Análisis y Diseño de Bases de Datos
2024-2025

Memoria: Base de Datos Ocupación Vía Pública Málaga 2024

Base de datos: ocupación de la vía pública de Málaga

Grupo G25
Almaraz Arranz, Marcos
Carbajo Orgaz, Ainhoa
De Diego Martín, Marcos
Peña Marqués, Rodrigo

Índice

Introducción

La siguiente memoria describe el análisis y diseño de un sistema de base de datos a partir de la Ordenanza Reguladora de la Ocupación de la Vía Pública del Ayuntamiento de Málaga.

A continuación se detallará en profundidad el análisis de requisitos, especificando qué información forma parte de la base de datos; el diseño conceptual, representado mediante un diagrama Entidad-Relación en notación UML que detallará las entidades definidas; el diseño lógico, mediante un diagrama relacional que reflejará el esquema anteriormente mencionado mostrando la traducción a tablas realizada; y finalmente la implementación del diseño lógico en una base de datos relacional incluyendo la creación, borrado, modificación y recuperación de datos.

Además, se incluyen las consultas para los esquemas propuestos.

Para la realización de la base de datos se ha utilizado *PostgreSQL* con una implementación estándar del diseño lógico, salvo en unas restricciones concretas que se mencionarán en el último apartado.

Diseño Conceptual

La lectura de la ordenanza llevó a un primer análisis superficial que identificase los datos más importantes. La base de datos será un sistema de gestión y almacenamiento de las solicitudes de ocupación de la vía pública.

Tras un análisis en profundidad y debido a la extensión de los requisitos, se decidió que la base de datos almacenaría exclusivamente las solicitudes autorizadas, de aquí en adelante llamadas *autorizaciones* de los cuatro títulos disponibles. Por consecuencia, las restricciones que impliquen dos tablas del diagrama relacional no serán comprobadas por esta base de datos y serán una cuestión relegada al cargo de a la implementación bien de la interfaz o del sistema de gestión externo a este trabajo.

La base de datos conservará las posibles infracciones asociadas a cada autorización, titulares y establecimientos. Se concretará un establecimiento para cada autorización, pudiendo consultar todas las autorizaciones por parte de un mismo establecimiento o titular. De igual forma, se permite la consulta del historial de titulares de un establecimiento.

La gestión de elementos (sillas, mesas, sombrillas, toldos...) se llevará a cabo mediante una petición, la cual registrará el número de elementos del mismo tipo. La causa de esta decisión es la posibilidad de realizar varias peticiones en una misma autorización (por ejemplo*: 4 sillas, 1 mesa*) de varios tipos de elementos siempre que las restricciones lo permitan (una autorización de tipo 1 no podrá solicitar elementos de tipo *toldo*).

A continuación, se resumen las entidades identificadas:

- Autorización: identificada mediante *idAutorizacion* constituye una superclase que contiene los atributos comunes a los cuatro tipos de autorizaciones definidas en el documento.
- Infracción: identificada mediante *idInfracción* registra del incumplimiento de las normas asignado a una autorización.
- Autorizacion1y2: entidad que representa a las autorizaciones que forman parte tanto del Título I como del Título II. Serán distinguibles mediante el atributo *tipo*.
- Autorización3: entidad que representa a las autorizaciones que forman parte del título 3, las cuales se referirán a un *Establecimiento*.
- Autorización4: entidad que representa a las autorizaciones que forman parte del título 4, las cuales se referirán a un *Titular*.
- Petición: reflejará la cantidad de elementos de un mismo tipo que solicita una autorización.
- Elemento: mobiliario que solicita la autorización y los datos referentes a sus características. Debido a la variedad de los mismo, se distinguirá mediante el atributo *tipoElemento*.
- Toldo: entidad identificada mediante el atributo *codigoReferencia* de tipo *toldo*.
- Establecimiento: local asociado a la solicitud, identificado mediante el *isd*.
- Titularidad: modela la relación entre Titular y Establecimiento indicando la fecha de inicio. Dado que un establecimiento no puede tener dos titulares simultáneos se define únicamente la fecha de inicio de la titularidad, considerando que coincidirá con la de fin del titular anterior.

