Hace mucho mucho tiempo, en una galaxia muy cercana, el Imperio creó una red de comunicaciones intergaláctica entre sus bases, para permitir la comunicación y el envío de información a cualquier lugar del Imperio, por remoto que éste fuese. Se te ha encargado diseñar la base de datos que se usará para optimizar los servicios de búsqueda en esta red. La información se encuentra organizada en UEBs (Unidades de Exploración de Bases), cada una de ellas indexada con su propia id y especificando necesariamente la dirección a la que conectarse. Las búsquedas en la red, en caso de encontrar la información deseada, pueden mostrar varias de estas UEB para conectarse; por optimización, se debe conocer la posición dentro de la búsqueda en la que aparecía cada UEB y si ésta ha sido visitada tras esto o no. Sobre cada búsqueda, se debe almacenar también el momento en el que se realizó, el texto que la misma contenía y una lista con todas las palabras clave de la misma. Pero obviamente las búsquedas no salen de la nada; todas y cada una de ellas estarán realizadas por un único usuario, que podría estar registrado en el sistema o no. Todos los usuarios se reconocen mediante un ID. Si el usuario no está registrado, este identificador se asocia simplemente a la dirección del protocolo imperial o pi, desde la que se ha realizado la conexión, que por tanto es exclusiva de cada ID genérico. Pero si el usuario está registrado, entonces el sistema tiene almacenado además su id_imperial -el sistema de identificación estándar del Imperio- y la fecha de alta con la que se registró. En cada conexión se almacena y/o añade la pi de la misma a la lista de pi del usuario registrado; éstas obviamente sí que pueden repetirse entre distintos usuarios registrados, nada que ver con la pi correspondiente al usuario genérico desconocido. Es vital para la optimización del sistema que las búsquedas de cada usuario se puedan reconocer ordenadas con el índice exacto en que las realizó, y además este historial será borrado al dar de baja a los usuarios. Por otro lado, la red tiene un sistema de reconocimiento que relaciona las búsquedas con una serie de temas, cada una puede tratar sobre más de un tema pero siempre será asignada al menos a uno de los mismos. Estos temas se reconocen por su nombre y tienen siempre asignado un índice de popularidad, "alta", "media" o "baja". Además, se quiere conocer la fecha de la última consulta relacionada con cada tema. Debes tener presente que algunos temas pueden tener otro tema relacionado, solo uno, que se considera un tema secundario del anterior. No hay inconveniente en que haya temas que se consideren secundarios de varios otros ni en que un tema considerado secundario de otro tenga a su vez un tema secundario.

Modelo Relacional para la Base de Datos

1. Entidades y sus atributos

1. **UEB (Unidad de Exploración de Bases)**
   - `id_ueb` (PK): Identificador único de la UEB.
   - `direccion`: Dirección de conexión de la UEB.
2. **Usuario**
   - `id_usuario` (PK): Identificador único del usuario.
   - `pi`: Dirección del protocolo imperial asociada al usuario.
   - `id_imperial` (NULLABLE): ID estándar del Imperio (para usuarios registrados).
   - `fecha_alta` (NULLABLE): Fecha de alta del usuario registrado.
3. **Búsqueda**
   - `id_busqueda` (PK): Identificador único de la búsqueda.
   - `id_usuario` (FK): Relación con el usuario que realiza la búsqueda.

- `momento`: Timestamp de la búsqueda.
- `texto`: Texto de la búsqueda.

