Datamart de UBTicket

Objetivo

En la entrega anterior habíamos hecho una aplicación para venta de tickets de espectáculos. Los datos de estas transacciones quedaban persistidos en una base de datos relacional.

Ahora lo que se nos pide es hacer un Datamart que nos permita analizar el rendimiento económico. Para ello, lo primero que hemos hecho es un esquema conceptual y la estructura física del cubo multidimensional que contendrá todos los datos agregados que necesitaremos.

Acto seguido, con el diseño listo, podemos crear todas las tablas necesarias dentro de la base de datos PostgreSQL, para después utilizar Mondrian Workbench para crear el diseño lógico del cubo, que hará un mapa entre las tablas físicas y el modelo abstracto del datamart.

Por último, configurar jPivot para que interactúe con la base de datos relacional, y use el diseño lógico para que pueda lanzar consultas MDX sobre el cubo, y podamos recoger datos.

Los datos acumulados (en un proceso ETL no declarado) son las ventas hechas entre el año 2010 y 2013.

Recursos

- Mondrian
  - Mondrian Workbench
- PostgreSQL
- Python

Aunque no es estrictamente necesario Python para la ejecución de la práctica, sí que lo necesitamos a la hora de generar las sentencias SQL a ejecutar para llenar la información de las dimensiones **Tiempo** y **Edad**.

En el entregable presentado hay un script SQL que debe ser ejecutado sobre la base de datos PostgreSQL para poder tener un entorno estable con el que poder jugar con Mondrian.

Para configurar jPivot, hay que copiar el contenido de la carpeta **mondrian** dentro de:

(CARPETA_RAÍZ_JETTY)\webapps.demo\mondrian

La carpeta **data** adjunta en el entregable son los scripts Python usados para generar los datos, y un fichero Excel con el que poder hacer consultas rápidas sobre algunos valores de la tabla de hechos.

Diseño del Datamart

Cambios en el modelo relacional

Antes de comenzar con el diseño del cubo, hemos hecho algunos cambios con respecto a la otra práctica sobre el modelo relacional. Hemos añadido una nueva entidad, **Ubicación**, que nos permitiría guardar de una forma más estructurada el lugar donde reside un usuario, o la localización de un espacio donde se harán espectáculos.

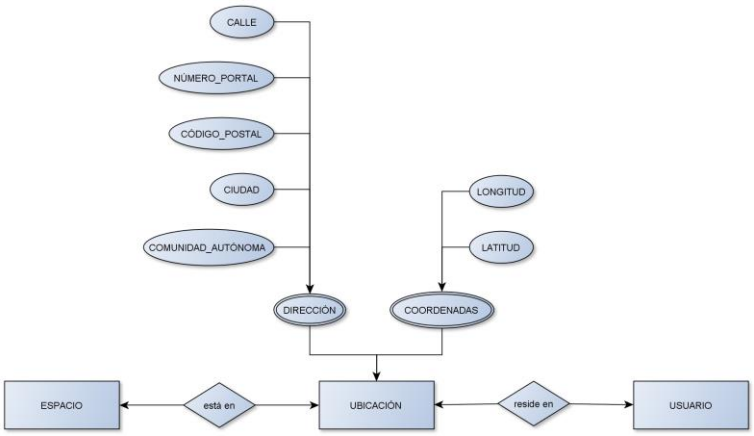


Figura 1. La entidad **Ubicación** y su relación con **Espacio** y **Usuario**.

A efectos prácticas, los datos que requerimos son los atributos *COMUNIDAD_AUTÓNOMA* y *CIUDAD* para el cubo, pero para ser coherentes, hemos agregado otros campos que serían necesarios para especificar la ubicación de una residencia o lugar de espectáculos.

Los otros cambios que se han hecho han sido sobre **Entrada**, dónde se ha agregado un nuevo atributo *FECHA_VENTA* y sobre **Usuario**, se ha agregado el atributo *FECHA_NACIMIENTO*.

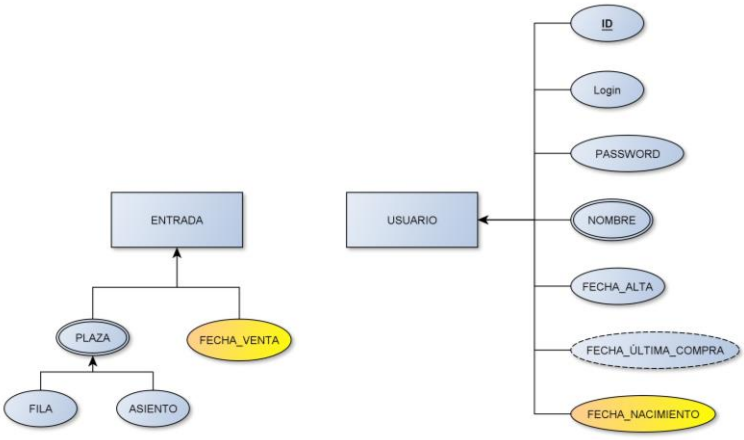


Figura 2. Cambios hechos sobre entidades Entrada y Usuario. Los atributos resaltados son los nuevos atributos agregados

Con este cambio, podemos añadir la dimensión **Edad** al cubo, y podemos tener información sobre la edad de los compradores, y agregarla a otras dimensiones, dado que en tiempo de ETL podemos generar los identificadores que referenciarán a fila de la dimensión **Edad**.

