DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo práctico - Primera Parte

1er Cuatrimestre 2016 — Bases de Datos

Integrante	LU	Correo electrónico
Almansi, Emilio	674/12	ealmansi@gmail.com
Fixman, Martín	391/11	martinfixman@gmail.com
Gunski, María Celeste	899/03	celestegunski@gmail.com
Maurizio, Miguel	635/11	miguelmaurizio_92@hotmail.com

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

http://www.fcen.uba.ar

Índice

1. Introducción

El presente trabajo describe en detalle el diseño y la implementación de un sistema de persistencia de datos presentado sobre un problema representativo -aunque simplificado- del mundo real.

La metodología de diseño utilizada fue la Metodología de Diseño Lógico para Bases de Datos Relacionales (LRDM - Logical Relational Design Methodology), la cual tiene como primer paso la desambiguación de los requerimientos planteados en el problema y la construcción consecuente del Modelo Entidad Relación para el escenario en cuestión. La forma de representación utilizada para dicho modelo conceptual fue el Diagrama Entidad Relación.

En una etapa posterior, pasamos al diseño lógico de la solución transformando el conocimiento comprendido en el DER al modelo relacional, al partir del cual se procede a la implementación de la solución en un motor de base de datos relacional. En este caso, el motor elegido para la implementación concreta fue SQLite[1].

1.1. Descripción del Problema

El problema para el cual se desarrolló un sistema de persistencia de datos consiste en un Mercado Virtual de intención similar a las comunidades de compra y venta online como MercadoLibre u OLX.

El sistema permite que sus usuarios publiquen artículos o servicios para ser comprados o contratados por otros usuarios. Los usuarios del sistema pueden ser particulares o empresas, teniendo iguales atribuciones y deberes a la hora de publicar, comprar o vender productos en el mercado. Los usuarios pueden ofrecer productos o servicios (o combinaciones de ambos) a un precio fijo o someter sus publicaciones a una subasta donde se establece un precio inicial y durante un tiempo determinado los demás usuarios pueden realizar ofertas con valores crecientes.

El mercado tiene una comisión por cada publicación realizada que deviene en una compra, pudiendo ser las publicaciones de distintos tipos y con diferentes valores de comisión o extensión del período de vigencia de las mismas. Todas las publicaciones permiten adicionalmente que los usuarios que las visitan realicen preguntas al vendedor, quien puede responderlas.

Las compras y ventas realizadas en el sitio siempre llevan una calificación obligatoria por parte de tanto el vendedor como el comprador, con una valoración numérica de 1 (no recomendable) a 10 (muy recomendable) así como un comentario opcional para el otro usuario involucrado en la transacción.

El sistema almacena adicionalmente el historial de compras de los usuarios, sus visitas a publicaciones, y aquellos productos o servicios que los usuarios hayan marcado como favoritos.

Los detalles más específicos del problema se encuentran en el enunciado del trabajo práctico[2].

[1] https://www.sqlite.org/

[2] http://www.dc.uba.ar/materias/bd/2016/c1/descargas/TP1

1.2. Funcionalidades a implementar

Como requerimientos adicionales a la persistencia, hay una serie de funcionalidades particulares que implementamos de forma tal de verificar el correcto funcionamiento de la base. Estas son:

- Consulta por usuario: obtener, para un usuario específico, información sobre los artículos que ha comprado y vendido, los artículos que ha visitado con su fecha de visita, los artículos que tiene en su lista de favoritos, y las primeras 3 categorías de artículos que visitó con mayor frecuencia en el último año.
- Consulta por categoría de producto: obtener, dada una categoría de producto ("Computación", "Hogar, muebles y jardín", etc), un listado de los vendedores que han publicado artículos de dicha categoría y la cantidad de ventas que efectuó cada uno de dichos vendedores.
- Función "Ofertar": debe permitir al usuario ofertar una suma en una subasta. Dicha suma debe ser superior en al menos 1 peso, a la oferta actual, e inferior al doble de la oferta actual.
- Consulta por usuario y preguntas: obtener para un usuario específico, la lista de preguntas que ha realizado, con las respectivas respuestas que haya recibido (sólo la pregunta, si aún no recibió respuesta).
- Consulta por keyword: obtener, para un cierto keyword (por ejemplo "mesa"), la lista de publicaciones vigentes que tengan en el título, dicha keyword. El usuario debe poder restringir su búsqueda sólo a cierta categoría de artículos o servicios.
- Consulta por ganador/es anual de viaje a Khan El-Khalili: obtener, para un año específico, el ganador/es

2. Modelo de Entidad Relación

Como dijimos en la sección 1, para representar el Modelo de Entidad Relación realizado utilizamos el Diagrama Entidad Relación (DER). Presentado en las subsiguientes figuras, el DER fue dividido en diferentes secciones lógicas para facilitar su navegación, pero conceptualmente se trata de un único diagrama. Las divisiones constan de una sección central que incluye las interrelaciones más relevantes entre las entidades de Usuario, Publicación y Compra; y luego, una sección por cada una de estas tres entidades mostrando en detalle su composición y entidades relacionadas.