4. **Palabra Clave**
   - `id_palabra` (PK): Identificador único de la palabra clave.
   - `texto`: Palabra clave.
5. **Tema**
   - `id_tema` (PK): Identificador único del tema.
   - `nombre`: Nombre del tema.
   - `popularidad`: Índice de popularidad ("alta", "media", "baja").
   - `ultima_consulta`: Fecha de la última consulta relacionada con el tema.
   - `id_tema_secundario` (FK, NULLABLE): Relación con otro tema considerado secundario.
6. **Historial de Búsqueda**
   - `id_historial` (PK): Identificador único del historial.
   - `id_usuario` (FK): Relación con el usuario que realiza la búsqueda.
   - `id_busqueda` (FK): Relación con la búsqueda específica.
   - `orden`: Índice exacto de la búsqueda en el historial del usuario.
7. **Resultados de Búsqueda**
   - `id_resultado` (PK): Identificador único del resultado.
   - `id_busqueda` (FK): Relación con la búsqueda.
   - `id_ueb` (FK): Relación con la UEB encontrada.
   - `posicion`: Posición de la UEB en los resultados de búsqueda.
   - `visitada`: Booleano que indica si la UEB fue visitada.

2. Relaciones

1. **Palabras Clave de una Búsqueda**
   - Tabla intermedia:
     - `id_busqueda` (FK): Relación con la búsqueda.
     - `id_palabra` (FK): Relación con la palabra clave.
2. **Temas de una Búsqueda**
   - Tabla intermedia:
     - `id_busqueda` (FK): Relación con la búsqueda.
     - `id_tema` (FK): Relación con el tema.

Esquema en SQL

```
sql
CopiarEditar
-- Tabla UEB
CREATE TABLE UEB (
    id_ueb SERIAL PRIMARY KEY,
    direccion VARCHAR(255) NOT NULL
);

-- Tabla Usuario
CREATE TABLE Usuario (
    id_usuario SERIAL PRIMARY KEY,
    pi VARCHAR(255) NOT NULL UNIQUE,
    id_imperial VARCHAR(50),
```

```
    fecha_alta DATE
);

-- Tabla Búsqueda
CREATE TABLE Busqueda (
    id_busqueda SERIAL PRIMARY KEY,
    id_usuario INT NOT NULL,
    momento TIMESTAMP NOT NULL,
    texto TEXT NOT NULL,
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario) ON DELETE
CASCADE
);

-- Tabla Palabra Clave
CREATE TABLE PalabraClave (
    id_palabra SERIAL PRIMARY KEY,
    texto VARCHAR(255) NOT NULL UNIQUE
);

-- Tabla Tema
CREATE TABLE Tema (
    id_tema SERIAL PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL UNIQUE,
    popularidad ENUM('alta', 'media', 'baja') NOT NULL,
    ultima_consulta DATE,
    id_tema_secundario INT,
    FOREIGN KEY (id_tema_secundario) REFERENCES Tema(id_tema)
);

-- Tabla Historial de Búsqueda
CREATE TABLE HistorialBusqueda (
    id_historial SERIAL PRIMARY KEY,
    id_usuario INT NOT NULL,
    id_busqueda INT NOT NULL,
    orden INT NOT NULL,
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario) ON DELETE
CASCADE,
    FOREIGN KEY (id_busqueda) REFERENCES Busqueda(id_busqueda) ON
DELETE CASCADE
);

-- Tabla Resultados de Búsqueda
CREATE TABLE ResultadoBusqueda (
    id_resultado SERIAL PRIMARY KEY,
    id_busqueda INT NOT NULL,
    id_ueb INT NOT NULL,
    posicion INT NOT NULL,
    visitada BOOLEAN NOT NULL,
    FOREIGN KEY (id_busqueda) REFERENCES Busqueda(id_busqueda) ON
DELETE CASCADE,
    FOREIGN KEY (id_ueb) REFERENCES UEB(id_ueb) ON DELETE CASCADE
);

-- Relación Palabras Clave de una Búsqueda
CREATE TABLE BusquedaPalabraClave (
    id_busqueda INT NOT NULL,
    id_palabra INT NOT NULL,
    PRIMARY KEY (id_busqueda, id_palabra),
    FOREIGN KEY (id_busqueda) REFERENCES Busqueda(id_busqueda) ON
DELETE CASCADE,
```

```
    FOREIGN KEY (id_palabra) REFERENCES PalabraClave(id_palabra) ON
DELETE CASCADE
);

-- Relación Temas de una Búsqueda
CREATE TABLE BusquedaTema (
    id_busqueda INT NOT NULL,
    id_tema INT NOT NULL,
    PRIMARY KEY (id_busqueda, id_tema),
    FOREIGN KEY (id_busqueda) REFERENCES Busqueda(id_busqueda) ON
DELETE CASCADE,
    FOREIGN KEY (id_tema) REFERENCES Tema(id_tema) ON DELETE CASCADE
);
```