El último cambio que hemos añadido ha sido introducir una nueva entidad **Mantenimiento_Sesión**.

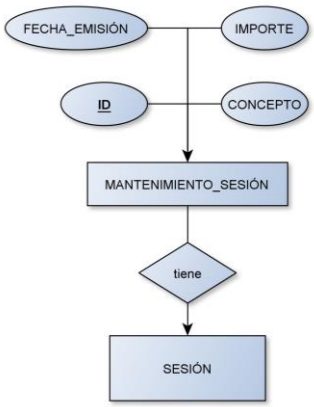


Figura 3. La nueva entidad **Mantenimiento_Sesión**.

Con esta nueva entidad hemos querido introducir la existencia de un coste para mantener una sesión de espectáculo en un espacio. Esta entidad representa las facturas que se generan por la sesión de un espectáculo, y guardamos la cantidad de dinero pagada.

Esto nos permite después calcular en tiempo de extracción la métrica *Coste_Mantenimiento*.

En el anexo incluido en el informe se ha añadido

El cubo multidimensional

Hemos decidido hacer el esquema del datamart en forma de estrella, en principio, por simplicidad de diseño, pero también por flexibilidad a la hora de agregar las dimensiones cuando hacemos consultas al cubo.

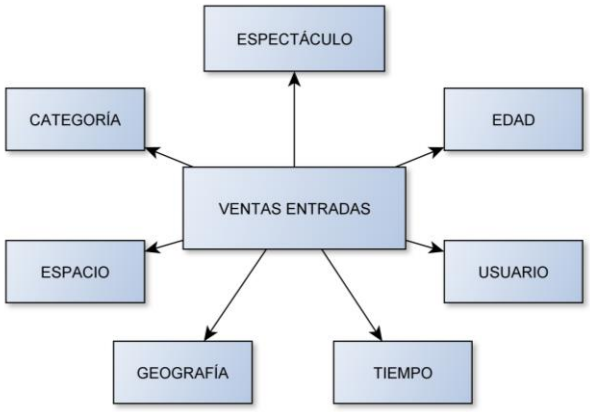


Figura 4. Esquema en estrella del datamart

En la figura 4 se puede ver un esquema de la tabla de hechos, y las dimensiones disponibles. A continuación, vamos a describir estas dimensiones.

Dimensiones

Espectáculo, Espacio, Categoría, y Usuario

Estas dimensiones son bastante directas con respecto al modelo transaccional. Por cada una de estas entidades hemos creado una dimensión con una jerarquía de un solo nivel, que es el nombre del espacio, categoría, o el de usuario, o el título del espectáculo.

Espacio

En el único nivel de su jerarquía le hemos añadido la propiedad *Aforo*.

Usuario

Del mismo modo que en espacio, hemos añadido dos propiedades: *Login* y *Tipo de usuario*. Aunque no son necesarias, siempre puede sernos útil tener a mano estos datos para consultar.

Tiempo

Esta dimensión se ha generado con una jerarquía de 4 niveles:

- Año
- Trimestre
- Mes
- Día

De esta manera, se guardan los datos a nivel de día en la tabla de hechos.

Geografía

Hemos diseñado esta dimensión con dos niveles de jerarquía:

- Comunidad autónoma
- Comarca

Con esta jerarquía, somos capaces de agregar las ventas hechas para una **Sesión** de **Espectáculo** en un **Espacio** que está en una *Comarca* de una *Comunidad autónoma*.

Edad

Con esta dimensión podemos ver el tipo de público que asiste a los espectáculos. Por simplicidad del diseño, no observaremos la discrepancia que hay en el hecho de una persona que compre varias entradas para otras personas. Se ha jerarquizado en dos niveles:

- Rango de edad en decenas de años
- Edad

"Ventas Entradas", la tabla de hechos

Esta tabla de hechos guarda, además de las claves foráneas de cada una de las dimensiones, dos métricas que nos permiten ver el rendimiento económico:

- Ingresos ("Entradas Ingresos"). Esta métrica es el sumatorio de todo el dinero recaudado en ventas de entradas diarias.
- Entradas vendidas. Esta métrica es el sumatorio de la cantidad de entradas diarias vendidas.
- Coste Mantenimiento. Esta métrica es el sumatorio de costes de mantenimiento que hay al tener una sesión de espectáculo.
- Ganancia. Esta métrica calculada nos indica el superávit o el déficit de ingresos de ventas de entradas.