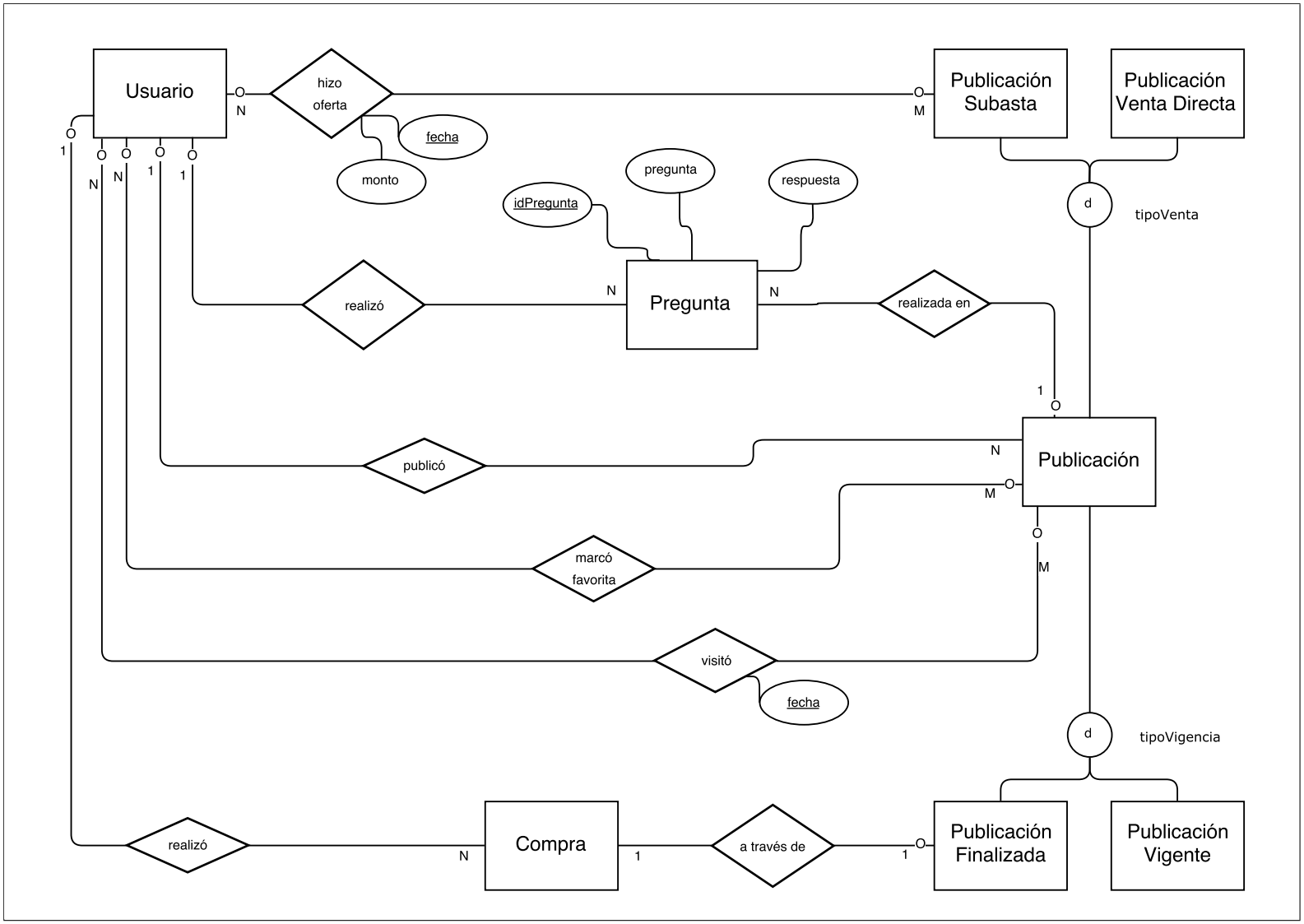

Figura 1: Diagrama Entidad Relación: Sección Central.

