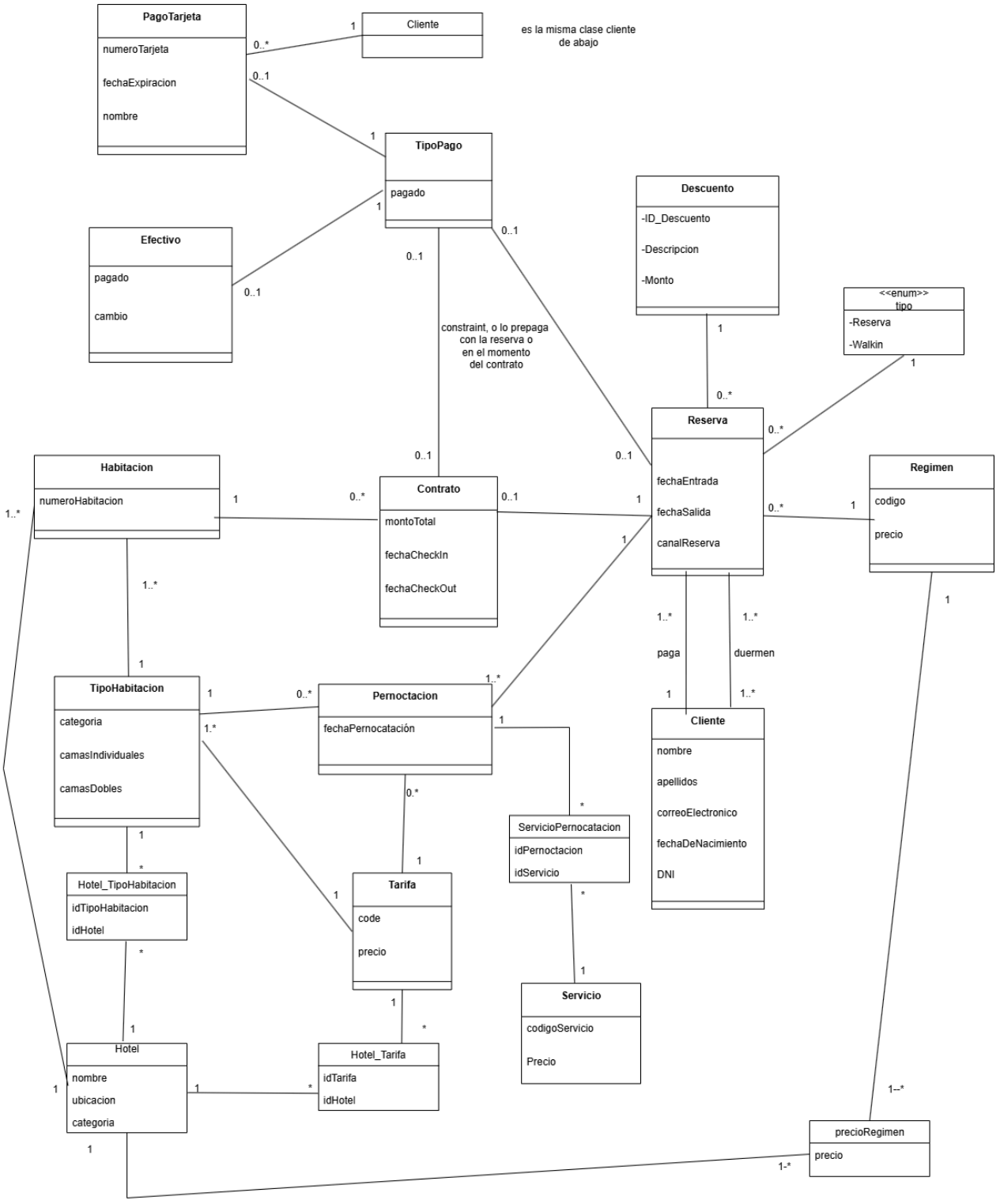
PMS. Modelado de la base de datos

Solucions pel Sector Turístic.

Nasim Hosam Benyacoub Terki.

Modelo Conceptual.



