Data Warehouse

App de citas online

Adrián Edreira Gantes

Lucas García García

Sergio Liste Vázquez

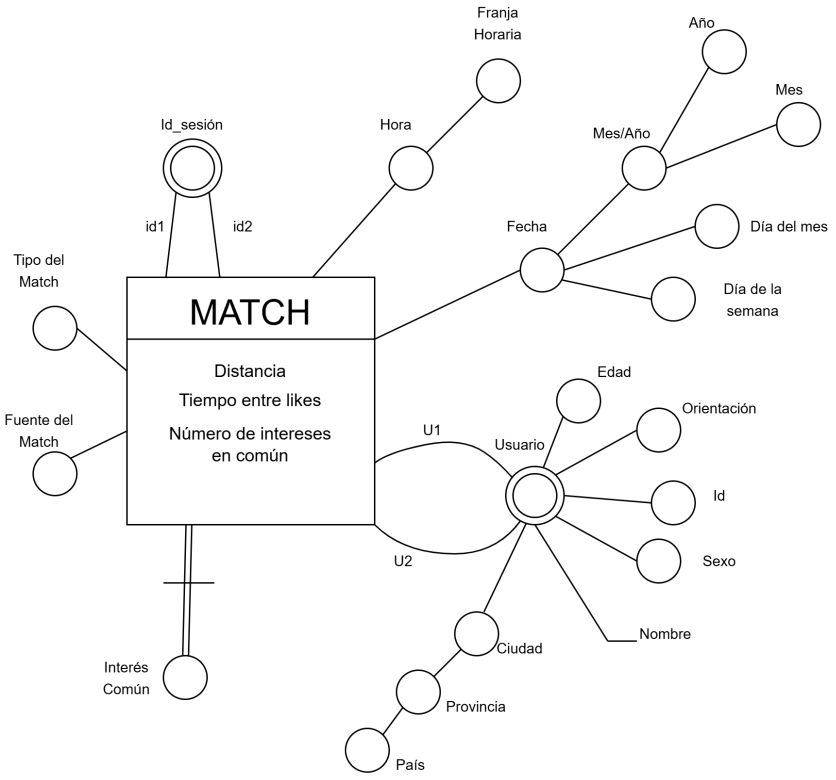
Proceso de negocio

El proceso de negocio a analizar serán los emparejamientos realizados entre dos personas dentro de nuestra app, que denominaremos match. Este es el proceso fundamental de cualquier app de citas, ya que permite a los usuarios conectarse entre sí y cuanto mejor sea el sistema de emparejamiento más usuarios decidirán utilizar nuestra app sobre las de la competencia. El objetivo del Data Warehouse será analizar qué factores influyen más para que se produzca un match entre dos usuarios diferentes, con el fin de mejorar la experiencia del usuario, optimizar el algoritmo de emparejamiento y tomar decisiones estratégicas basadas en datos.

Nivel de granularidad

La granularidad de nuestra tabla de hechos se establece a nivel de match, cada fila en dicha tabla representa la coincidencia entre dos usuarios e incluye información variada como la fecha en la que se da el hecho, el tipo de match, intereses en común… Por esto nuestra tabla de hechos es de grano fino, ya que permite hacer diferentes consultas gracias a que registramos toda la información necesaria para analizar un match concreto.

Diagrama DFM



Dimensiones

Entre las dimensiones que encontramos en el siguiente diagrama DFM, con un esquema en estrella, destacamos algunas como:

- Tipo de match (normal, supermatch, envío de un mensaje directo…)
- Fuente del match (swipe, recomendados, comunidades…)
- Id_sesión1 e Id_sesión2, dos dimensiones degeneradas que guardan la sesión de cada usuario en la que le gustó el perfil del otro usuario (Id_sesión1 para el Usuario1 que es el que da el primer like, Id_sesión2 para el Usuario2 que es el que completa el match).
- Interés común, una dimensión multivaluada y opcional.
- Usuario, una dimensión de jerarquía compartida.
- Hora, que registra la hora en la que se realiza el match, y con una franja horaria (mañana, mediodía, tarde, noche…)
- Fecha, que registra la fecha completa y permite extraer la información necesaria acerca de ella

Entre los atributos que cambian con el tiempo nos encontramos con la edad y la ubicación, pertenecientes a la dimensión usuario. La edad cambia cada año y la ubicación podría cambiar diariamente, por ejemplo si es una persona que viaja mucho. Lo más adecuado para gestionar estos cambios será usar una SCD2, ya que permite guardar la información anterior y la fecha de validez de cada fila.