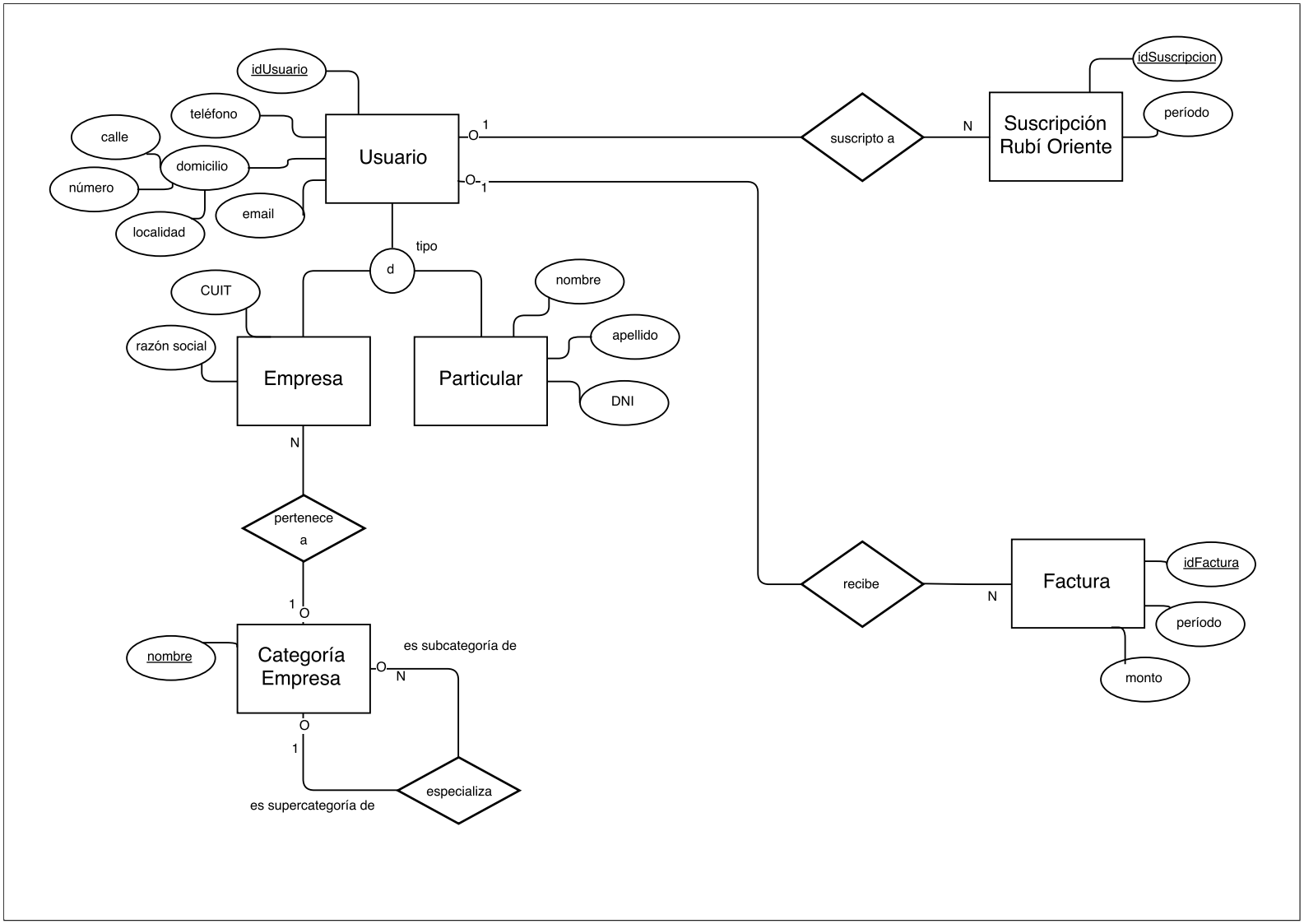

Figura 2: Diagrama Entidad Relación: Sección Usuario.

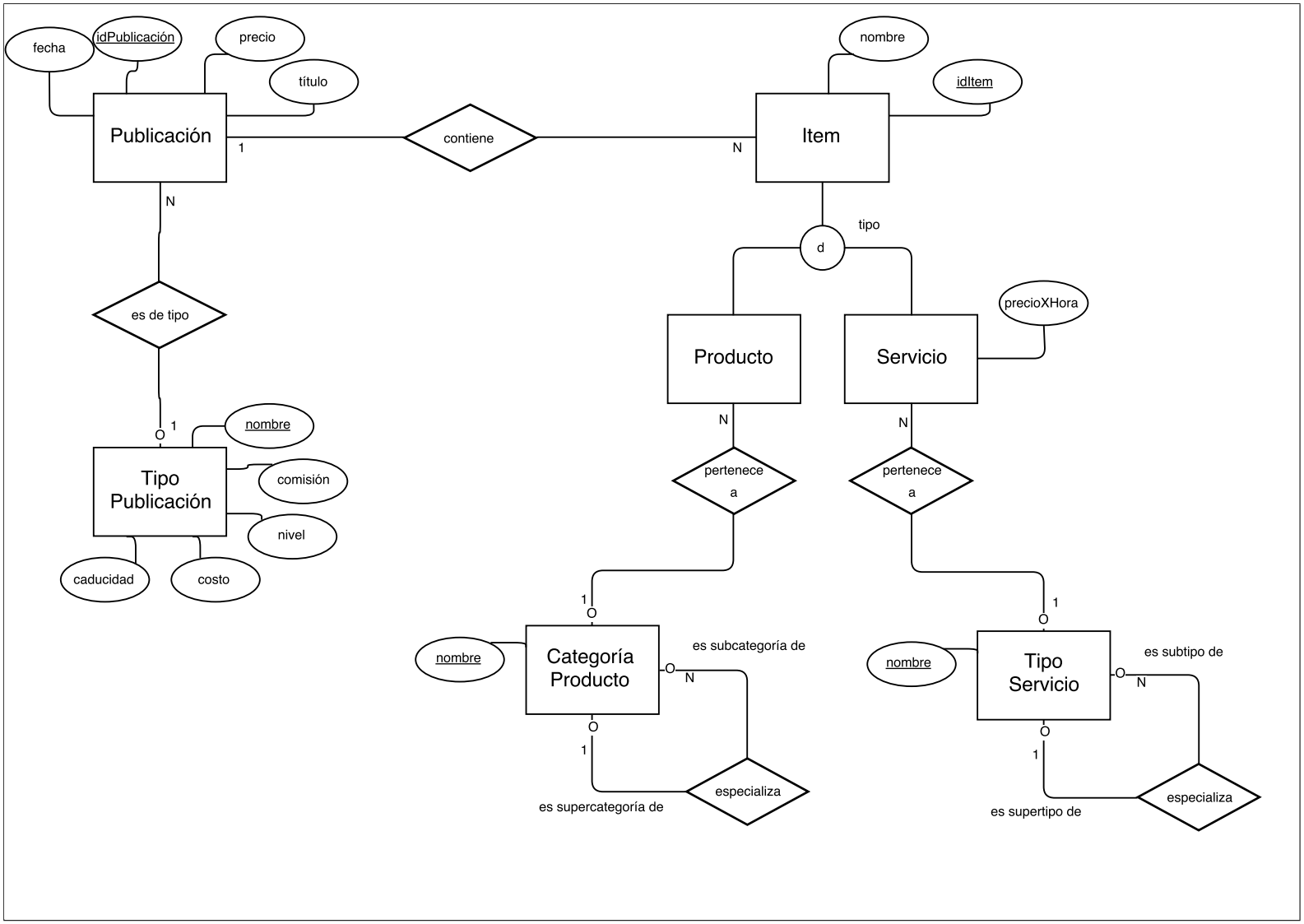

Figura 3: Diagrama Entidad Relación: Sección Publicación.