1. Introducción

El presente documento tiene como objetivo principal describir y justificar en profundidad el modelo conceptual diseñado para un Sistema de Gestión Hotelera (PMS). En la industria de la hospitalidad, un PMS es la herramienta central que orquesta todas las operaciones relacionadas con el huésped, desde la reserva inicial hasta la facturación y el check-out. Un diseño de base de datos robusto y bien fundamentado es, por tanto, el pilar sobre el que se construye un sistema exitoso.

A diferencia de un documento técnico de implementación, este análisis se centra en explicar cómo las entidades, sus atributos y relaciones reflejan fielmente las reglas de negocio y los procesos operativos de un establecimiento hotelero. El propósito es demostrar la coherencia y robustez del diseño conceptual, explicando las decisiones tomadas para modelar escenarios complejos como el proceso de reserva, la asignación de habitaciones, la gestión de pagos y la contratación de servicios adicionales.

El alcance de este modelo se centra en el núcleo de la gestión de huéspedes, dejando fuera deliberadamente otros módulos como la contabilidad interna, la gestión de recursos humanos o el mantenimiento de instalaciones, que podrían integrarse en el futuro.

2. Descripción del Proceso Central: De la Reserva al Check-Out

El modelo conceptual se articula en torno a un flujo de proceso lógico que representa el ciclo de vida de la estancia de un cliente, también conocido como el "viaje del huésped" (guest journey). Este proceso se puede dividir en dos grandes fases, cada una representada por entidades clave en el diagrama:

1. **Fase de Reserva (Conceptual):** El viaje comienza cuando un `Cliente` solicita alojamiento para unas fechas determinadas. En esta fase, el sistema no asigna una habitación física, sino que garantiza la disponibilidad de un *tipo* de habitación. La entidad principal aquí es **Reserva**, que actúa como un expediente que agrupa las condiciones del futuro alojamiento, incluyendo una o más **Pernoctacion**es.
2. **Fase de Estancia (Física):** Cuando el cliente llega al hotel para realizar el check-in, la reserva conceptual se materializa en una estancia real. Es en este momento cuando se crea un **Contrato**, se le asigna una **Habitacion** específica y se gestiona el pago final. Esta fase abarca toda la duración de la estancia y termina con el check-out del cliente.

A continuación, se detallan las decisiones de diseño tomadas para modelar cada parte de este proceso de manera exhaustiva.

3. Análisis de Decisiones de Diseño

3.1. Reserva y Pernoctación: El Núcleo Conceptual

La decisión más importante en el diseño es separar la idea de `Reserva` de la ocupación física de una habitación. Esta abstracción es fundamental para una gestión hotelera flexible y eficiente.

- **Entidad `Reserva`:** Actúa como un contenedor que agrupa las condiciones de una futura estancia: fechas (`fechaEntrada`, `fechaSalida`), el cliente responsable (`paga`), el régimen de comidas (`Regimen`) y el tipo de cliente (`tipo`). Al tratarla como un "expediente" conceptual, se evita el bloqueo prematuro de recursos físicos (habitaciones), permitiendo una gestión de inventario mucho más dinámica. Además, se ha establecido la regla de negocio de que cada `Reserva` corresponde a una única habitación. Si un cliente o grupo desea reservar varias habitaciones, deberá generar una reserva independiente para cada una, aunque todas puedan estar asociadas al mismo cliente pagador.
- **Entidad `Pernoctacion`:** Se ha decidido desglosar cada noche de la estancia en una entidad `Pernoctacion`. Aunque el precio de la habitación es único por temporada, esta granularidad sigue siendo fundamental. Permite un control detallado sobre los servicios contratados para noches específicas (por ejemplo, añadir una cuna solo las dos primeras noches de la estancia) y facilita la gestión del inventario y el estado de las habitaciones a un nivel diario, lo cual es esencial para las operaciones de limpieza y mantenimiento.
- **`TipoHabitacion` vs. `Habitacion`:** En la fase de reserva, el sistema se compromete a proporcionar un `TipoHabitacion` (ej: "Doble con vistas al mar"), no una `Habitacion` específica (ej: "la habitación 305"). Esta decisión es vital para la operativa del hotel:
    - **Optimización:** Permite al personal de recepción asignar la habitación concreta en el último momento, optimizando la limpieza y la distribución de huéspedes.
    - **Flexibilidad:** Facilita la gestión de imprevistos. Si la habitación 305 tiene una avería, se le puede asignar al cliente la 306 (del mismo tipo) sin alterar la reserva.
    - **Overbooking Controlado:** Permite al hotel gestionar un nivel controlado de overbooking por tipo de habitación, una práctica común en la industria.
- **Clientes `Walk-in`:** e ha tomado la decisión de unificar el tratamiento de los clientes con reserva previa y los clientes `Walk-in` (que llegan sin reserva). Para ello, la entidad `Reserva` está vinculada a una enumeración `Tipo` que la clasifica explícitamente como 'Reserva' o 'Walkin'. Aunque ambos generan una entidad `Reserva`, esta clasificación formal permite diferenciarlos a nivel de datos. A partir de su llegada, el proceso que siguen es idéntico. La alternativa sería tener un flujo completamente separado, lo que duplicaría la lógica de negocio y la complejidad del sistema. Este enfoque unificado simplifica el software y la formación del personal.