Tipo de match y Fuente del match podrían formar parte de una dimensión basura llamada Información del match.

Algunos atributos o dimensiones descartadas fueron el tipo de usuario, el número de fotos en el perfil, la descripción del perfil o el tiempo que llevan usando nuestra app desde la primera vez.

Hechos medibles

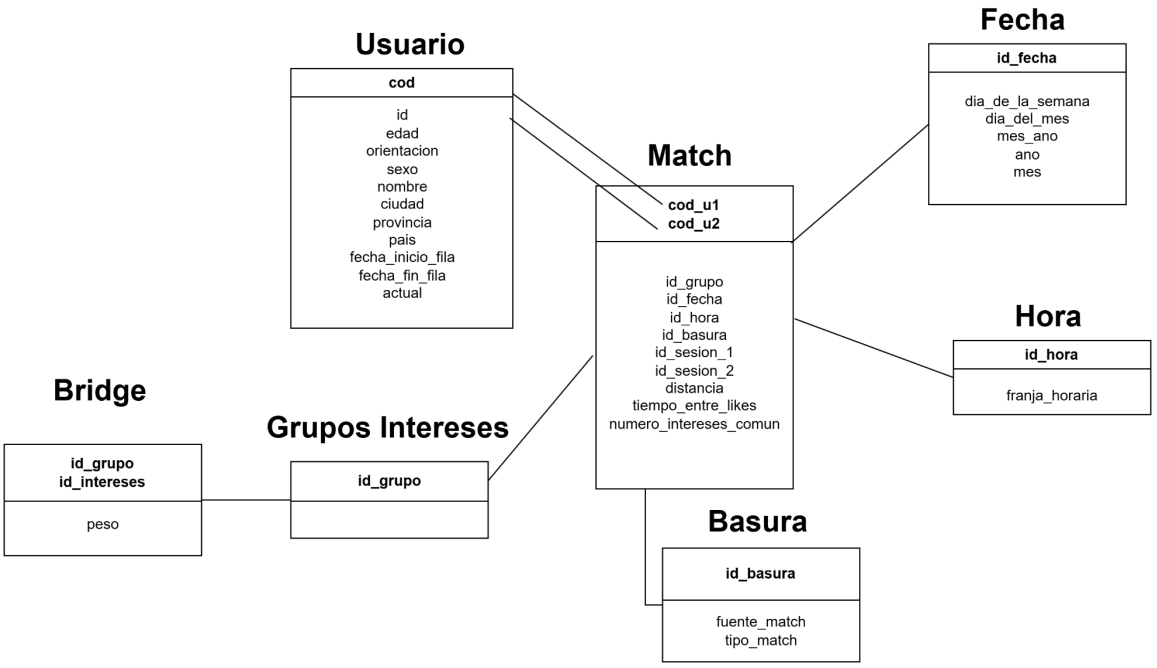
Los hechos medibles son la distancia entre los dos usuarios del match, la cantidad de intereses en común mutuos y el tiempo transcurrido desde que uno da el primer like hasta que se formaliza el match con el segundo like. Todos ellos son no aditivos, ya que se usarán para hacer medias principalmente. Algunos hechos descartados fueron la diferencia de edad de los usuarios o la diferencia de popularidad entre ambos perfiles (diferencias de cantidad de matches de cada usuario).

Ejemplos de explotación

Entre los ejemplos de consultas que podríamos realizar destacamos los siguientes:

- ¿Qué día de la semana es el más activo?
- ¿En qué rango de horas se producen más matches?
- ¿Existe alguna relación entre la orientación sexual y la cantidad de matches?
- ¿Cuánto tiempo de media tarda en producirse un match desde el primer like?
- ¿Cuál es la media de distancia entre dos personas en un match?
- ¿Cuántos matches se realizan por cada sesión?
- ¿Cuál es el tipo de match más común?
- ¿Mediante qué fuente se producen más matches?
- ¿Cuántos intereses de media hay en común en cada match?
- ¿Qué rango de edad usa más nuestra app?
- ¿Dónde se encuentra una mayor cantidad de usuarios?
- ¿Tiene relación la edad con la fuente del match?
- ¿El sexo influye en el tiempo entre likes?
- ¿La orientación sexual aumenta la distancia entre dos usuarios? (por ejemplo, ¿un pansexual necesitaría buscar más lejos?)
- ¿Qué ubicaciones reducen la distancia?
- ¿Un rango de edades tiene más matches a horas concretas?