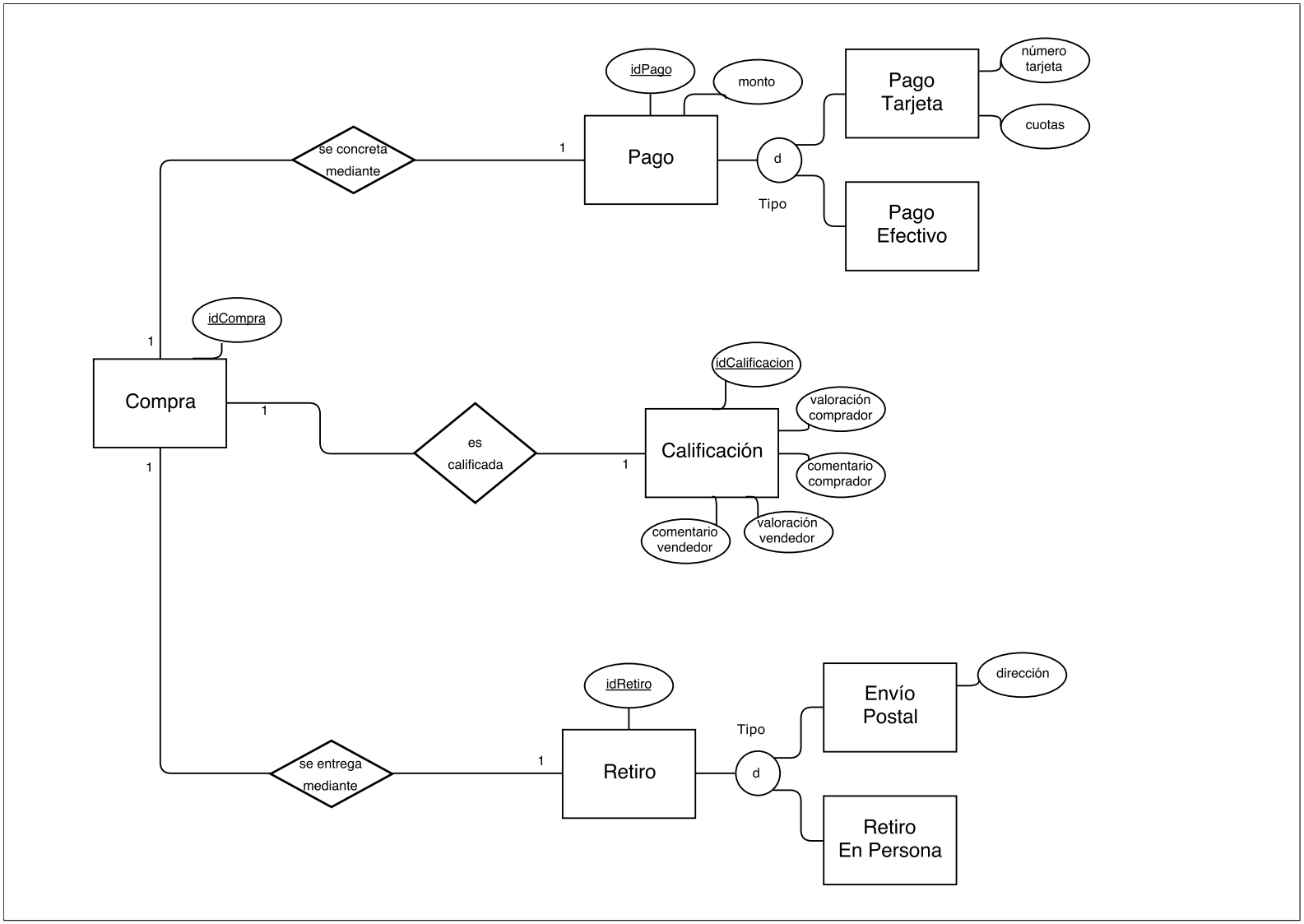

Figura 4: Diagrama Entidad Relación: Sección Compra.

3. Decisiones de diseño

- Preguntas a una publicación: consideramos que las preguntas realizadas en una publicación pueden tener una única respuesta, siguiendo el modelo de Mercado-Libre. Si el usuario desea replicar la respuesta, debe hacerlo mediante una nueva pregunta.

4. Modelo Relacional

A continuación incluimos los esquemas de relación que se derivan del MER de la sección 2.

Usuario (idUsuario, calle, numero, localidad, telefono, email, tipo)
PK=CK={idUsuario}
FK={}

Particular (idUsuario, DNI, nombre, apellido)
PK={idUsuario}
CK={idUsuario, DNI}
FK={idUsuario}
Particular.idUsuario debe estar en Usuario.idUsuario

Empresa (idUsuario, CUIT, razonSocial, nombreCategoriaEmp)
PK={idUsuario}
CK={idUsuario, CUIT}
FK={idUsuario, nombreCategoriaEmp}
Empresa.idUsuario debe estar en Usuario.idUsuario
Empresa.nombreCategoriaEmp debe estar en CategoriaEmpresa.nombre