3.2. Contrato: La Formalización de la Estancia

El `Contrato` es la entidad que hace de puente entre la fase conceptual (la reserva) y la fase física (la estancia). Su existencia representa el compromiso legal y operativo entre el hotel y el huésped.

- **Asignación de Habitación:** La relación entre `Contrato` y `Habitacion` es fundamental. Es en el momento de crear el contrato (típicamente durante el check-in) cuando se asigna una habitación física y única al cliente para toda su estancia. Este desacoplamiento permite que el personal de recepción pueda asignar la mejor habitación disponible que cumpla con el `TipoHabitacion` reservado, teniendo en cuenta las preferencias del cliente (piso alto, lejos del ascensor) o las necesidades operativas del hotel en ese preciso instante.
- **Registro Real de la Estancia:** Mientras que la `Reserva` contiene las fechas *planificadas*, el `Contrato` incluye los atributos `fechaCheckIn` y `fechaCheckOut`. Esta decisión de diseño permite registrar la hora y día exactos de la llegada y salida. Esto es crucial no solo para la facturación (posibles cargos por late check-out), sino también para la inteligencia de negocio (analizar horas pico de check-in, duración media real de las estancias, etc.).

3.3. Gestión de Pagos: Flexibilidad y Suposiciones

El modelo de pagos está diseñado para ser flexible, pero se basa en una suposición clave para simplificar la gestión.

- **Doble Vínculo del Pago:** La entidad `TipoPago` (y sus especializaciones `Efectivo` y `PagoTarjeta`) se relaciona de forma excluyente (`constraint`) tanto con `Reserva` como con `Contrato`. Esta decisión es deliberada y modela dos escenarios de negocio claros:
  - Si el pago se asocia a la **Reserva**, se considera un **prepago**. El cliente abona el importe total antes de su llegada para garantizar su reserva.
  - Si el pago se asocia al **Contrato**, significa que el cliente paga el importe en el momento del check-in.
  - La restricción de exclusividad (`XOR`) asegura que un mismo pago no puede estar asociado a ambos, manteniendo la integridad de los datos.
- **Suposición de Pago Único:** Para este modelo, se ha realizado la suposición de que **no se permiten pagos fraccionados**. Es decir, o se prepaga el 100% del importe de la reserva, o se paga el 100% en el hotel al formalizar el contrato. Un modelo más complejo que permitiese pagos parciales requeriría atributos adicionales como `saldoPendiente` en el contrato y posiblemente una entidad `Transaccion` asociada a cada `Pago`. Se ha optado por esta simplificación para mantener el modelo enfocado en el alcance definido.

3.4. Servicios y Consumos

El modelo permite añadir servicios adicionales a la estancia, con una lógica de pago diferenciada para mejorar la experiencia del cliente y simplificar las operaciones.