Resolución del problema: diseño de la base de datos

---

1. Modelo Conceptual (E-R)

Primero definimos las entidades, atributos y relaciones basados en el enunciado:

- **Entidades principales**:
    1. **UEB (Unidad de Exploración de Bases)**
        - Atributos: `id_ueb (PK)`, `direccion`.
    2. **Usuario**
        - Atributos: `id_usuario (PK)`, `pi`, `id_imperial (opcional)`, `fecha_alta (opcional)`.
    3. **Búsqueda**
        - Atributos: `id_busqueda (PK)`, `momento`, `texto`.
    4. **Palabra Clave**
        - Atributos: `id_palabra (PK)`, `texto`.
    5. **Tema**
        - Atributos: `id_tema (PK)`, `nombre`, `popularidad`, `ultima_consulta`, `id_tema_secundario (FK opcional)`.

---

2. Modelo Relacional

Tablas derivadas:

1. **UEB**
    - `id_ueb` (PK)
    - `direccion` (NOT NULL)
2. **Usuario**
    - `id_usuario` (PK)
    - `pi` (NOT NULL, UNIQUE para usuarios genéricos)
    - `id_imperial` (NULLABLE)
    - `fecha_alta` (NULLABLE)
3. **Búsqueda**
    - `id_busqueda` (PK)

    - `id_usuario` (FK, NOT NULL)
    - `momento` (NOT NULL)
    - `texto` (NOT NULL)
4. **PalabraClave**
    - `id_palabra` (PK)
    - `texto` (NOT NULL, UNIQUE)
5. **Tema**
    - `id_tema` (PK)
    - `nombre` (NOT NULL, UNIQUE)
    - `popularidad` (NOT NULL, valores: "alta", "media", "baja")
    - `ultima_consulta` (NULLABLE)
    - `id_tema_secundario` (FK NULLABLE)
6. **HistorialBusqueda**
    - `id_historial` (PK)
    - `id_usuario` (FK, NOT NULL)
    - `id_busqueda` (FK, NOT NULL)
    - `orden` (NOT NULL)
7. **ResultadoBusqueda**
    - `id_resultado` (PK)
    - `id_busqueda` (FK, NOT NULL)
    - `id_ueb` (FK, NOT NULL)
    - `posicion` (NOT NULL)
    - `visitada` (BOOLEAN, NOT NULL)

Tablas intermedias:

1. **BusquedaPalabraClave**
    - `id_busqueda` (FK, NOT NULL)
    - `id_palabra` (FK, NOT NULL)
2. **BusquedaTema**
    - `id_busqueda` (FK, NOT NULL)
    - `id_tema` (FK, NOT NULL)