CategoriaEmpresa (nombre, nombreSuperCategoria)
PK=CK={nombre}
FK={nombreSuperCategoria}
CategoriaEmpresa.nombreSuperCategoria puede ser nulo o debe estar en CategoriaEmpresa.nombre
CategoriaEmpresa.nombre puede no estar en CategoriaEmpresa.nombreSuperCategoria

SuscripcionRubiOriente (idSuscripcion, periodo, idUsuario)
PK=CK={idSuscripcion}
FK={idUsuario}
SuscripcionRubiOriente.idUsuario debe estar en Usuario.idUsuario
Usuario.idUsuario puede no estar en SuscripcionRubiOriente.idUsuario

Factura (idFactura, periodo, monto, idUsuario)
PK={idFactura}
CK={idFactura, (idUsuario, periodo)}

FK={idUsuario}
Factura.idUsuario debe estar en Usuario.idUsuario
Usuario.idUsuario puede no estar en Factura.idUsuario

Publicacion (idPublicacion, titulo, fecha, precio, tipoPublicacion, tipoVigencia, tipoVenta, idUsuarioPublicador)
PK=CK={idPublicacion}
FK={tipoPublicacion, idUsuarioPublicador}
Publicacion.tipoPublicacion debe estar en TipoPublicacion.nombre
TipoPublicacion.nombre puede no estar en Publicacion.tipoPublicacion

PublicacionSubasta (idPublicacion)
PK=CK={idPublicacion}
FK={tipoPublicacion}
PublicacionSubasta.idPublicacion debe estar en Publicacion.idPublicacion
Publicacion.idPublicacion puede no estar en PublicacionSubasta.idPublicacion

PublicacionFinalizada (idPublicacion)
PK=CK={idPublicacion}
FK={tipoPublicacion}
PublicacionFinalizada.idPublicacion debe estar en Publicacion.idPublicacion
Publicacion.idPublicacion puede no estar en PublicacionFinalizada.idPublicacion

TipoPublicacion (nombre, comision, costo, nivel, caducidad)
PK=CK={nombre}
FK={}

Item (idItem, idPublicacion, nombre, tipo)
PK=CK={idItem}
FK={idPublicacion}

Producto (idItem, nombreCategoriaProd)
PK=CK={idItem}
FK={idItem, nombreCategoriaProd}
Producto.idItem debe estar en Item.idItem
Producto.nombreCategoriaProd debe estar en CategoriaProducto.nombre
CategoriaProducto.nombre puede no estar en Producto.nombreCategoriaProd

Servicio (idItem, precioXHora, nombreTipoServicio)
PK=CK={idItem}
FK={idItem, nombreTipoServicio}
Servicio.idItem debe estar en Item.idItem
Servicio.nombreTipoServicio debe estar en TipoServicio.nombre
TipoServicio.nombre puede no estar en Servicio.nombreTipoServicio

CategoriaProducto (nombre, nombreSuperCategoria)
PK=CK={nombre}
FK={nombreSuperCategoria}
CategoriaProducto.nombreSuperCategoria puede ser nulo o debe estar en CategoriaProducto.nombre
CategoriaProducto.nombre puede no estar en CategoriaProducto.nombreSuperCategoria

TipoServicio (nombre, nombreSuperTipo)
PK=CK={nombre}
FK={nombreSuperTipo}
TipoServicio.nombreSuperTipo puede ser nulo o debe estar en TipoServicio.nombre
TipoServicio.nombre puede no estar en TipoServicio.nombreSuperTipo

Compra (idCompra, idUsuario, idPublicacion, idPago, idCalificacion, idRetiro)
PK=CK={idCompra}
FK={idUsuario, idPublicacion, idPago, idCalificacion, idRetiro}
Compra.idUsuario debe estar en Usuario.idUsuario
Compra.idPublicacion debe estar en Publicacion.idPublicacion Compra.idPago debe estar en Pago.idPago
Compra.idCalificacion debe estar en Calificacion.idCalificacion
Compra.idRetiro debe estar en Retiro.idRetiro

Pago (idPago, monto, tipo)
PK=CK={idPago}
FK={}

PagoTarjeta (idPago, numeroTarjeta, cuotas)
PK=CK={idPago}
FK={idPago}
PagoTarjeta.idPago debe estar en Pago.idPago

Calificacion (idCalificacion, valoracionComprador, valoracionVendedor, comentarioComprador, comentarioVendedor)
PK=CK={idCalificacion}
FK={}

Retiro (idRetiro, tipo)
PK=CK={idRetiro}
FK={}

EnvioPostal (idRetiro, direccion)
PK=CK={idRetiro}
FK={idRetiro}
EnvioPostal.idRetiro debe estar en Retiro.idRetiro

HizoOferta (idUsuario, idPublicacion, fecha, monto)
PK=CK={(idUsuario, idPublicacion, fecha)}
FK={idUsuario, idPublicacion}
HizoOferta.idUsuario debe estar en Usuario.idUsuario
HizoOferta.idPublicacion debe estar en Publicacion.idPublicacion

Pregunta (idPregunta, idUsuario, idPublicacion, pregunta, respuesta)
PK=CK={idPregunta}
FK={idUsuario, idPublicacion}
Pregunta.idUsuario debe estar en Usuario.idUsuario
Pregunta.idPublicacion debe estar en Publicacion.idPublicacion
Usuario.idUsuario puede no estar en Pregunta.idUsuario
Publicacion.idPublicacion puede no estar en Pregunta.idPublicacion

MarcoFavorita (idUsuario, idPublicacion)
PK=CK={(idUsuario, idPublicacion)}
FK={idUsuario, idPublicacion}
MarcoFavorita.idUsuario debe estar en Usuario.idUsuario
MarcoFavorita.idPublicacion debe estar en Publicacion.idPublicacion