- **Asociación de Servicios:** Los servicios se asocian a nivel de `Pernoctacion` a través de la entidad `ServicioPernoctacion`. Esto permite un control granular. Por ejemplo, un cliente podría reservar un `Servicio` de 'Cuna para bebé' para todas las `Pernoctacion`es de su estancia, pero un `Servicio` de 'Cena romántica' para una sola noche.
- **Pago de Servicios:** Se distinguen dos escenarios operativos:
    1. **Servicios Pre-contratados:** Si el cliente contrata servicios al hacer la reserva, el coste de estos se incluye en el `montoTotal` del `Contrato` y se liquida junto con la estancia (ya sea en el prepago o en el hotel).
    2. **Servicios Durante la Estancia:** Se ha adoptado la regla de negocio de que **cualquier servicio solicitado durante la estancia se paga al momento de su consumo**. Esto implica que se generaría un nuevo `Pago` independiente. Esta decisión simplifica drásticamente el proceso de check-out, ya que no hay que añadir consumos de última hora, evitando sorpresas en la factura final del cliente y agilizando su salida.

3.5. Convenciones de Nomenclatura: Elección del Idioma

Una decisión transversal a todo el diseño ha sido la de utilizar el castellano para nombrar todas las entidades, atributos y relaciones del modelo. Esta elección, aunque simple, tiene un impacto significativo. El objetivo es alinear el modelo de datos lo más estrechamente posible con el lenguaje del negocio (*Domain-Driven Design*). Al utilizar términos como `Reserva`, `Pernoctacion` o `Contrato`, el modelo se vuelve inmediatamente comprensible para los *stakeholders* y los usuarios finales del sistema en un contexto de habla hispana. Esto reduce la ambigüedad, facilita la comunicación entre el equipo de desarrollo y los expertos del dominio, y simplifica el mantenimiento futuro del software, ya que el código y la base de datos hablarán el mismo idioma que los usuarios.

4. Entidades de Soporte y Parametrización

Más allá del flujo principal, el modelo incluye varias entidades que actúan como catálogos de configuración. Estas entidades permiten que el sistema sea flexible y adaptable sin necesidad de modificar su código.

- **`Hotel`, `Tarifa`, `Regimen`, `Descuento`, `Servicio`:** Estas entidades no registran transacciones, sino que definen la "oferta" del hotel. La dirección del hotel puede añadir nuevos servicios, crear tarifas para una nueva temporada o diseñar un nuevo tipo de régimen de alojamiento simplemente añadiendo registros a estas tablas.
- **Relaciones de Configuración y Lógica de Precios:** Las tablas asociativas como `Hotel_TipoHabitacion` y `Hotel_Tarifa` son cruciales. Permiten definir qué tipos de habitación ofrece cada hotel y, lo más importante, qué `Tarifa` única aplica. Según la regla de negocio, el precio es único por `Hotel`, `TipoHabitacion` y por `Temporada`. Por lo tanto, aunque no haya una entidad "Temporada" explícita en el diagrama, se asume que la entidad `Tarifa` representa el precio para una

combinación específica de estos tres elementos. La dirección del hotel puede definir fácilmente los precios para la 'Temporada Alta', 'Temporada Baja', etc., creando diferentes registros de `Tarifa`.

5. Conclusión

El modelo conceptual presentado resuelve de manera eficaz la problemática de la gestión de un negocio hotelero. Las decisiones de diseño, como la separación entre reserva conceptual y contrato físico, la gestión unificada de clientes walk-in, y las reglas claras para los pagos y servicios, dan como resultado un sistema coherente, lógico y que se adapta a los flujos de trabajo del mundo real.

Las fortalezas clave de este diseño son su **flexibilidad** para adaptarse a distintas operativas, su **fidelidad** al reflejar los procesos reales de la industria hotelera, y su **escalabilidad**. La sólida base conceptual permitiría incorporar fácilmente módulos futuros, como programas de fidelización de clientes, gestión avanzada de limpieza (housekeeping) o la integración con Channel Managers externos, confirmando que el diseño actual es una base robusta para el desarrollo de un PMS completo y competitivo.

