Sistemas de Información - Practica 2

Fernando López Berrocal

Índice

Contenido

Preguntas teóricas

A) Identificar el hecho, la granularidad, las dimensiones y las medidas.

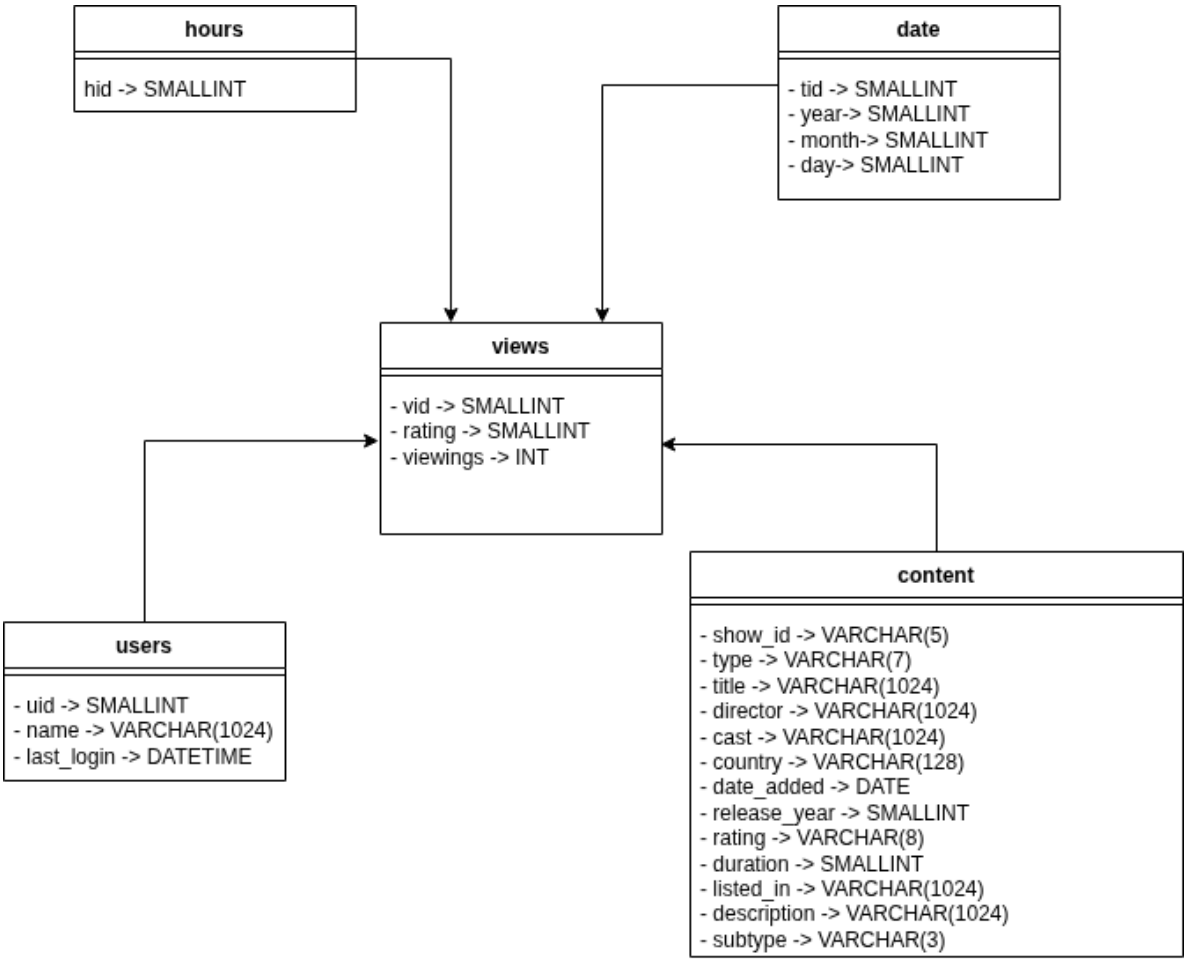
- Hecho: Visionado
  - Valoración
  - Visualizaciones

Dimensiones	Granularidad	Atributos
Contenido	Tipo y Subtipo	Director, casting, duración...
Usuario	No	Nombre, fecha ultimo login
Fecha	Año, Mes, Día	-
Hora	No	-

B) ¿Identificáis alguna fuente de datos externa cuya información sea necesaria para el almacén?

