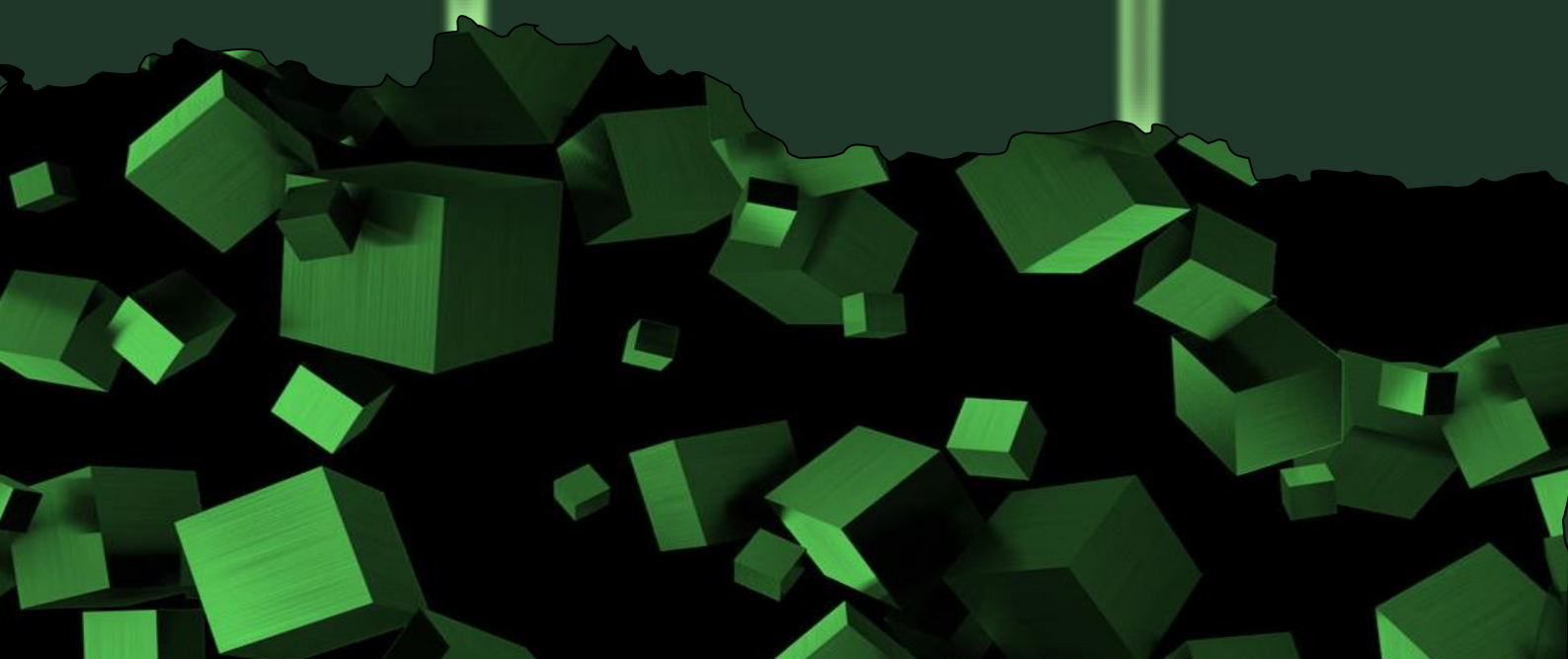
CUBE-X

PROYECTO BASE DE DATOS: CUBE_X

REALIZADO POR: ESTELA DE VEGA



ÍNDICE

UNIVERSO DEL DISCURSO

Un desarrollador ha decidido implementar un sistema de base de datos para gestionar la información sobre los tiempos de resolución de cubos de Rubik registrados por los usuarios en su aplicación.

Para los usuarios, se desea almacenar su identificación, su nombre (único), su correo electrónico, su contraseña, su rol (indicando si es socio o no), su nivel de experiencia (que aumenta con cada resolución) y la fecha de creación del usuario.

Los usuarios tienen la opción de convertirse en socios. Un socio puede beneficiarse de descuentos en torneos, este descuento aumenta con su antigüedad. En el caso de los socios, es fundamental registrar su identificación, la fecha de inicio de su membresía y el descuento asociado. Es importante destacar que un socio es un tipo de usuario. Asimismo, un usuario puede ser o no ser socio.