DUMP SQL

```
-- ------------------------------------------------------
-- Script de Creación de Base de Datos para PMS
-- Versión: 2.1
-- Autor: Nasim Hosam Benyacoub Terki
-- Basado en el modelo conceptual actualizado.
-- ------------------------------------------------------

-- Se eliminan las tablas si ya existen para permitir una nueva
creación desde cero.
-- El orden es importante para evitar errores de claves foráneas.
DROP TABLE IF EXISTS Servicio_Pernoctacion;
DROP TABLE IF EXISTS Reserva_Descuento;
DROP TABLE IF EXISTS Reserva_Huespedes;
DROP TABLE IF EXISTS PagoEfectivo;
DROP TABLE IF EXISTS PagoTarjeta;
DROP TABLE IF EXISTS TipoPago;
DROP TABLE IF EXISTS Contrato;
DROP TABLE IF EXISTS Pernoctacion;
DROP TABLE IF EXISTS Reserva;
```

```sql
DROP TABLE IF EXISTS Hotel_Tarifa;
DROP TABLE IF EXISTS Hotel_TipoHabitacion;
DROP TABLE IF EXISTS Tarifa;
DROP TABLE IF EXISTS Descuento;
DROP TABLE IF EXISTS Servicio;
DROP TABLE IF EXISTS PrecioRegimen;
DROP TABLE IF EXISTS Regimen;
DROP TABLE IF EXISTS Cliente;
DROP TABLE IF EXISTS Habitacion;
DROP TABLE IF EXISTS TipoHabitacion;
DROP TABLE IF EXISTS Hotel;


-- ------------------------------------------------------
-- Entidades de Configuración y Catálogos
-- ------------------------------------------------------

-- Tabla para almacenar los hoteles
CREATE TABLE Hotel (
  idHotel INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  ubicacion VARCHAR(255) NOT NULL,
  categoria INT NOT NULL COMMENT 'Ej: 3, 4, 5 estrellas'
);

-- Tabla para los tipos de habitación (Doble, Individual, Suite, etc.)
CREATE TABLE TipoHabitacion (
  idTipoHabitacion INT AUTO_INCREMENT PRIMARY KEY,
  categoria VARCHAR(50) NOT NULL UNIQUE,
  camasIndividuales INT DEFAULT 0,
  camasDobles INT DEFAULT 0
);

-- Tabla para las habitaciones físicas del hotel
CREATE TABLE Habitacion (
  numeroHabitacion VARCHAR(10) PRIMARY KEY,
  idTipoHabitacion INT NOT NULL,
  idHotel INT NOT NULL,
  FOREIGN KEY (idTipoHabitacion) REFERENCES
TipoHabitacion(idTipoHabitacion) ON DELETE RESTRICT,
  FOREIGN KEY (idHotel) REFERENCES Hotel(idHotel) ON DELETE CASCADE
```

```
);

-- Tabla para los regímenes de alojamiento (Solo Alojamiento, Media
Pensión, etc.)
CREATE TABLE Regimen (
  idRegimen INT AUTO_INCREMENT PRIMARY KEY,
  codigo VARCHAR(10) NOT NULL UNIQUE
);

-- Tabla para almacenar los precios de los regímenes, asociados a un
hotel
CREATE TABLE PrecioRegimen (
    idPrecioRegimen INT AUTO_INCREMENT PRIMARY KEY,
    idRegimen INT NOT NULL,
    idHotel INT NOT NULL, -- MODIFICADO: Relación con Hotel añadida
    precio DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (idRegimen) REFERENCES Regimen(idRegimen) ON DELETE
CASCADE,
    FOREIGN KEY (idHotel) REFERENCES Hotel(idHotel) ON DELETE CASCADE
-- MODIFICADO
);

-- Tabla para los servicios adicionales que ofrece el hotel
CREATE TABLE Servicio (
  codigoServicio VARCHAR(10) PRIMARY KEY,
  Precio DECIMAL(10, 2) NOT NULL