---

3. Modelo Físico DDL (MySQL)

Creación de tablas

```
sql
CopiarEditar
-- Tabla UEB
CREATE TABLE UEB (
    id_ueb INT AUTO_INCREMENT PRIMARY KEY,
    direccion VARCHAR(255) NOT NULL
);

-- Tabla Usuario
CREATE TABLE Usuario (
    id_usuario INT AUTO_INCREMENT PRIMARY KEY,
    pi VARCHAR(255) NOT NULL UNIQUE,
    id_imperial VARCHAR(50),
```

```
    fecha_alta DATE
);

-- Tabla Búsqueda
CREATE TABLE Busqueda (
    id_busqueda INT AUTO_INCREMENT PRIMARY KEY,
    id_usuario INT NOT NULL,
    momento DATETIME NOT NULL,
    texto TEXT NOT NULL,
    CONSTRAINT fk_busqueda_usuario FOREIGN KEY (id_usuario) REFERENCES
Usuario(id_usuario) ON DELETE CASCADE
);

-- Tabla Palabra Clave
CREATE TABLE PalabraClave (
    id_palabra INT AUTO_INCREMENT PRIMARY KEY,
    texto VARCHAR(255) NOT NULL UNIQUE
);

-- Tabla Tema
CREATE TABLE Tema (
    id_tema INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL UNIQUE,
    popularidad ENUM('alta', 'media', 'baja') NOT NULL,
    ultima_consulta DATE,
    id_tema_secundario INT,
    CONSTRAINT fk_tema_secundario FOREIGN KEY (id_tema_secundario)
REFERENCES Tema(id_tema)
);

-- Tabla Historial de Búsqueda
CREATE TABLE HistorialBusqueda (
    id_historial INT AUTO_INCREMENT PRIMARY KEY,
    id_usuario INT NOT NULL,
    id_busqueda INT NOT NULL,
    orden INT NOT NULL,
    CONSTRAINT fk_historial_usuario FOREIGN KEY (id_usuario)
REFERENCES Usuario(id_usuario) ON DELETE CASCADE,
    CONSTRAINT fk_historial_busqueda FOREIGN KEY (id_busqueda)
REFERENCES Busqueda(id_busqueda) ON DELETE CASCADE
);

-- Tabla Resultados de Búsqueda
CREATE TABLE ResultadoBusqueda (
    id_resultado INT AUTO_INCREMENT PRIMARY KEY,
    id_busqueda INT NOT NULL,
    id_ueb INT NOT NULL,
    posicion INT NOT NULL,
    visitada BOOLEAN NOT NULL,
    CONSTRAINT fk_resultado_busqueda FOREIGN KEY (id_busqueda)
REFERENCES Busqueda(id_busqueda) ON DELETE CASCADE,
    CONSTRAINT fk_resultado_ueb FOREIGN KEY (id_ueb) REFERENCES
UEB(id_ueb) ON DELETE CASCADE
);

-- Relación Palabras Clave y Búsquedas
CREATE TABLE BusquedaPalabraClave (
    id_busqueda INT NOT NULL,
    id_palabra INT NOT NULL,
    PRIMARY KEY (id_busqueda, id_palabra),
```

```
    CONSTRAINT fk_bpc_busqueda FOREIGN KEY (id_busqueda) REFERENCES
Busqueda(id_busqueda) ON DELETE CASCADE,
    CONSTRAINT fk_bpc_palabra FOREIGN KEY (id_palabra) REFERENCES
PalabraClave(id_palabra) ON DELETE CASCADE
);

-- Relación Temas y Búsquedas
CREATE TABLE BusquedaTema (
    id_busqueda INT NOT NULL,
    id_tema INT NOT NULL,
    PRIMARY KEY (id_busqueda, id_tema),
    CONSTRAINT fk_bt_busqueda FOREIGN KEY (id_busqueda) REFERENCES
Busqueda(id_busqueda) ON DELETE CASCADE,
    CONSTRAINT fk_bt_tema FOREIGN KEY (id_tema) REFERENCES
Tema(id_tema) ON DELETE CASCADE
);
```

---

4. Normalización

1. Todas las tablas están en **1FN** al eliminar atributos multivaluados y asegurarse de que todos los campos contienen valores atómicos.
2. En **2FN**, se elimina la dependencia parcial asegurando que cada campo no clave dependa completamente de la clave principal.
3. En **3FN**, eliminamos la dependencia transitiva, por lo que cada atributo no clave depende únicamente de la clave primaria.