- Titular: identificada mediante el *idf* [1] persona física o jurídica dueña del establecimiento.

El diagrama representado en la Figura 1 se ha llevado a cabo utilizando la notación UML estándar, por lo que las claves foráneas estarán representadas mediante las líneas de asociación entre entidades. De esta manera destacamos que *Petición* y *Titularidad* son entidades débiles (debilidad de identificación), ya que su identificación depende de sus dos clases adyacentes: *Petición* toma el de *Autorizacion1y2,* así como el de *Elemento* y *Titularidad* el de *Establecimiento* y *Titular*.

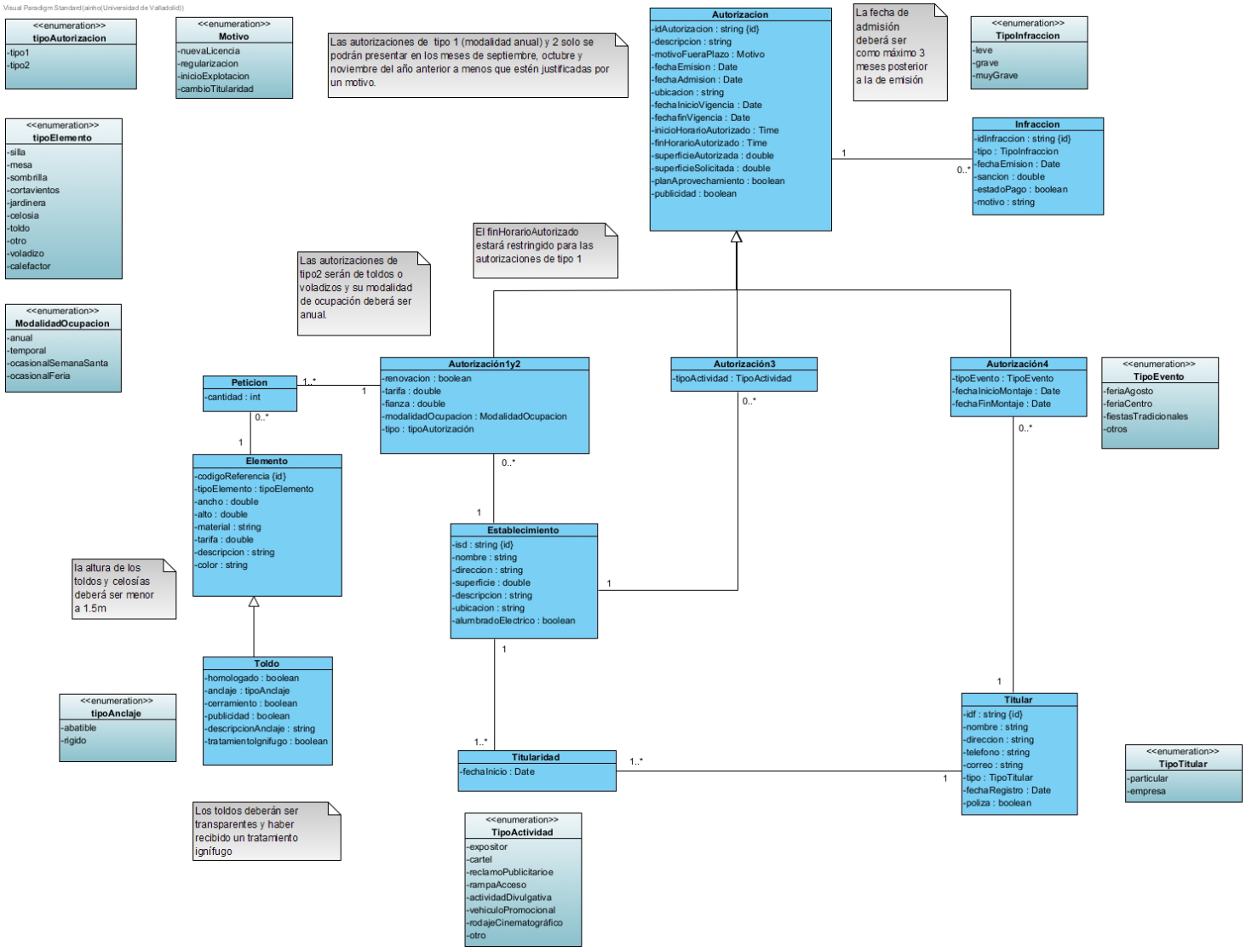

Figura 1.Diagrama Entidad-Relación. Fuente: Elaboración propia.