Visito (idUsuario, idPublicacion, fecha)
PK=CK={(idUsuario, idPublicacion, fecha)}
FK={idUsuario, idPublicacion}
Visito.idUsuario debe estar en Usuario.idUsuario
Visito.idPublicacion debe estar en Publicacion.idPublicacion

5. Restricciones

El problema presenta diferentes restricciones que no se encuentran modeladas en las representaciones previas (MER, MR), así como no se imponen de forma automática por el motor de la base de datos. Por lo tanto, es responsabilidad de quien ingresa los datos asegurar que se cumplan. Exampresamos en lenguaje natural dichas restricciones:

- En la entidad Calificación, las valoraciones toman valores en el rango de 1 a 10.
- Tanto para Categoria Producto, Tipo Servicio y Categoria Empresa, las subcategorias de las mismas no deben tener ciclos.
- Dentro de Tipo de Publicacion estan tanto RubíDeOriente, Oro, Plata, Bronce y Libre!. Además, estas cumplen las restricciones del enunciado[3] respecto a porcentajes y prioridad en las busquedas.
- Los pagos y los precios siempre son numeros positivos. Esto es, tanto monto en la entidad hizoOferta, precio en publicacion y monto en pago no son negativos

6. Implementación SQL

6.1. Motor elegido

El motor elegido para implementar el sistema de persistencia del mercado virtual fue SQLite[4]. Elegimos el mismo dada las facilidades que presenta a la hora del desarrollo de la base de datos, pudiendo almacenar la misma íntegramente en un archivo standalone que se puede compartir fácilmente. Adicionalmente, no es necesario configurar un entorno SQL para consultar la base, sino que se puede hacer sencillamente desde un programa de línea de comandos de fácil instalación.

6.2. Diseño de tablas

El diseño lógico de las tablas se describe a modo de sentencias SQL de tipo CREATE TABLE, las cuales se encuentran al final de este documento en el apéndice A.

6.3. Aclaraciones

A la hora de implementar la herencia disjunta en las entidades debimos colocar un identificador para saber de que tipo es.
Dicho identificador es un INTEGER, a continuación explicamos que implica cada número.

[3] http://www.dc.uba.ar/materias/bd/2016/c1/descargas/TP1
[4] https://www.sqlite.org/

Publicacion En tipoVigencia, un 0 implica vigente y un 1 finalizada. Para el tipo de publicación, 0 implica onrmal, 1 subasta.

Item En la entidad Item, 0 implica que es un prooducto y 1 que es un servicio.

Pago Aquí un 0 implica pago al contado y 1 pago con tarjeta.

Usuario Un 0 en esta entidad significa particular y un 1 que es una empresa.

Retiro En cuanto a Retiros, un 0 implica retiro personal, en cambio, un 1 implica envío postal.

6.4. Funcionalidades implementadas

Las funcionalidades adicionales descriptas en la sección 1 se implementaron como consultas en SQL, cuyo código puede encontrarse al final de este documento en el apéndice B.