Con esto, ya tenemos definido el esquema del cubo, y podemos definir el modelo físico, que se puede ver en la siguiente figura.

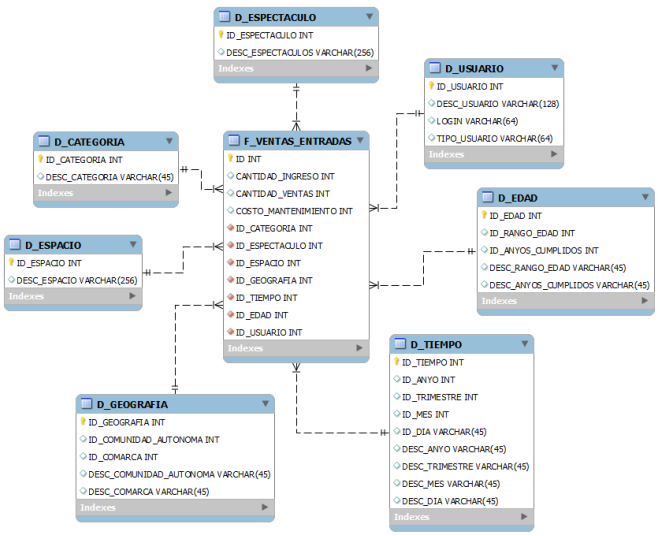


Figura 5. Diagrama de la estructura física del datamart.
(En el anexo hay un diagrama a escala más grande)

Anotaciones sobre el diseño

Particularidades de la métrica Coste_Mantenimiento

Aunque en principio no hacía falta definir ningún proceso de ETL, porque no tenemos datos de los que partir, sí que hay algunas consideraciones a tener en cuenta a la hora de generar los datos de nuestra tabla de hechos.

En primer lugar, la interpretación que le damos a cada una de las filas de la tabla de hechos:

Cada fila se considera las entradas compradas (tanto cantidad de entradas, como la cantidad de dinero generada) por un usuario que tiene una edad, para un espectáculo de una categoría realizado en un espacio de una zona geográfica concreta.

Entonces, podemos pensar que tendremos una fila por cada usuario, que ha comprado entradas para un espectáculo en un espacio. Sin embargo, si miramos con atención en la frase, ¿qué sucede con los costes de mantenimiento de una sesión?

Una primera aproximación es que si queremos agregar los costes de mantenimiento para un espectáculo de un día, lo podemos hacer de la siguiente manera:

$$Costes\ Matenimiento = \frac{Coste\_Mantenimiento}{N}$$

Donde *Coste_Mantenimiento* sería la suma de *Importes* de la entidad **Mantenimiento_Sesión**, y *N* sería el número de usuarios que compraron entradas ese día.

Pero inmediatamente surge otra cuestión: ¿qué sucede si en un día nadie compra entradas, y sin embargo, sí que se han generado facturas? ¿Cómo solucionamos este problema?

Simple. Se siguen añadiendo más filas, las cuales tendrán 0 como valor de las métricas de compras, pero la métrica *Coste_Mantenimiento* tendrá la suma de importes de facturas de ese día para ese espectáculo.

Otro problema que también surge es cuando queremos agregar como filas las dimensiones Usuario o Edad, y tenemos visualizada la métrica de *Coste_Mantenimiento*. Veríamos un sumatorio de está métrica, pero que por cada usuario, tendríamos valor 0.

	Medidas			
Usuario	Entradas Ingresos	Entradas Vendidas	Costo Mantenimiento	Ganancia
-Todos los usuarios	3.683	60	1.400	2.283
Oriol	1.324	26	0	1.324
Domenico	2.359	34	0	2.359
Administrador				
Usuario				

Figura 6. Ejemplo del problema

A pesar de esta aparente incoherencia, tiene sentido, porque un usuario no genera costes de mantenimiento a una sesión. Esta métrica solo tiene sentido cuando estamos buscando información sobre los espectáculos, no sobre los usuarios.