En relación con los tiempos de resolución de los cubos, se almacenará un identificador único y la fecha en que el usuario registró ese tiempo. Dado que la diferencia entre dos tiempos sucesivos siempre será de segundos, se garantiza que las fechas de registro nunca se repetirán, por tanto, un tiempo siempre será único. Además, se registrará el tiempo realizado por el usuario y los comentarios asociados, que pueden ser opcionales. Un usuario puede registrar varios tiempos, pero cada tiempo estará asociado a un único usuario.

Para registrar la acción de resolver un cubo, se utilizará un algoritmo único llamado scramble, que consiste en una combinación de entre 20 y 25 letras que nunca se repite. Se guardará el identificador junto con una descripción de la combinación. Dado que cada “scramble” es único, un tiempo solo puede estar asociado a un scramble y viceversa.

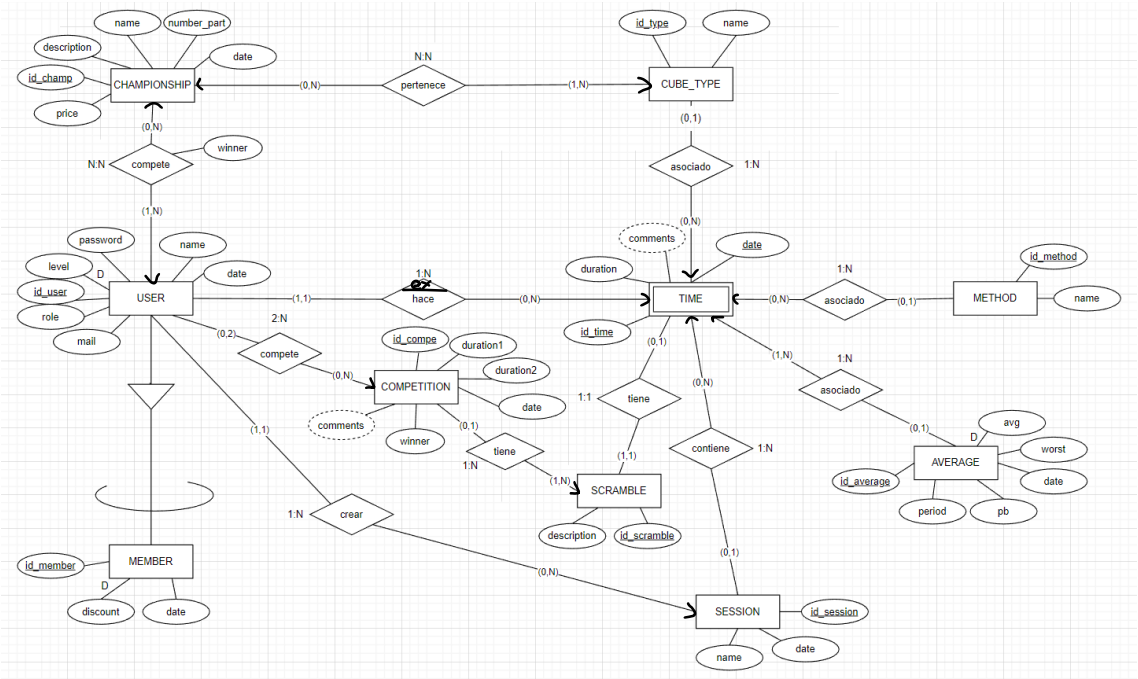
Adicionalmente, los usuarios pueden participar en competiciones de uno versus uno. Se busca registrar los tiempos que hicieron ambos usuarios, el ganador, los comentarios (opcionales), el scramble utilizado, asi como cuando se creó, identificada por un identificados único. Un usuario participar en varias competiciones, pero en cada competición solo puede haber dos usuarios.

Los usuarios también tienen la capacidad de crear sesiones para registrar tiempos. Cada sesión se identifica por un identificador único y un nombre, el cual no se podrá repetir. La fecha en que se crea la sesión también se registra. Cada sesión puede contener varios tiempos, pero un tiempo solo puede estar asociado a una sesión, ya que no se pueden repetir los tiempos.

Para el cálculo de la media, se conservarán datos como el identificados, el promedio (calculado con el periodo y los tiempos registrados), el periodo (que establece la cantidad de tiempos considerados para la media), el mejor y peor tiempo registrado, junto con la fecha de creación. Para calcular la media se necesitan varios tiempos, pero cada tiempo solo puede estar asociado a una media.