7. Conclusiones

A. Creación de tablas

```
.bail ON
.echo ON
.headers ON
.mode column

PRAGMA foreign_keys = ON;

CREATE TABLE usuario (
idUsuario INTEGER NOT NULL PRIMARY KEY,
calle TEXT NOT NULL,
numero INTEGER NOT NULL,
localidad TEXT NOT NULL,
telefono TEXT NOT NULL,
email TEXT NOT NULL,
tipo INTEGER NOT NULL
);

CREATE TABLE particular (
idUsuario INTEGER NOT NULL PRIMARY KEY,
DNI INTEGER NOT NULL,
nombre TEXT NOT NULL,
apellido TEXT NOT NULL,
FOREIGN KEY(idUsuario) REFERENCES usuario(idUsuario)
);

CREATE TABLE empresa (
idUsuario INTEGER NOT NULL PRIMARY KEY,
CUIT TEXT NOT NULL,
razonSocial TEXT NOT NULL,
nombreCategoriaEmpresa TEXT NOT NULL,
FOREIGN KEY(idUsuario) REFERENCES usuario(idUsuario)
FOREIGN KEY(nombreCategoriaEmpresa) REFERENCES categoriaEmpresa(nombre)
);

CREATE TABLE categoriaEmpresa (
nombre TEXT NOT NULL PRIMARY KEY,
nombreSuperCategoria TEXT,
FOREIGN KEY(nombreSuperCategoria) REFERENCES categoriaEmpresa(nombre)
);

CREATE TABLE suscripcionRubiOriente (
```

```
idSuscripcion INTEGER NOT NULL PRIMARY KEY,
periodo DATE NOT NULL,
idUsuario INTEGER NOT NULL,
FOREIGN KEY(idUsuario) REFERENCES usuario(idUsuario)
);

CREATE TABLE factura (
idFactura INTEGER NOT NULL PRIMARY KEY,
periodo DATE NOT NULL,
monto INTEGER NOT NULL CHECK(monto > 0),
idUsuario INTEGER NOT NULL,
FOREIGN KEY(idUsuario) REFERENCES usuario(idUsuario)
);

CREATE TABLE publicacion (
idPublicacion INTEGER NOT NULL PRIMARY KEY,
titulo TEXT NOT NULL,
fecha DATE NOT NULL,
precio INTEGER NOT NULL CHECK(precio > 0),
nombreTipoPublicacion TEXT NOT NULL,
tipoVigencia INTEGER NOT NULL,
tipoVenta INTEGER NOT NULL,
idUsuario INTEGER NOT NULL,
FOREIGN KEY(nombreTipoPublicacion) REFERENCES tipoPublicacion(nombre),
FOREIGN KEY(idUsuario) REFERENCES usuario(idUsuario)
);

CREATE TABLE publicacionSubasta (
idPublicacion INTEGER NOT NULL PRIMARY KEY,
FOREIGN KEY(idPublicacion) REFERENCES publicacion(idPublicacion)
);

CREATE TABLE publicacionFinalizada (
idPublicacion INTEGER NOT NULL PRIMARY KEY,
FOREIGN KEY(idPublicacion) REFERENCES publicacion(idPublicacion)
);

CREATE TABLE tipoPublicacion (
nombre TEXT NOT NULL PRIMARY KEY,
comision FLOAT NOT NULL,
costo INTEGER NOT NULL,
nivel INTEGER NOT NULL,
caducidad DATE NOT NULL
```

```
);

CREATE TABLE item (
idItem INTEGER NOT NULL PRIMARY KEY,
idPublicacion INTEGER NOT NULL,
nombre TEXT NOT NULL,
tipo INTEGER NOT NULL,
FOREIGN KEY(idPublicacion) REFERENCES publicacion(idPublicacion)
);

CREATE TABLE producto (
idItem INTEGER NOT NULL PRIMARY KEY,
nombreCategoriaProducto TEXT NOT NULL,
FOREIGN KEY(idItem) REFERENCES item(idItem),
FOREIGN KEY(nombreCategoriaProducto) REFERENCES categoriaProducto(nombre)
);

CREATE TABLE servicio (
idItem INTEGER NOT NULL PRIMARY KEY,
precioPorHora INTEGER NOT NULL CHECK(precioPorHora > 0),
nombreTipoServicio TEXT NOT NULL,
FOREIGN KEY(idItem) REFERENCES item(idItem),
FOREIGN KEY(nombreTipoServicio) REFERENCES tipoServicio(nombre)
);

CREATE TABLE categoriaProducto (
nombre TEXT NOT NULL PRIMARY KEY,
nombreSuperCategoria TEXT,
FOREIGN KEY(nombreSuperCategoria) REFERENCES categoriaProducto(nombre)
);

CREATE TABLE tipoServicio (
nombre TEXT NOT NULL PRIMARY KEY,
nombreSuperTipo TEXT,
FOREIGN KEY(nombreSuperTipo) REFERENCES tipoServicio(nombre)
);

CREATE TABLE compra (
idCompra INTEGER NOT NULL PRIMARY KEY,
idUsuario INTEGER NOT NULL,
idPublicacionFinalizada INTEGER NOT NULL,
idPago INTEGER NOT NULL,
idCalificacion INTEGER NOT NULL,
```

```
idRetiro INTEGER NOT NULL,
FOREIGN KEY(idUsuario) REFERENCES usuario(idUsuario),
FOREIGN KEY(idPublicacionFinalizada) REFERENCES publicacionFinalizada(idPublicacion),
FOREIGN KEY(idPago) REFERENCES pago(idPago),
FOREIGN KEY(idCalificacion) REFERENCES calificacion(idCalificacion),
FOREIGN KEY(idRetiro) REFERENCES retiro(idRetiro)
);

CREATE TABLE pago (
idPago INTEGER NOT NULL PRIMARY KEY,
monto INTEGER NOT NULL CHECK(monto > 0),
tipo INTEGER not NULL
);

CREATE TABLE pagoTarjeta (
idPago INTEGER NOT NULL PRIMARY KEY,
numeroTarjeta INTEGER NOT NULL,
cuotas INTEGER NOT NULL,
FOREIGN KEY(idPago) REFERENCES pago(idPago)
);

CREATE TABLE calificacion (
idCalificacion INTEGER NOT NULL PRIMARY KEY,
valoracionComprador INTEGER CHECK(valoracionComprador BETWEEN 1 AND 10),
valoracionVendedor INTEGER CHECK(valoracionVendedor BETWEEN 1 AND 10),
comentarioComprador TEXT,
comentarioVendedor TEXT
);

CREATE TABLE retiro (
idRetiro INTEGER NOT NULL PRIMARY KEY,
tipo INTEGER not NULL
);

CREATE TABLE envioPostal (
idRetiro INTEGER NOT NULL PRIMARY KEY,
direccion TEXT NOT NULL,
FOREIGN KEY(idRetiro) REFERENCES retiro(idRetiro)
);

CREATE TABLE hizoOferta (
idUsuario INTEGER NOT NULL,
idPublicacion INTEGER NOT NULL,
```

```
fecha DATE NOT NULL,
monto INTEGER NOT NULL CHECK(monto > 0),
PRIMARY KEY(idUsuario, idPublicacion, fecha),
FOREIGN KEY(idUsuario) REFERENCES usuario(idUsuario),
FOREIGN KEY(idPublicacion) REFERENCES publicacionSubasta(idPublicacion)
);

CREATE TABLE pregunta (
idPregunta INTEGER NOT NULL PRIMARY KEY,
idUsuario INTEGER NOT NULL,
idPublicacion INTEGER NOT NULL,
pregunta TEXT NOT NULL,
respuesta TEXT,
FOREIGN KEY(idUsuario) REFERENCES usuario(idUsuario),
FOREIGN KEY(idPublicacion) REFERENCES publicacion(idPublicacion)
);

CREATE TABLE marcoFavorita (
idUsuario INTEGER NOT NULL,
idPublicacion INTEGER NOT NULL,
PRIMARY KEY(idUsuario, idPublicacion),
FOREIGN KEY(idUsuario) REFERENCES usuario(idUsuario),
FOREIGN KEY(idPublicacion) REFERENCES publicacion(idPublicacion)
);

CREATE TABLE visito (
idUsuario INTEGER NOT NULL,
idPublicacion INTEGER NOT NULL,
fecha DATE NOT NULL,
PRIMARY KEY(idUsuario, idPublicacion, fecha),
FOREIGN KEY(idUsuario) REFERENCES usuario(idUsuario),
FOREIGN KEY(idPublicacion) REFERENCES publicacion(idPublicacion)
);

-- Trigger para validar ingreso de oferta
CREATE TRIGGER ofertaValida BEFORE INSERT ON hizoOferta
 WHEN (NEW.monto < (SELECT monto FROM hizoOferta ORDER BY fecha DESC LIMIT 1)
  OR NEW.monto > (SELECT monto FROM hizoOferta ORDER BY fecha DESC LIMIT 1) * 2)
BEGIN
 SELECT RAISE(ABORT, 'El monto de la oferta debe superar el monto actual y ser inferior a
END;
```