Discrepancias entre las dimensiones Geografía, Usuario y la métrica Coste de Mantenimiento

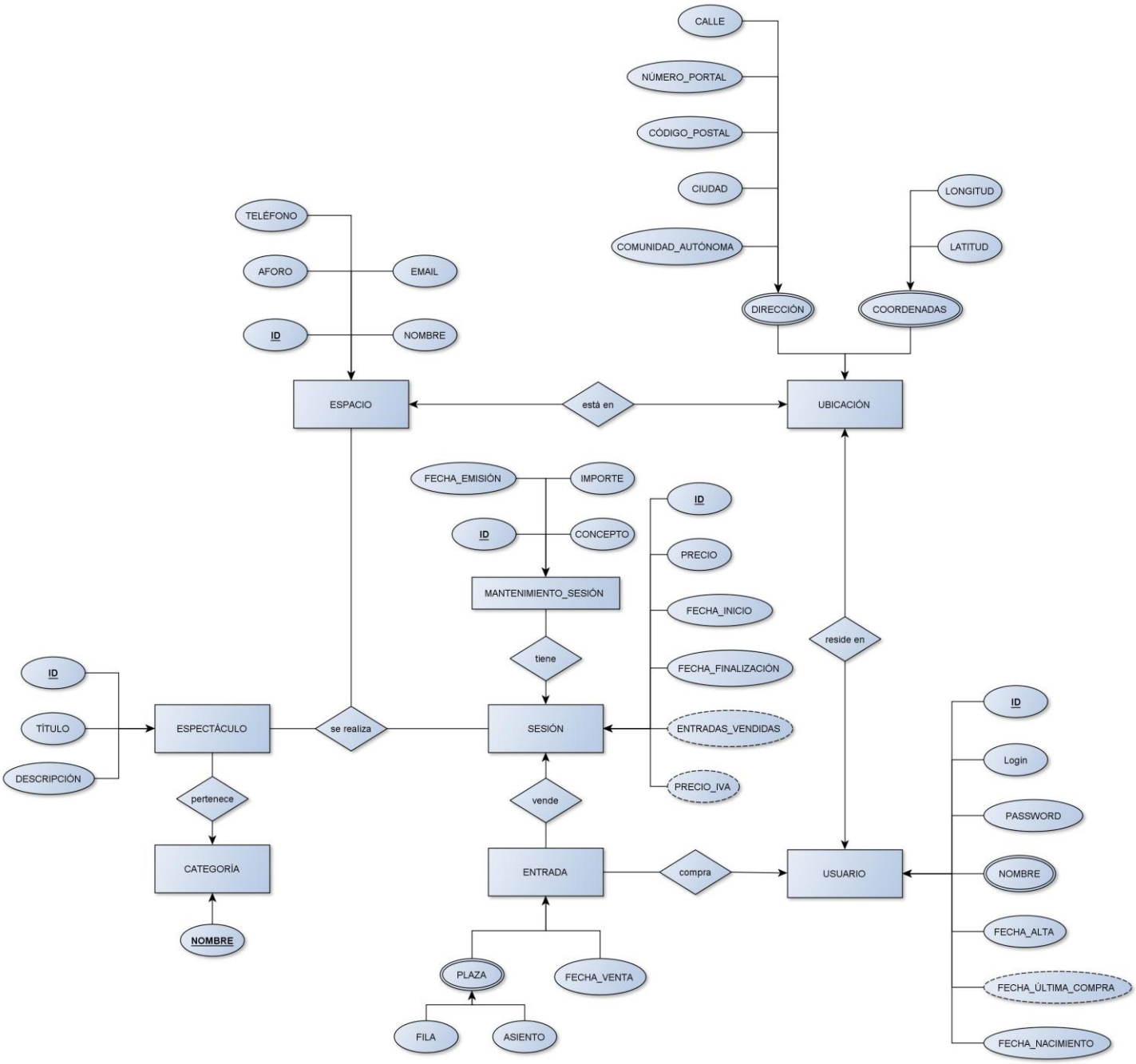
En el cubo presentado, tenemos las dimensiones **Geografía** y **Usuario**, pero son dos dimensiones que a priori nos hace pensar que si las agregamos podríamos obtener métricas de usuarios en una geografía, pero no es así. Lo que obtenemos son métricas de las compras que han hecho los usuarios para espectáculos en un espacio.

¿Y si quisiéramos obtener las primeras métricas mencionadas? En este caso, tendríamos que hacer una serie de cambios:

1. **Publicar todas las dimensiones**. Inicialmente, pensamos que las únicas dimensiones públicas que tendríamos sería **Tiempo** y **Edad**. Pero entonces, todas las demás dimensiones que ahora viven solamente en el cubo de **Ventas_Entradas** deberían ser publicadas y estar disponibles para otros cubos. Lo que nos lleva al segundo paso.
2. **Crear un nuevo cubo, Ventas_Usuarios**. Dentro de este cubo guardaríamos la información propia de las ventas, pero respecto al **Usuario**, en lugar de la **Sesión** de **Espectáculo**.

Anexo

Modelo transaccional



Modelo físico de la BBDD Analítica