1. Modelo relacional normalizado (3 PUNTOS) Realiza el paso a tablas del siguiente diagrama E-R, siguiendo el modelo relacional, con las notaciones vistas en las TC. Realiza una normalización de todas las tablas a 3FN y explica las posibles restricciones (si las hay), pérdidas semánticas (si las hay) y las posibles debilidades (si las hay). Enumera tablas y restricciones, indica los pasos cuando normalices y justifica cada decisión de manera breve y concisa, dibujando los diagramas de dependencia solo cuando provoquen cambios en las tablas. Sé coherente con la notación. DAW/DAM. EXAMEN 1ª EVALUACIÓN. Modelo B 2 de 6 2. Modelo físico DDL 2.1. Modelo físico DDL (crear metadatos) 2.2. Modelo físico DDL (modificar metadatos) (10 PUNTOS) (6,5 PUNTOS) Indica las sentencias SQL necesarias en MySQL, en el orden adecuado, para crear las tablas necesarias para modelar el diagrama E-R del apartado 1 anterior, usando la palabra reservada CONSTRAINT siempre que tengas la posibilidad de hacerlo. El atributo cod_mina es de tipo texto, número de galería es entero positivo, las fechas (día y f_liberación) son de tipo fecha. La clasificación se refiere a las 3 posibles cualificaciones profesionales de un minero: Peón, Oficial, Capataz. Un planeta está ubicado en un sistema estelar. Una mina se encuentra en un planeta. (3,5 PUNTOS) Imagina que la base de datos ya está creada y tus sentencias SQL del ejercicio anterior se han ejecutado correctamente. Indica las sentencias necesarias en el orden adecuado para implementar los siguientes cambios: • El Imperio quiere saber el precio de compra de un esclavo, que debe tener 2 decimales y oscilar entre -1.11 y 10.99 • El comité de explotación minera ha decidido que un esclavo puede ser sustituido por otros, aunque solo puede sustituir a uno. DAW/DAM. EXAMEN 1ª EVALUACIÓN. Modelo B 3 de 6 3. Modelo físico DML (manipular datos) (10 PUNTOS) Suponiendo que tus sentencias SQL de los ejercicios anteriores se han ejecutado correctamente, indica las sentencias necesarias en el orden adecuado para añadir o modificar los datos en las tablas: (Inventa los valores de los campos que no te indiquemos) • Crea una mina con

cod_mina S1CIN y descripción Explotación de cinabrio. • Crea una galería en la mina anterior, con número 1. La galería pertenece al sector A1. • Crea un minero con expediente M01, nombre Cuñao, apellidos Chewbacca. • Este minero trabaja en la galería 1 de la mina anterior con fecha el día de hoy. • El minero deja de trabajar en la mina.

Resolución paso a paso

---

1. Modelo relacional normalizado

Paso a tablas del diagrama E-R (basado en el examen)

1. **Entidad Mina**
   - **Atributos**:
     - `cod_mina` (PK)
     - `descripcion`
2. **Entidad Galería**
   - **Atributos**:
     - `num_galeria` (PK)
     - `cod_mina` (FK, referencia a Mina)
     - `sector`
3. **Entidad Minero**
   - **Atributos**:
     - `expediente` (PK)
     - `nombre`
     - `apellidos`
     - `clasificacion` (valores posibles: Peón, Oficial, Capataz)
4. **Relación Trabaja**
   - **Atributos**:
     - `expediente` (FK, referencia a Minero)
     - `num_galeria` (FK, referencia a Galería)
     - `cod_mina` (FK, referencia a Mina)
     - `dia` (fecha en que trabaja)

Normalización

1. **Primera Forma Normal (1FN)**:
   Todas las tablas tienen atributos atómicos y no multivaluados.
2. **Segunda Forma Normal (2FN)**:
   - La tabla Trabaja tiene clave compuesta (`expediente`, `num_galeria`, `cod_mina`). Todos los atributos dependen completamente de la clave primaria.
3. **Tercera Forma Normal (3FN)**:
   - No existen dependencias transitivas.