[1] Idf: *número de identificación fiscal*

Diseño Lógico

A la hora de reflejar el esquema anterior en un modelo relacional (representado en la Figura 2), se optó por 2 tablas para las autorizaciones: una para la superclase y las subclases referentes a los tipos 3 y 4; y otra tabla para la subclase referente a las autorizaciones de tipo 1 y 2. Es una solución intermedia entre la especialización total (1 tabla para cada entidad) y la genericidad (1 tabla para todas las entidades de tipo Autorización). La primera opción complicaría en exceso el diseño y las consultas, mientras que la segunda no sería lo suficientemente concreta para ser eficiente en el mundo real y generaría numerosos nulos.

Esta decisión se basó en la limitada cantidad de información que diferencia los títulos III y IV, en lo que se refiere a atributos no genera una cantidad de nulos excesiva.

Por otro lado, la principal diferencia entre los títulos I y II radica en los elementos solicitados (Toldos para el título II) y en la temporalidad (la modalidad de ocupación debe ser anual para el título II obligatoriamente). Esto se ha resuelto mediante un atributo *tipo* que indica el tipo de autorización, y mediante la especificación de una subclase *Toldo* para *Elemento*. A su vez, estas dos últimas entidades se almacenarán cada una en una tabla.

Este diseño facilita actualizaciones consistentes (si se elimina un establecimiento, es fácil garantizar que se eliminen automáticamente las autorizaciones asociadas).

A pesar de que la existencia de claves compuestas puede complicar operaciones de búsqueda y aumentar la complejidad de las consultas SQL, las relaciones pretenden reflejar el mundo real de la mejor manera posible y permitir una gestión concreta de cada concesión por parte de la administración responsable.