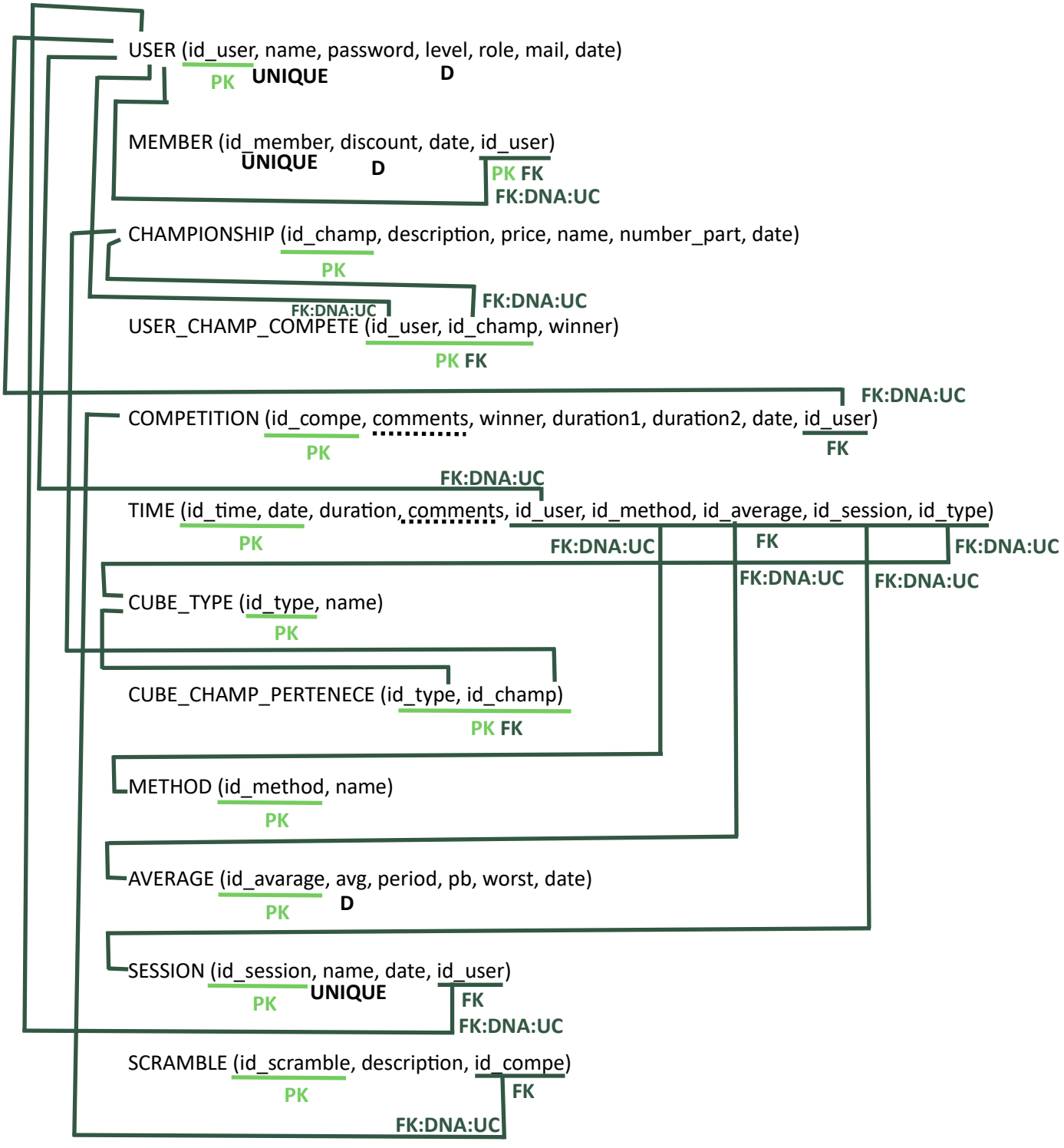
Dentro de los tiempos, se establecen relaciones con el método y el tipo de cubo utilizados. Para el método y el tipo de cubo, se busca almacenar el nombre e identificación. Además, con el tipo de cubo, se pueden llevar a cabo torneo, cuyos detalles incluyen su identificación, descripción, nombre, número participantes, precio y cuando se creó. En un torneo puede haber varios tipos de cubos.