Modelo lógico



Usamos SCD2 para la dimensión que varían con el tiempo, que es Usuario. Al utilizar SCD2 tenemos que crear las columnas cod, fecha_inicio_fila, fecha_fin_fila y actual para saber si esa fila es la actual. Esta última variable es un booleano, si la fila contiene la información actual tendrá un 1 y un 0 si esa fila está desactualizada. En el caso de que los datos estén actualizados la fecha_fin_fila estará por defecto a 31/12/9999. Por otra parte, la columna cod identifica cada fila de la tabla usuario e id identifica a un mismo usuario, de esta forma podemos saber qué información del usuario cambia.

La dimensión basura contiene las diferentes combinaciones de las dimensiones fuente del match y tipo de match. Para el grupo de intereses creamos una tabla bridge y una tabla intermedia para cumplir la integridad referencial. En la siguiente imagen vemos como existe una tupla con el id “-1” para mostrar todos los matches que no tengan intereses en común.

	ID_GRUPO	ID_INTERES	PESO
1	-1	sin_interes	1
2	7	cine	0,3333
3	7	comics	0,3333
4	7	peliculas indie	0,3333
5	8	cocina	0,5
6	8	reposteria	0,5
7	9	videojuegos	1

La tabla match ha sido creada como “matches” debido a que MATCH es una palabra reservada por Oracle. Además, se le ha asignado como clave primaria la combinación de “cod_u1” y “cod_u2” ya que solo existirá un único match entre dos usuarios, de esta forma se podrá identificar unívocamente a cada match.

Creación del DW

Para la creación e inserción de tablas solo debemos ejecutar los archivos .sql de la carpeta “Creacion-SQL”. Primero debemos ejecutar el archivo “create-tables”, en el caso de que estén creadas las tablas en nuestra base de datos, deberemos ejecutar con anterioridad el archivo “drop-tables”. Posteriormente, deberemos ejecutar los archivos comenzados por “insert”, dejando como últimos a “insert-bridges” y a “insert-matches” para cumplir la integridad referencial del data warehouse.

Por otra parte, aunque en los scripts de inserción de datos en las tablas contengan fechas, grupos de intereses o usuarios sin matches, el objetivo es mostrar a grandes rasgos cómo funciona el data warehouse. Se ha usado la ayuda de ChatGPT para la creación de inserciones de los datos y se revisó que fueran correctos de cara al ámbito del data warehouse.

Al tratarse de datos ficticios muchos datos no son usados, pero en un caso real solo estarían los datos que realmente hayan ocurrido y existan en nuestra tabla de hechos. Por ejemplo, si un conjunto de intereses no han hecho match anteriormente no debe estar en su correspondiente tabla. Esto afectará en los resultados de las consultas realizadas en el apartado de explotación, ya que los valores estarán distribuidos de forma mayoritariamente equitativa y no se extraerán resultados concluyentes, como si debería suceder en un caso real.

Explotación del DW

Las siguientes consultas se encuentran en sus respectivos archivos .sql de la carpeta “Consultas-SQL”:

Consulta 1 (PIVOT)

Esta consulta analiza cómo se distribuyen las distintas fuentes de generación de match en función de la edad de los usuarios implicados. Utilizando la cláusula PIVOT, se transforman las fuentes de match en columnas, mostrando cuántos matches de cada tipo se han generado por cada año de edad. Aunque se podría utilizar rangos de edad, consideramos que mediante otras herramientas podrían crearse dichos rangos y mantener en el data warehouse un grano más fino en relación a los datos.