- No se ha necesitado usar fuentes de datos externas

C) Realizar el diseño lógico del almacén.



D) ¿Qué diseño ROLAP usaríais para este almacén?

- Esquema de estrella. La tabla de hechos tiene claves ajenas de todas las dimensiones. Cada dimensión tiene su propia tabla.

E) Realizar el diseño ROLAP.
El almacén de datos se ha implementado en la clase **Data_warehouse** del script **data_warehouse.py** que usa la base de datos **si_prac2_db** como fuente de datos persistente (para mantener los datos de la mínima granularidad después de hacer un roll o un slice & dice)
Los atributos de la clase son Dataframes de panda, uno por cada dimensión y otro para la tabla de hechos. Estos Dataframes cambiarán con los operadores OLAP y se perderán datos de los Dataframes, pero se mantendrán en la BBDD MySQL. También hay un identificador (String) por cada dimensión para saber el nivel de granularidad de cada una.
Slice & Dice y Roll son los dos operadores OLAP implementados en sus respectivos métodos **sliceNdice()** y **roll()**. Los otros operadores no se han implementado por tiempo y porque no se han necesitado en los ejercicios propuestos.

- **sliceNdice():** Toma como parámetros la dimensión sobre la que se hace el slice, el atributo que se va a comprobar y la condición que el atributo debe cumplir (solo comprobara igualdad). Primero se recortará el Dataframe de la dimensión y luego del Dataframe de la tabla de hechos.
- **roll()**: Toma como parámetro la dimensión. Primero se agrupan valores en la dimensión, se cambian las id (ya no tienen sentido las anteriores) y se eliminan las columnas que también dejan de tener sentido. Luego en el Dataframe de la tabla de hechos se agrupan los valores de las medidas: viewings se suma y rating se hace la media. Por último, se cambia el identificador de nivel de granularidad de la dimensión.

Ejercicios Prácticos

Para realizar esta práctica se han realizado 3 scripts de Python, un notebook de Jupyter y 2 bases de datos MySQL en un contenedor Docker.

Las 2 bases de datos son:

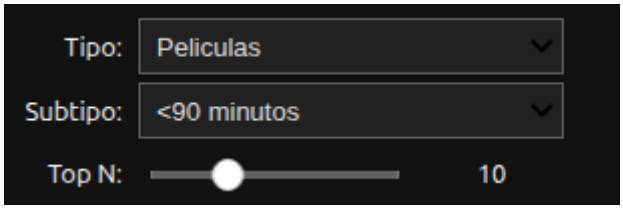
- **si_prac1_db**: Es la BBDD creada en la primera práctica. Representa la base de datos transaccional. Se sigue usando porque es la fuente de datos interna para el almacén de datos
- **si_prac2_db**: Fuente de datos persistente para el almacén de datos.

Los scripts de Python:

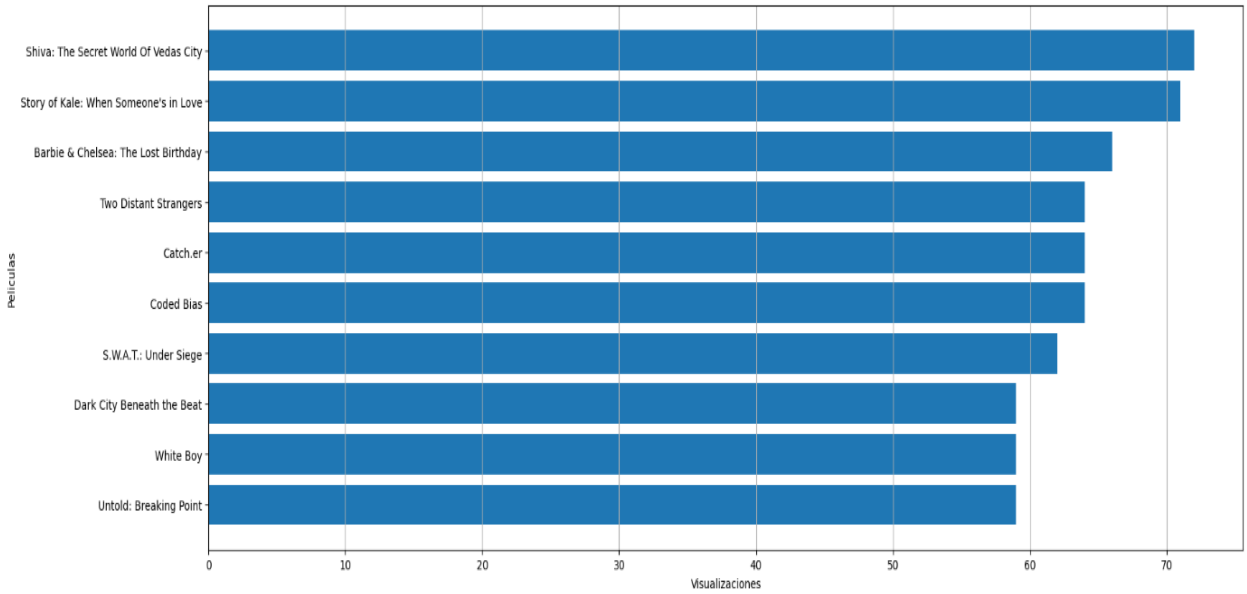
- **generator.py**: Script auxiliar (no existiría en una situación real con datos reales), sirve para crear datos para las tablas de usuarios y visionados en **si_prac1_db**. Los nuevos datos están pensados para que los ejercicios que se realizan a continuación tengan datos sobre los que se pueda trabajar.
  Primero se crean usuarios aleatoriamente (200), con nombres sacados de una lista (nombres de alumnos de la clase y fecha de ultimo inicio de sesión aleatoria. Se insertan en la tabla usuarios de **si_prac1_db**
  Luego se crean e insertan los visionados en la tabla visionados de **si_prac1_db**. Para cada usuario, primero se selecciona que tipo de usuario será (aleatoriamente entre 4 según el número de películas y numero de series que ha visto). Después se termina de rellenar el visionado con la película o series (solo se seleccionan entre las 150 películas y las primeras 150 series) y una valoración aleatoria del 1 al 10.
- **etl.py**: Script que funciona como un ETL. Recoge datos de **si_prac1_db**, luego los transforma y formatea para que sean compatibles en **si_prac2_db** y los introduce en esta.
  Primero se rellenan las dimensiones y luego la tabla de hechos.
  La dimensión content usa la tabla peliculas_series. Hay que añadirle una columna subtype según el tipo y la duración.
  La dimensión users se copia sin cambios de la tabla usuarios
  La dimensión date se debe generar porque no estaba en sistema transaccional. Se generan todos los días de 2024.
  Lo mismo ocurre con la dimensión hours.
  La tabla de hechos views usa la tabla visionados. Hay que añadirle la columna viewings que serán todo 1 al inicio con la mínima granularidad. También se añaden claves hacia dimensiones hours y date, tienen ciertas probabilidades para que en los siguientes ejercicios se pueda analizar algunos patrones.
- **data_warehouse.py**: Contiene la clase del almacén de datos. Explicada anteriormente.
- **cmi.py**: Este script contiene funciones que se van a ejecutar en el notebook de Jupyter. Los ejercicios crearan instancias del almacén de datos sobre las que realizaran operaciones OLAP, así que cada una acabara con una vista diferente.

Ejercicio 2, 3 y 4

Estos 3 ejercicios propuestos se han unido en uno solo. En Jupyter usando **ej1_widgets()** primero se mostrarán unos desplegables para seleccionar el tipo (Películas, Series, o ambos) y luego el subtipo (<90 min, >90 min en caso de Películas; y 1 Temporada, 2 Temporadas, >2 Temporadas en caso de Series). También hay un slider para seleccionar el número de elementos que se obtendrán.



Después se mostrará la gráfica con los resultados de la consulta usando **ej1_display()**. Usa por debajo **ej1_query()** para realizar la consulta: Se hace roll de usuarios porque no necesitamos diferenciar por usuarios, lo mismo para horas, y fecha. Después se hace un slice de contenido según el que se ha seleccionado en los widgets. Después la salida del almacén hay que limitarla y ordenarla para visualizarla en la gráfica. Para la grafica se ha usado una "barh" de **matplotlib**.