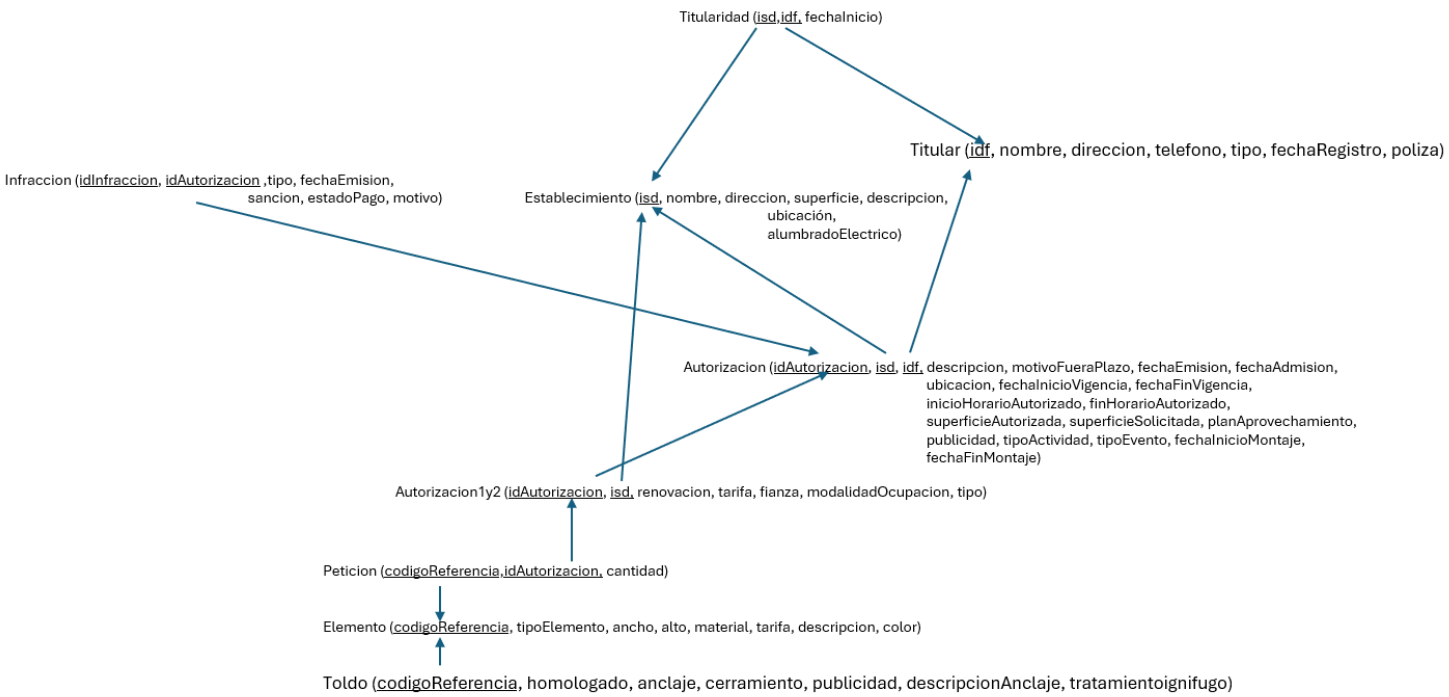

Figura 2.Esquema Relacional. Fuente: Elaboración propia.

Mantenemos la integridad referencial de la base de datos aplicando la cláusula "ON DELETE CASCADE ON UPDATE CASCADE" en las claves foráneas. Esto asegura que cualquier modificación en la clave primaria de una tabla se refleje automáticamente en todas las filas relacionadas de las tablas dependientes. Asimismo, si una fila de la tabla primaria es eliminada, todas las filas asociadas en las tablas hijas también serán eliminadas, evitando registros huérfanos y garantizando la consistencia de los datos en la base de datos.

Consultas Implementadas

1. **Consulta de Autorización con su Titular actual y Establecimiento**
    - **Relevancia:** Proporciona una visión integral de las autorizaciones y su relación con titulares y establecimientos. Es útil para gestionar autorizaciones y determinar la asignación de establecimientos.
    - **Particularidad:** Utiliza la unión de tres subconjuntos para incluir diferentes casos de autorización y relación con titulares o establecimientos.
2. **Autorizaciones Título I**
    - **Relevancia:** Clasifica las autorizaciones del título I.
3. **Autorizaciones Título II**
    - **Relevancia:** Clasifica las autorizaciones del título II.
4. **Autorizaciones Título III**
    - **Relevancia:** Clasifica las autorizaciones del título III.
5. **Autorizaciones Título IV**
    - **Relevancia:** Clasifica las autorizaciones del título IV.
6. **Lista de Titulares y Titularidades**
    - **Relevancia:** Detalla la relación entre titulares y establecimientos a lo largo del tiempo. Útil para consultas históricas y auditorías.
7. **Historial de Titulares de un Establecimiento**
    - **Relevancia:** Muestra cómo han cambiado los titulares de un establecimiento específico, en este caso, "La Esquina".
8. **Historial de Autorizaciones de un Establecimiento**
    - **Relevancia:** Permite auditar y analizar todas las autorizaciones otorgadas a un establecimiento, como "El Puerto".
9. **Autorizaciones fuera de Plazo**
    - **Relevancia:** aquellas autorizaciones que por un motivo justificado no respetas los plazos de presentación

10. **Historial de infracciones**
    - **Relevancia**: permite consultar todas aquellas infracciones relacionadas con una autorizacion o establecimiento

Anexo I : SQL estándar

El script realizado está escrito en SQL estándar salvo dos contadas ocasiones:

CASE

En la consulta marcada como 8 se emplea la expresión CASE simplemente como medio para una presentación más clara. Esta expresión auqneu funciona para la mayoría de SQL no es estándar.

::TEXT

Con los mismo fines de presentación se utiliza :: TEXT en la misma consulta.

TRIGGERS

En un principio la base de datos pretendía ser más restrictiva, el diseño lógico obligaba a establecer restricciones para varias tablas mediante el uso de triggers.