Esto nos ayuda a ver que fuente de match es más usual en cada edad. El objetivo de esta consulta es mejorar la experiencia del usuario y/o promocionar otras formas de hacer match no tan utilizadas por ciertas edades.

```
SELECT *
FROM (
   SELECT u.edad, b.fuente_match
   FROM matches m
   JOIN usuario u ON m.cod_u1 = u.cod OR m.cod_u2 = u.cod
   JOIN basura b ON m.id_basura = b.id_basura
)
PIVOT (
   COUNT(fuente_match)
   FOR fuente_match IN (
         'recomendados' AS recomendados,
      'comunidades' AS comunidades,
      'sugerencias' AS sugerencias,
      'swipe' AS swipe,
      'eventos' AS eventos,
      'explorar' AS explorar)
)
ORDER BY edad;
```

	123 EDAD	123 RECOMENDADOS	123 COMUNIDADES	123 SUGERENCIAS	123 SWIPE	123 EVENTOS	123 EXPLORAR
1	19	10	1	6	11	8	6
2	20	3	8	8	9	5	2
3	21	6	12	12	12	9	10
4	22	9	5	4	7	7	6
5	23	5	4	13	7	9	6
6	24	10	4	7	5	4	6
7	25	16	7	5	5	6	8
8	26	7	5	5	6	7	6
9	27	5	8	2	7	4	7
10	28	13	4	9	10	6	8
11	29	4	5	4	8	2	6
12	30	6	5	7	8	11	7

Consulta 2 (CUBE)

Esta consulta genera el número total de matches agrupados por año, mes y día de la semana, lo que nos sirve para obtener los subtotales y totales acumulados de cada periodo. El objetivo es buscar si existen ciertos días de la semana, meses o periodos concretos en los que nuestra plataforma tiene más actividad.

Según los resultados que obtengamos podemos hacer campañas de publicidad en temporadas con pocos usuarios para obtener un mayor número de matches en dichos espacios de tiempo o mejorar la calidad de nuestras recomendaciones durante esa época para mejorar las estadísticas de matches.

También podremos aumentar la capacidad de nuestros servidores en temporadas con muchas conexiones a nuestra plataforma o enviar notificaciones a nuestros usuarios los días que se hacen mayor cantidad de matches para que inicien sesión o se suscriban a planes premium.

```
SELECT f.ANO, f.mes, f.DIA_DE_LA_SEMANA, count(*)
FROM MATCHES m JOIN FECHA f ON m.ID_FECHA = f.ID_FECHA
GROUP BY CUBE(f.ANO, (f.mes, f.MES_ANO), f.DIA_DE_LA_SEMANA)
ORDER BY f.ANO, f.mes_ano;
```

	123 ANO	A-Z MES	A-Z DIA_DE_LA_SEMANA	123 COUNT(*)
1	2.020	enero	lunes	6
2	2.020	enero	miércoles	4
3	2.020	enero	martes	3
4	2.020	enero	domingo	1
5	2.020	enero	sábado	8
6	2.020	enero	[NULL]	33
7	2.020	enero	viernes	4
8	2.020	enero	jueves	7

Consulta 3 (ORDER BY y RANK)

Esta consulta calcula la media de tiempo entre likes según la orientación sexual y luego los ordena con RANK para identificar qué orientación obtiene más rápido los matches. Esto nos ayuda a hacer un análisis del comportamiento de los usuarios, viendo qué orientaciones tienen mayor éxito en la aplicación.

Nos puede ayudar a tomar decisiones de negocio como publicidad orientada a ciertas orientaciones sexuales, pues si personas homosexuales promedian un tiempo menor entre likes podemos hacer una publicidad orientada a captar a esta parte del mercado. Por contraposición, si vemos que cierta orientación sexual tarda mucho en hacer match con otras personas corremos el riesgo de que abandonen nuestra plataforma, lo que nos permite tomar la decisión de mejorar el algoritmo para ese tipo de orientaciones y mantener la cantidad de usuarios.

```
SELECT
  U.ORIENTACION,
  AVG(M.TIEMPO_ENTRE_LIKES) AS PROMEDIO_TIEMPO_ORIENTACION,
  RANK() OVER (ORDER BY AVG(M.TIEMPO_ENTRE_LIKES) ASC) AS
RANK_ORIENTACION
FROM MATCHES M
JOIN USUARIO U ON M.COD_U1 = U.COD
GROUP BY U.ORIENTACION;
```

	A-Z ORIENTACION	123 PROMEDIO_TIEMPO_ORIENTACION	123 RANK_ORIENTACION
1	bisexual	243.013,5622317597	1
2	asexual	254.691,0092165899	2
3	homosexual	258.151,0637450199	3
4	heterosexual	261.173,1739130435	4