);

-- Tabla para las tarifas de las habitaciones, que dependen de la
temporada/hotel/tipo
CREATE TABLE Tarifa (
  idTarifa INT AUTO_INCREMENT PRIMARY KEY,
  codigo VARCHAR(50) NOT NULL COMMENT 'Ej: TARIFA_ALTA_2025',
  precio DECIMAL(10, 2) NOT NULL
);

-- Tabla para los descuentos aplicables
CREATE TABLE Descuento (
  idDescuento INT AUTO_INCREMENT PRIMARY KEY,
  Descripcion VARCHAR(255) NOT NULL,
```

```
  Monto DECIMAL(10, 2) NOT NULL COMMENT 'Puede ser un monto fijo o un
porcentaje, la lógica se aplicaría en la aplicación.'
);


-- -----------------------------------------------------
-- Entidades Principales y de Transacción
-- -----------------------------------------------------

-- Tabla para los clientes
CREATE TABLE Cliente (
  idCliente INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  apellidos VARCHAR(150) NOT NULL,
  correoElectronico VARCHAR(255) NOT NULL UNIQUE,
  fechaDeNacimiento DATE,
  DNI VARCHAR(20) NOT NULL UNIQUE
);

-- Tabla para las reservas
CREATE TABLE Reserva (
  idReserva INT AUTO_INCREMENT PRIMARY KEY,
  fechaEntrada DATE NOT NULL,
  fechaSalida DATE NOT NULL,
  canalReserva VARCHAR(50),
  tipo ENUM('Reserva', 'Walkin') NOT NULL,
  idCliente_paga INT NOT NULL,
  idPrecioRegimen INT NOT NULL,
  FOREIGN KEY (idCliente_paga) REFERENCES Cliente(idCliente) ON DELETE
RESTRICT,
  FOREIGN KEY (idPrecioRegimen) REFERENCES
PrecioRegimen(idPrecioRegimen) ON DELETE RESTRICT
);

-- Tabla para las pernoctaciones (noches) de una reserva
CREATE TABLE Pernoctacion (
  idPernoctacion INT AUTO_INCREMENT PRIMARY KEY,
  fechaPernoctacion DATE NOT NULL,
  idReserva INT NOT NULL,
  idTipoHabitacion INT NOT NULL,
```

```
  FOREIGN KEY (idReserva) REFERENCES Reserva(idReserva) ON DELETE
CASCADE,
  FOREIGN KEY (idTipoHabitacion) REFERENCES
TipoHabitacion(idTipoHabitacion) ON DELETE RESTRICT
);

-- Tabla para los contratos, que materializan una reserva
CREATE TABLE Contrato (
  idContrato INT AUTO_INCREMENT PRIMARY KEY,
  montoTotal DECIMAL(10, 2) NOT NULL,
  fechaCheckIn DATETIME NULL COMMENT 'Nulo hasta que el cliente llega',
  fechaCheckOut DATETIME NULL COMMENT 'Nulo hasta que el cliente se
va',
  idReserva INT NOT NULL UNIQUE,
  numeroHabitacion VARCHAR(10) NOT NULL,
  FOREIGN KEY (idReserva) REFERENCES Reserva(idReserva) ON DELETE
RESTRICT,
  FOREIGN KEY (numeroHabitacion) REFERENCES
Habitacion(numeroHabitacion) ON DELETE RESTRICT
);

-- -----------------------------------------------------
-- Tablas de Pago (Estrategia: Tabla por Clase)
-- -----------------------------------------------------

-- Tabla padre para los pagos, con los atributos comunes
CREATE TABLE TipoPago (
  idTipoPago INT AUTO_INCREMENT PRIMARY KEY,
  montoPagado DECIMAL(10, 2) NOT NULL,
  idReserva INT NULL COMMENT 'Se rellena si es un prepago',
  idContrato INT NULL COMMENT 'Se rellena si el pago es en el hotel',
  FOREIGN KEY (idReserva) REFERENCES Reserva(idReserva) ON DELETE SET
NULL,
