente**. Así, por ejemplo, si quisiéramos calcular el día de la semana, sería lógico pensar en que se podría tener una tabla con todos los días del año y el día de la semana que ha sido, ya que su tamaño es pequeño en proporción al DW (como máximo 365 tuplas por año) y ahorra muchísimo tiempo de cálculo. Esto podría llevarnos a tener unas tablas de dimensión que presentasen una jerarquía de atributos respecto de las fechas que sería implantable en el modelo relacional aplicable a gestión tradicional (bien podría ser el modelo necesario para el estudio del comportamiento de las audiencias televisivas por franjas horarias a lo largo del año):



Otro aspecto importante en el diseño de un DW es la **inclusión de tablas que contengan los datos ya agregados**, dado que ese será el tipo de consulta más común. Esto nos lleva a la inclusión de tablas de hechos más reducidas en cuanto a su tamaño. Para hacer esto se pueden plantear dos filosofías: **a) la tabla incluye en su nombre la dimensión por la que se agrega** (lo que hace que se pierda una dimensión) y contiene los datos agregados. En este caso es posible que se puedan eliminar otras dimensiones que no tengan sentido o interés. **b) la tabla contiene una consulta sobre la tabla de hechos**, de tal forma que es un filtro y se consigue una generalización en cuanto a que no presenta los posibles problemas de actualización de la primera solución.

En cualquier caso, el **gestor de consultas debe "monitorizar" las tablas de agregados** de tal forma que detecte la utilización que de ellas se hace para eliminarlas cuando no se usen o proponer la creación de alguna más si detecta que mejoraría la eficiencia global del sistema. Por otra parte, en estas tablas sí que es recomendable la utilización de los códigos y claves con significado semántico dado que su vigencia es absolutamente temporal.

Como último aspecto de este tema cabe destacar la gran importancia que presenta la existencia de un buen diccionario de datos que pueda ser usado de forma "inteligente" por los diferentes módulos y en especial por el gestor de almacenamiento y el gestor de consultas. Para ello debería presentar un mínimo de datos:

De cada tabla fuente de datos:

- Nombre.

- Atributos, nombre y momento en que se carga cada uno.

Para cada tabla del DW que contenga datos Raw (detalle):

- Significado de cada uno de los atributos.
- De qué atributos proviene en las bases de datos operacionales
- Métodos utilizados en su transformación.
- Nombre del proceso que lleva a cabo la transformación.
- Condiciones para la ejecución de los procesos anteriores.
- Políticas de particionamiento en la tabla.
- Vigencia de los datos

Para cada tabla de agregados:

- Consulta que la crea
- De qué tablas proviene (momento de actualización)

Para cada usuario:

- Perfil/permisos

Para cada consulta almacenar su ejecución, momentos, estadística, etc.

Etc.