MODELO ENTIDAD-RELACIÓN



MODELO RELACIONAL

A continuación, se muestra el modelo relacional del diagrama entidad-relación:



JUSTIFICACIÓN

En el modelo relacional propuesto, la generalización/especialización entre USER y MEMBER se ha resuelto siguiendo la técnica de estallar. Esto se traduce en la creación de dos tablas: la tabla del supertipo USER y, además, se ha creado la tabla para el subtipo MEMBER, donde se incluyen los atributos específicos del subtipo. La elección de esta técnica se justifica por la existencia de atributos únicos en MEMBER y la participación de USER en múltiples interrelaciones, asi como la posesión de varios atributos exclusivos.

Se han creado las relaciones USER_CHAMP_COMPETE y CUBE_CHAMP_ASSOCIATION, derivadas de las interrelaciones "compete" entre las entidades "CHAMPIONSHIP" y "USER" y, de la interrelación "pertenece" entre "CHAMPIONSHIP" y "CUBE_TYPE", respectivamente. El atributo "winner", que era originalmente un atributo propio de la interrelación "compete", se incorpora como atributo de la tabla USER_CHAMP_COMPETE.

Los atributos clave ajena se han configurado con la eliminación sin acción (ON DELETE NO ACTION). Por ejemplo, un usuario está asociado a varios torneos (CHAMPIONSHIP) a través de la relación USER_CHAMP_COMPETE. Si se intenta eliminar un usuario qeu participa en uno o más torneos, la configuración ON DELETE NO ACTION evitará que se elimine el usuario si existen registros relacionados en la tabla USER_CHAMP_COMPETE.

También se establece la configuración de restricción de actualización en cascada (ON UPDATE CASCADE). Esto simplifica la actualización de valores asociados cuando se modifican las claves primarias. Por ejemplo, si el id_champ de un torneo en la tabla CHAMPIONSHIP se actualiza, si hay registros en USER_CHAMP_COMPETE que referencian ese id_champ, la configuración ON UPDATE CASCADE actualizará automáticamente el valor correspondiente en USER_CHAMP_COMPETE, manteniendo la coherencia de los datos.

Los atributos derivados, como "level" en USER, "discount" en MEMBER y "avg" en TIME, se actualizan dinámicamente. Por ejemplo, "level" se incrementa según la experiencia del usuario, "discount" se ajusta según la antigüedad del miembro, y "avg" se calcula mediante una fórmula específica aplicada a un periodo de tiempo. Esta estrategia evita redundancias y garantiza la coherencia de los datos.

Las claves primarias se han definido como dominios de números enteros y de tipo autonumérico, de manera que su valor se insertaría automáticamente al agregar un registro en una tabla, para agilizar las búsquedas y para que las claves primarias sean únicas.

Con esta solución se considera que el diseño lógico está suficientemente documentado para llevar a cabo el diseño físico de manera correcta.

ESQUEMA NORMALIZADO

El modelo relacional propuesto sigue los principios de normalización hasta la Tercera Forma Normal (3FN), lo que garantiza que las redundancias sean mínimas.

En primer lugar, todas las tablas cumplen con la Primera Forma Normal (1FN) al asegurar que todos los atributos son atómicos, es decir, cada atributo en las tablas contiene un solo valor, como por ejemplo “USER”,” MEMBER”,” CHAMPIONSHIP”, etc. Además, no hay valores duplicados en las columnas. Por ejemplo, en la tabla “USER”, el atributo “name” contiene un solo valor y no se descompone en subvalores, evitando conjuntos de valores en un solo atributo.

La Segunda Forma Normal (2FN) se cumple ya que se cumple la Primera Forma Normal (1FN) y se logra eliminar las dependencias parciales, asegurando que cada atributo no clave dependa completamente de la clave primaria. Tomando la tabla “USER_CHAMP_COMPETE” como ejemplo. La clave primeria esta compuesta por dos atributos: “id_user” e “id_champ”. En esta tabla, todos los demas atributos (“winner”) dependen completamente de la combinación de “id_user” e “id_champ”. No hay dependencias parciales, cumpliendo asi con la 2FN.

Se evitan dependencias transitivas y se cumple la Segunda Forma Normal (2FN), cumpliendo asi con la Tercera Forma Normal (3FN), y cada atributo no clave depende solo de la clave primaria de su respectiva tabla. Tomando la tabla “USER” como ejemplo. Suponiendo que “level” es derivado solo de “id_user” y “discount” es derivado solo de “id_member” no habría dependencias transitivas, ya que level y discount dependen directamente de las claves primarias de “id_user” e “id_member”, respectivamente, y no de otros atributos en esas tablas.

Además, se observa la normalización de atributos derivados, como “level” en “USER”, “discount” en “MEMBER” y “avg” en la tabla “TIME”.