B. Funcionalidades adicionales

• Consulta por usuario: obtener, para un usuario específico, información sobre los artículos que ha comprado y vendido, los artículos que ha visitado con su fecha de visita, los artículos que tiene en su lista de favoritos, y las primeras 3 categorías de artículos que visitó con mayor frecuencia en el último año.

```
SELECT publicacion.*
FROM publicacion
INNER JOIN publicacionFinalizada
    ON publicacionFinalizada.idPublicacion = publicacion.idPublicacion
INNER JOIN compra
    ON compra.idPublicacionFinalizada = publicacionFinalizada.idPublicacion
WHERE compra.idUsuario = :idUsuario
UNION
SELECT publicacion.*
FROM publicacion
INNER JOIN publicacionFinalizada
    ON publicacionFinalizada.idPublicacion = publicacion.idPublicacion
WHERE publicacion.idUsuario = :idUsuario;
```

• Consulta por categoría de producto: obtener, dada una categoría de producto ("Computación", "Hogar, muebles y jardín", etc), un listado de los vendedores que han publicado artículos de dicha categoría y la cantidad de ventas que efectuó cada uno de dichos vendedores.

```
SELECT u.idUsuario, count(*)
FROM usuario u
INNER JOIN publicacion pu1 ON u.idUsuario = pu1.idUsuario
INNER JOIN compra co1 ON co1.idPublicacionFinalizada = pu1.idPublicacion
WHERE u.idUsuario IN (
    SELECT DISTINCT pu2.idUsuario
    FROM producto pr
    INNER JOIN item i ON pr.idItem == i.idItem
    INNER JOIN publicacion pu2 ON i.idPublicacion = pu2.idPublicacion
    INNER JOIN compra co2 ON co2.idPublicacionFinalizada = pu2.idPublicacion
    WHERE nombreCategoriaProducto = :categoria
)
GROUP BY u.idUsuario;
```

• Función "Ofertar": debe permitir al usuario ofertar una suma en una subasta. Dicha suma debe ser superior en al menos 1 peso, a la oferta actual, e inferior al doble de la oferta actual.

```
INSERT INTO hizoOferta VALUES (:idUsuario, :idPublicacion, datetime('now'), :monto)
```

• Consulta por usuario y preguntas: obtener para un usuario específico, la lista de preguntas que ha realizado, con las respectivas respuestas que haya recibido (sólo la pregunta, si aún no recibió respuesta).

```
SELECT p.pregunta, p.respuesta
FROM pregunta p
WHERE p.idUsuario = :idUsuario;
```

• Consulta por keyword: obtener, para un cierto keyword (por ejemplo "mesa"), la lista de publicaciones vigentes que tengan en el título, dicha keyword. El usuario debe poder restringir su búsqueda sólo a cierta categoría de artículos o servicios.

```
SELECT p.idPublicacion
FROM publicacion p
WHERE p.tipoVigencia = 0
AND titulo LIKE :patron
AND (
    EXISTS (
        SELECT *
        FROM item i
        INNER JOIN producto pr ON i.idItem = pr.idItem
        WHERE i.idPublicacion = p.idPublicacion
        AND pr.nombreCategoriaProducto = :categoria
    ) OR EXISTS
    (
        SELECT *
        FROM item i
        INNER JOIN servicio s ON i.idItem = s.idItem
        WHERE i.idPublicacion = p.idPublicacion
        AND s.nombreTipoServicio = :categoria
    )
);
```

• Consulta por ganador/es anual de viaje a Khan El-Khalili: obtener, para un año específico, el ganador/es

```
SELECT IFNULL(razonSocial, nombre || apellido) ganador,
        AVG(valoracionVendedor) * SUM(precio) puntaje,
        AVG(valoracionVendedor) calificacion_promedio,
        SUM(precio) suma_montos_abonados
FROM calificacion
INNER JOIN compra ON compra.idCalificacion = calificacion.idCalificacion
INNER JOIN publicacion ON publicacion.idPublicacion = compra.idPublicacionFinalizada
INNER JOIN usuario ON usuario.idUsuario = publicacion.idUsuario
```

```
LEFT JOIN particular ON particular.idUsuario = usuario.idUsuario
LEFT JOIN empresa ON empresa.idUsuario = usuario.idUsuario
GROUP BY usuario.idUsuario
ORDER BY AVG(valoracionVendedor) * SUM(precio) DESC
LIMIT 1;
```