Restricciones

- **Integridad referencial**:
  - `cod_mina` en Galería referencia a Mina.
  - `expediente` en Trabaja referencia a Minero.
  - `num_galeria` en Trabaja referencia a Galería.
- **Pérdidas semánticas**: Ninguna pérdida significativa.
- **Debilidades**: Relación "Trabaja" puede crecer mucho si hay muchas asociaciones entre mineros y galerías.

---

2. Modelo físico DDL

2.1. Crear metadatos

```
sql
CopiarEditar
-- Tabla Mina
CREATE TABLE Mina (
    cod_mina VARCHAR(10) PRIMARY KEY,
    descripcion VARCHAR(255) NOT NULL
);

-- Tabla Galería
CREATE TABLE Galeria (
    num_galeria INT UNSIGNED PRIMARY KEY,
    cod_mina VARCHAR(10) NOT NULL,
    sector VARCHAR(10),
    CONSTRAINT fk_galeria_mina FOREIGN KEY (cod_mina) REFERENCES
Mina(cod_mina) ON DELETE CASCADE
);

-- Tabla Minero
CREATE TABLE Minero (
    expediente VARCHAR(10) PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    apellidos VARCHAR(50) NOT NULL,
    clasificacion ENUM('Peón', 'Oficial', 'Capataz') NOT NULL
);

-- Tabla Trabaja
CREATE TABLE Trabaja (
    expediente VARCHAR(10) NOT NULL,
    num_galeria INT UNSIGNED NOT NULL,
    cod_mina VARCHAR(10) NOT NULL,
    dia DATE NOT NULL,
    PRIMARY KEY (expediente, num_galeria, cod_mina),
    CONSTRAINT fk_trabaja_minero FOREIGN KEY (expediente) REFERENCES
Minero(expediente) ON DELETE CASCADE,
    CONSTRAINT fk_trabaja_galeria FOREIGN KEY (num_galeria) REFERENCES
Galeria(num_galeria) ON DELETE CASCADE,
    CONSTRAINT fk_trabaja_mina FOREIGN KEY (cod_mina) REFERENCES
Mina(cod_mina) ON DELETE CASCADE
);
```

2.2. Modificar metadatos

1. **Añadir precio de compra de esclavos**

```
sql
CopiarEditar
ALTER TABLE Minero
ADD COLUMN precio_compra DECIMAL(4, 2) CHECK (precio_compra
BETWEEN -1.11 AND 10.99);
```

2. **Permitir sustitución de esclavos**

```
sql
CopiarEditar
ALTER TABLE Minero
ADD COLUMN sustituto VARCHAR(10),
ADD CONSTRAINT fk_sustituto FOREIGN KEY (sustituto) REFERENCES
Minero(expediente);
```

---

3. Modelo físico DML

Insertar datos en las tablas

1. **Crear una mina**

```
sql
CopiarEditar
INSERT INTO Mina (cod_mina, descripcion)
VALUES ('S1CIN', 'Explotación de cinabrio');
```

2. **Crear una galería**

```
sql
CopiarEditar
INSERT INTO Galeria (num_galeria, cod_mina, sector)
VALUES (1, 'S1CIN', 'A1');
```

3. **Crear un minero**

```
sql
CopiarEditar
INSERT INTO Minero (expediente, nombre, apellidos,
clasificacion)
VALUES ('M01', 'Cuñao', 'Chewbacca', 'Peón');
```

4. **Asignar trabajo al minero**

```
sql
CopiarEditar
INSERT INTO Trabaja (expediente, num_galeria, cod_mina, dia)
VALUES ('M01', 1, 'S1CIN', CURDATE());
```

5. **Eliminar al minero de la mina**

```
sql
CopiarEditar
DELETE FROM Trabaja
WHERE expediente = 'M01' AND num_galeria = 1 AND cod_mina =
'S1CIN';
```