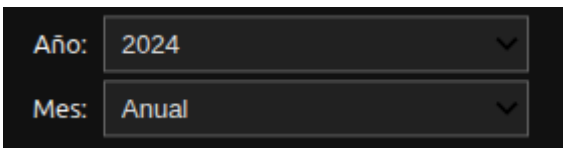


Ejercicio 5 (Libre)

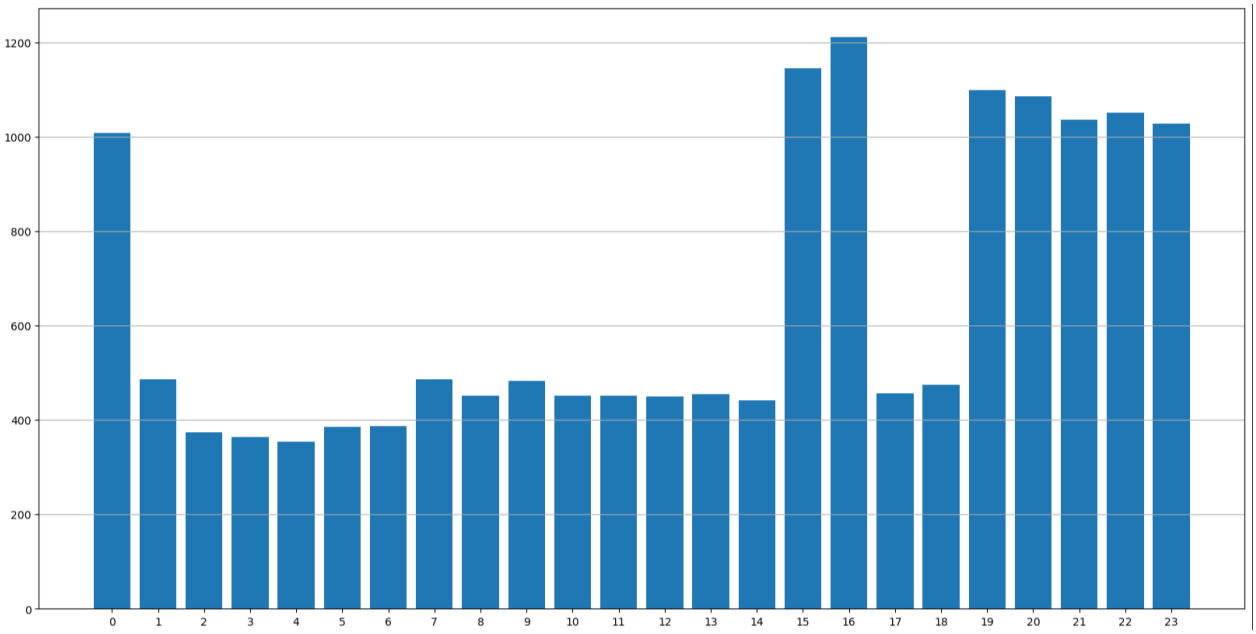
Se han realizado 3 ejercicios en este apartado

Audiencia por horas

Este ejercicio muestra una gráfica que compara el número de visualizaciones de cada hora del día. También tiene antes un desplegable para elegir si acotar a un mes concreto o de forma anual y otro para año (pero solo está disponible 2024 porque no se han realizado más fechas).



Para la operación de consulta se realiza roll para llegar a granularidad más gruesa en las dimensiones content y users. En la dimensión date se hace un roll (granularidad de mes). Según la selección, o bien se hace un slice del mes introducido, o bien se hace un roll (año) si se selecciona Anual. La dimensión hours se mantiene en granularidad atómica porque es lo que se quiere analizar. Para la grafica se ha usado bar de **matplotlib**.