-El ***nivel de usuario*** (“level”) se calcula en función de su experiencia, que aumenta cada vez que realiza tiempos. Este enfoque evita almacenar directamente el nivel, manteniendo la coherencia de los datos.

-El ***descuento de socio*** (“discount”) se determina en base a la antigüedad del miembro y va aumentando con el tiempo. Al calcular el descuento en lugar de almacenarlo directamente, se garantiza que el descuento siempre refleje la antigüedad actual del miembro.

-El ***promedio*** (“avg”) se calcula aplicando una formula especifica sobre el periodo de tiempo. Se elige un periodo, que será de cuantos tiempos se desee hacer la media, y se divide entre tres, quitando el mejor y peor tiempo (máximo y minimo tiempo). Al realizar este cálculo en lugar de almacenar el promedio directamente se asegure que el promedio siempre se calcule de acuerdo con las reglas establecidas, como excluir el mejor y peor tiempo.

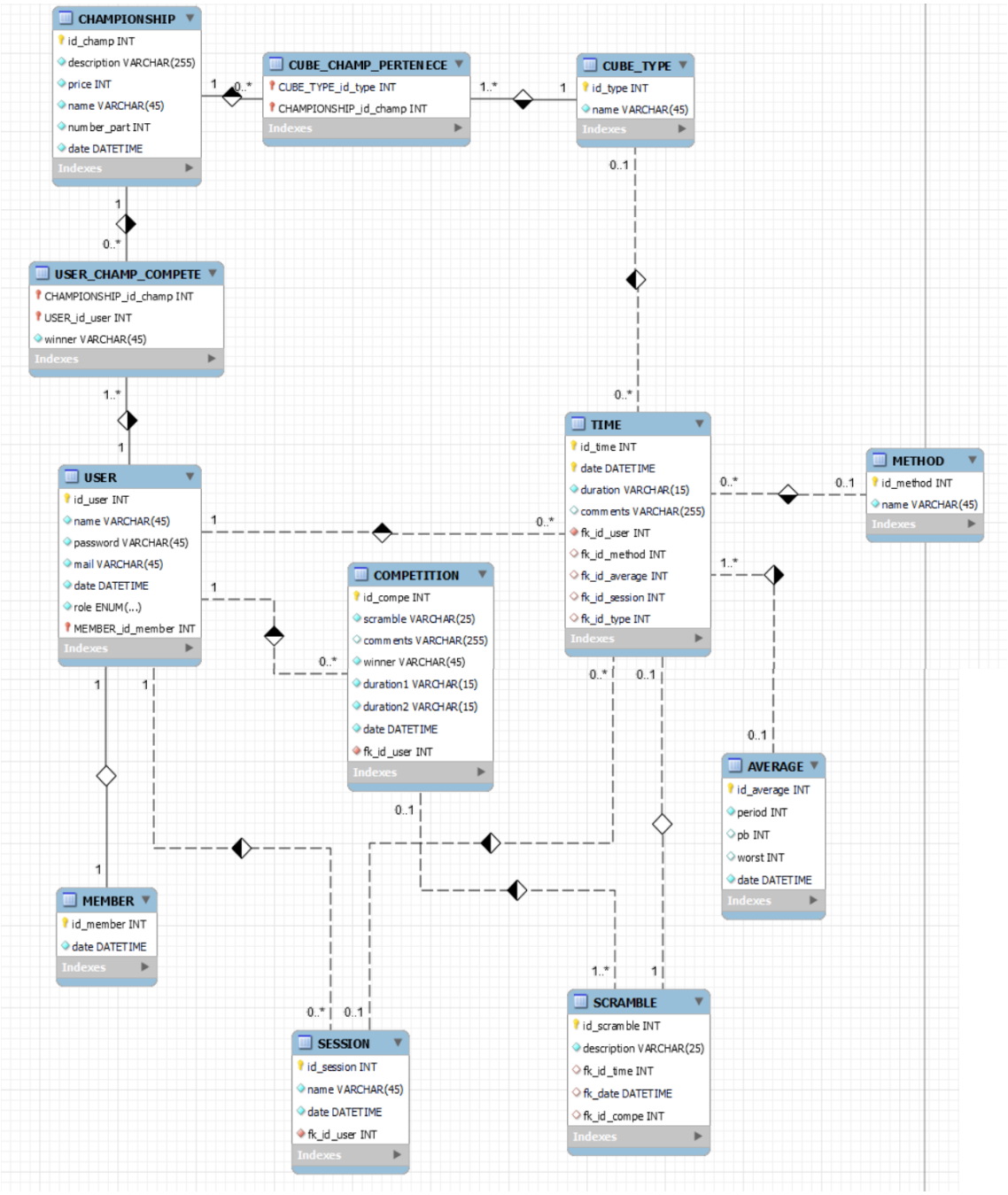
En todos estos casos, la normalización se logra al evitar redundancias. Los cálculos se realizan dinámicamente cuando es necesario obtener estos valores, garantizando que siempre estén actualizados. Esto mejora la eficiencia y la integridad de la base de datos.