Estos se utilizaron para comprobar restricciones relacionadas con los horarios autorizados, las tarifas y las ocndicioens necesarias para los Toldos.
Por no ser estándar no forman parte de la base de datos y simplemente se adjuntan en este documento como muestra del trabajo realizado.

```
CREATE OR REPLACE FUNCTION validar_fin_horario_autorizacion()
RETURNS TRIGGER AS $$
BEGIN
    -- Verificar si el tipo de elemento requiere la restricción
    IF EXISTS (
        SELECT 1
        FROM Peticion p
        JOIN Elemento e ON p.codigoReferencia = e.codigoReferencia
        WHERE p.idAutorizacion = NEW.idAutorizacion
        AND e.tipoElemento IN ('mesa', 'silla', 'celosia', 'sombrilla')
    ) THEN
        IF NOT (NEW.finHorarioAutorizado <= TIME '12:00:00'
        ) THEN RAISE EXCEPTION 'finHorarioAutorizado debe ser 00:00 para los
elementos especificados.';
        END IF;
        IF NOT (EXTRACT(DOW FROM NEW.fechaInicioVigencia) IN (5, 6)
                OR NEW.fechaInicioVigencia = '2024-12-24' OR
                NEW.finHorarioAutorizado <= TIME '00:30'
        ) THEN RAISE EXCEPTION 'Condiciones especiales no cumplidas para
finHorarioAutorizado.';
        END IF;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
-- Crear trigger asociado a la tabla
CREATE TRIGGER check_fin_horario_autorizado
BEFORE INSERT OR UPDATE ON Autorizacion1y2
FOR EACH ROW
EXECUTE FUNCTION validar_fin_horario_autorizacion();
CREATE OR REPLACE FUNCTION validar_tarifa()
RETURNS TRIGGER AS $$
BEGIN
    -- Si hay plan de aprovechamiento, reducir la tarifa a la mitad
    IF (SELECT planAprovechamiento FROM Autorizacion WHERE idAutorizacion =
NEW.idAutorizacion)THEN
        NEW.tarifa := NEW.tarifa / 2;
    END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Asignar el trigger a la tabla
CREATE TRIGGER validar_tarifa
BEFORE INSERT OR UPDATE ON
Autorizacion1y2
FOR EACH ROW
EXECUTE FUNCTION validar_tarifa();
```

```
-- Función para validar la restricción del toldo
CREATE OR REPLACE FUNCTION validar_toldo()
RETURNS TRIGGER AS $$
BEGIN
    -- Validar si existe un toldo con color 'transparente'
    IF EXISTS (
        SELECT 1
        FROM Elemento E
        JOIN Toldo T ON E.codigoReferencia = T.codigoReferencia
        WHERE E.codigoReferencia = NEW.codigoReferencia
          AND E.color = 'transparente'
          AND T.tratamientoIgnifugo = FALSE -- Agregar condiciones
adicionales si es necesario
    ) THEN
        RAISE EXCEPTION 'El toldo de color transparente debe tener
tratamiento ignífugo.';
    END IF;
    -- Si pasa la validación, devolver la fila
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;


-- Crear el trigger asociado a la tabla Toldo
CREATE TRIGGER check_toldo_restriccion
BEFORE INSERT OR UPDATE ON Toldo
FOR EACH ROW
EXECUTE FUNCTION validar_toldo();


CREATE OR REPLACE FUNCTION verificar_modalidad_anual()
RETURNS TRIGGER AS $$
BEGIN
    -- Verificar si el tipo de elemento es 'toldo' y la modalidad de
ocupación no es 'anual'
    IF EXISTS (
        SELECT 1
        FROM Peticion P
        JOIN Elemento E ON P.codigoReferencia = E.codigoReferencia
        JOIN Autorizacion1y2 A ON A.idAutorizacion = P.idAutorizacion
        WHERE E.tipoElemento = 'toldo'
          AND A.modalidadOcupacion <> 'anual'
    ) THEN
        RAISE EXCEPTION 'El tipo de elemento es "toldo" y la modalidad de
ocupación debe ser "anual"';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER trigger_comprobar_modalidad_anual
BEFORE INSERT OR UPDATE ON Peticion
FOR EACH ROW
EXECUTE FUNCTION verificar_modalidad_anual();
```