Sistema de recomendación basado en usuarios

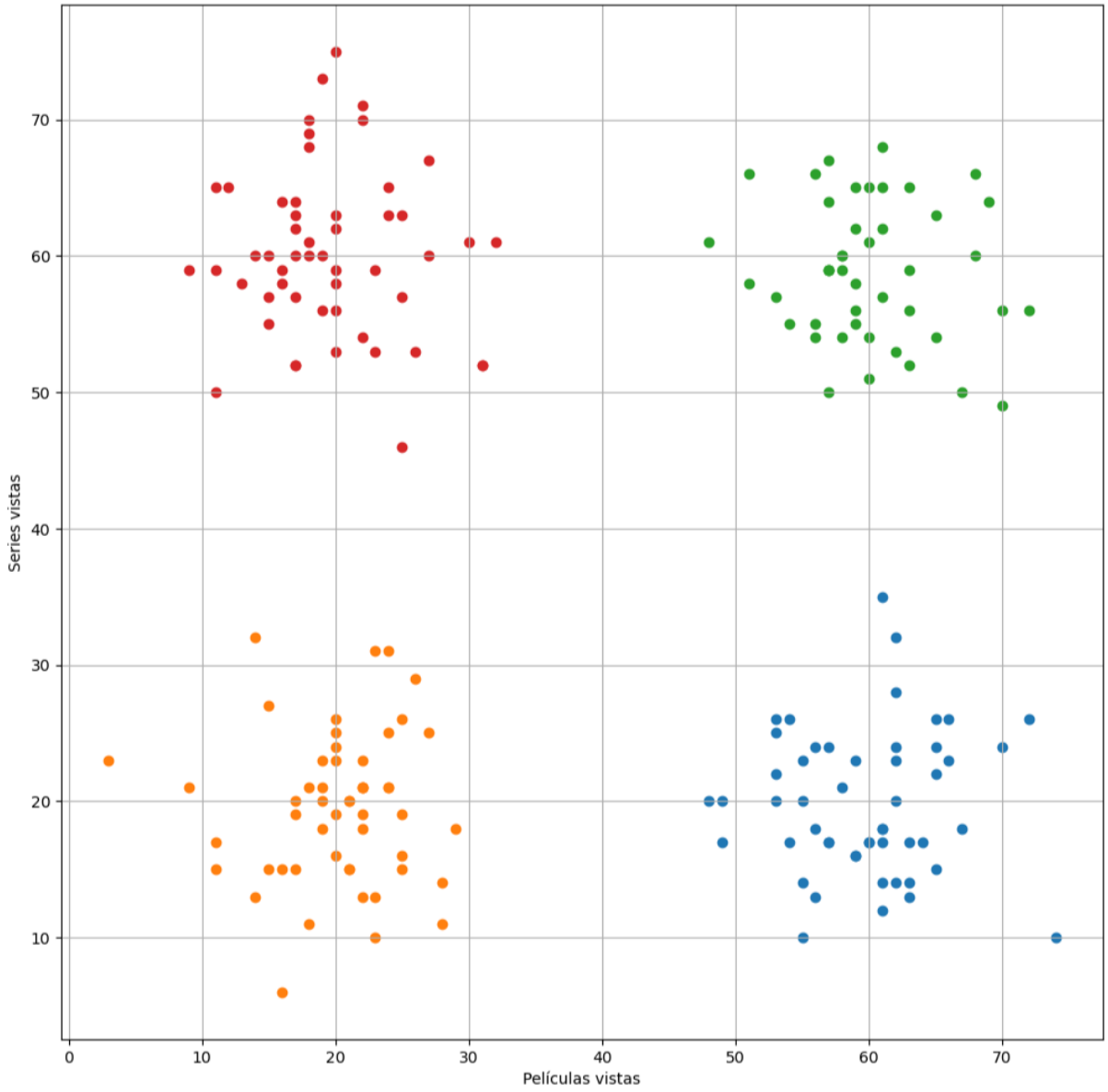
Se ha creado un sistema de recomendación basado en usuarios que usa la medida del coseno. Primero eliges a un usuario por su UID. Y se te muestra quien es el usuario más parecido y cuál es la mejor recomendación para el usuario seleccionado.


```
El usuario mas parecido de 0(Aaron) es 185(Victor). La recomendacion es s1286(Vincenzo)
```

En el algoritmo se han realizado consultas a la base de datos, pero no se han usado operaciones OLAP porque no tienen sentido en esta situación y usar operaciones MySQL es más sencillo.

Tipos de usuarios según visualizaciones

En este ejercicio se ha usado un algoritmo de clustering (K-means) para clasificar a los usuarios en grupos relacionados con el número de visualizaciones realizadas a películas frente a series.



Para este ejercicio el primer paso es generar un csv con 2 columnas: número de visualizaciones de películas y numero de visualizaciones de series. Cada entrada es un usuario. Para sacar este csv se ha usado el almacén de datos. Roll a granularidad más gruesa de date y hours. De la dimensión contenido se ha hecho roll hasta tipo (solo hay 2 entradas: Películas y Series). A partir de aquí se han cogido 2 vistas, una con un slice solo películas y otra con un slice solo series. Después mediante pandas se han mezclado para sacar el csv.

Ahora se llama a un comando de la terminal que usara **weka** con el algoritmo de K-means con el csv como entrada. La salida será un csv con una columna que identifica el cluster (a parte de la columna clave).

El csv de la salida se leerá y añadirla la columna del cluster al Dataframe del anterior csv. Por último, se dibuja la gráfica con matplotlib.