Los atributos opcionales, como “comments” en la tabla “COMPETITION” y “comments” en la tabla “TIME”, se manejan adecuadamente permitiendo valores nulos cuando sea necesario, esto ayuda a la adaptabilidad de la base de datos. La unicidad de datos se garantiza mediante el uso de restricciones de unicidad (UNIQUE) en atributos como “name” en las tablas “USER” y “SESSION”. Estas restricciones ayudan a mantener la integridad de los datos. No se identifican atributos compuestos y cada atributo en una tabla representa una única propiedad.

En conclusión, el modelo relacional propuesto esta normalizado hasta la Tercera Forma Normal (3FN).

BASE DE DATOS

MODELO ENTIDAD/RELACIÓN WORKBENCH



CÓDIGO BASE DE DATOS

modelo_er.txt (Línea de comandos)

CONSULTAS ATRIBUTOS DERIVADOS

1. Consulta para calcular el descuento que posee un socio despues de un año. Por cada año se le hara 5€ de descuento.

```
78      -- MODIFICAR SOCIO TENGA UN AÑO DE MEMBRESIA
79      UPDATE MEMBER
80      SET date = DATE_SUB(NOW(), INTERVAL 1 YEAR)
81      WHERE id_member = 1;
82
83      -- CALCULAR EL DESCUENTO BASSADO EN LA ANTIGUEDAD
84      SELECT id_member, date AS inicio,
85          DATEDIFF(NOW(), date) AS dias_socio,
86          CONCAT((DATEDIFF(NOW(), date) DIV 365) * 5, '€')  AS descuento
87      FROM MEMBER
88      WHERE id_member = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_member	inicio	dias_socio	descuento
1	2022-12-14 22:17:05	365	5€

2. Consulta para calcular el nivel que tiene un usuario que ha registrado 45 tiempos. Por cada tiempo registrado se le suma 25 puntos de experiencia. Cuando la experiencia llegue a 100 se le suma un nivel.

```
93      -- CALCULAR EL NIVEL DE UN USUARIO DEPENDIENDO DE LOS TIEMPOS REGISTRADO
94      SELECT id_user,
95          COUNT(id_time) * 25 AS experiencia,
96          FLOOR(COUNT(id_time) * 25 / 100)  AS nivel
97      FROM USER INNER JOIN TIME ON id_user = fk_id_user
98      WHERE id_user = 1
99      GROUP BY id_user;
100
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_user	experiencia	nivel
1	1125	11

3. Consulta para calcular la media de 5 tiempos. Se elimina el máximo y minimo y se hace divide entre 3

```
-- TIME
INSERT INTO TIME (id_time, date, duration, fk_id_user, fk_id_method, fk_id_average, fk_id_session, fk_id_type)
VALUES
  (1, NOW(), '13', 1, 1, 1, 1, 1),
  (2, NOW(), '12', 1, 1, 1, 1, 1),
  (3, NOW(), '11', 1, 1, 1, 1, 1),
  (4, NOW(), '14', 1, 1, 1, 1, 1),
  (5, NOW(), '13', 1, 1, 1, 1, 1);
```

```
63 •  SELECT fk_id_user,
64          (SUM(duration) - MIN(duration) - MAX(duration)) / 3 AS average_duration
65    FROM (
66        -- SUBCONSULTA OBTENER 5 ULTIMOS TIEMPOS
67        SELECT fk_id_user, duration
68        FROM TIME
69        WHERE fk_id_user = 1
70        ORDER BY date DESC
71        LIMIT 5
72    ) AS recent_times
73    WHERE duration IS NOT NULL
74    GROUP BY fk_id_user;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fk_id_user	average_duration
1	12.666666666666666

Valores que se han utilizado para realizar las consultas:

consultas.sql

consultas.txt (Línea de comandos)