  FOREIGN KEY (idContrato) REFERENCES Contrato(idContrato) ON DELETE
SET NULL,
  CONSTRAINT chk_pago_asociacion_exclusiva CHECK ((idReserva IS NOT
NULL AND idContrato IS NULL) OR (idReserva IS NULL AND idContrato IS
NOT NULL))
);

-- Tabla hija para pagos en efectivo
```

```sql
CREATE TABLE PagoEfectivo (
  idTipoPago INT PRIMARY KEY,
  cambio DECIMAL(10, 2) DEFAULT 0.00,
  FOREIGN KEY (idTipoPago) REFERENCES TipoPago(idTipoPago) ON DELETE
CASCADE
);

-- Tabla hija para pagos con tarjeta
CREATE TABLE PagoTarjeta (
  idTipoPago INT PRIMARY KEY,
  numeroTarjeta VARCHAR(20) NOT NULL,
  fechaExpiracion DATE NOT NULL,
  nombre_titular VARCHAR(150) NOT NULL,
  idCliente INT NOT NULL COMMENT 'Cliente titular de la tarjeta',
  FOREIGN KEY (idTipoPago) REFERENCES TipoPago(idTipoPago) ON DELETE
CASCADE,
  FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente) ON DELETE
RESTRICT
);


-- -----------------------------------------------------
-- Tablas Asociativas (Relaciones N:M)
-- -----------------------------------------------------

-- Asocia qué tipos de habitación tiene cada hotel
CREATE TABLE Hotel_TipoHabitacion (
  idHotel INT NOT NULL,
  idTipoHabitacion INT NOT NULL,
  PRIMARY KEY (idHotel, idTipoHabitacion),
  FOREIGN KEY (idHotel) REFERENCES Hotel(idHotel) ON DELETE CASCADE,
  FOREIGN KEY (idTipoHabitacion) REFERENCES
TipoHabitacion(idTipoHabitacion) ON DELETE CASCADE
);

-- Asocia las tarifas a un hotel y tipo de habitación (asumiendo
temporada implícita en Tarifa)
CREATE TABLE Hotel_Tarifa (
  idHotel INT NOT NULL,
  idTarifa INT NOT NULL,
  idTipoHabitacion INT NOT NULL,
```

```
  PRIMARY KEY (idHotel, idTarifa, idTipoHabitacion),
  FOREIGN KEY (idHotel) REFERENCES Hotel(idHotel) ON DELETE CASCADE,
  FOREIGN KEY (idTarifa) REFERENCES Tarifa(idTarifa) ON DELETE CASCADE,
  FOREIGN KEY (idTipoHabitacion) REFERENCES
TipoHabitacion(idTipoHabitacion) ON DELETE CASCADE
);

-- Asocia qué clientes se hospedan en una reserva (además del pagador)
CREATE TABLE Reserva_Huespedes (
  idReserva INT NOT NULL,
  idCliente INT NOT NULL,
  PRIMARY KEY (idReserva, idCliente),
  FOREIGN KEY (idReserva) REFERENCES Reserva(idReserva) ON DELETE
CASCADE,
  FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente) ON DELETE
CASCADE
);

-- Asocia descuentos a una reserva
CREATE TABLE Reserva_Descuento (
  idReserva INT NOT NULL,
  idDescuento INT NOT NULL,
  PRIMARY KEY (idReserva, idDescuento),
  FOREIGN KEY (idReserva) REFERENCES Reserva(idReserva) ON DELETE
CASCADE,
  FOREIGN KEY (idDescuento) REFERENCES Descuento(idDescuento) ON DELETE
RESTRICT
);

-- Asocia servicios a una pernoctación específica
CREATE TABLE Servicio_Pernoctacion (
  idPernoctacion INT NOT NULL,
  codigoServicio VARCHAR(10) NOT NULL,
  PRIMARY KEY (idPernoctacion, codigoServicio),
  FOREIGN KEY (idPernoctacion) REFERENCES Pernoctacion(idPernoctacion)
ON DELETE CASCADE,
  FOREIGN KEY (codigoServicio) REFERENCES Servicio(codigoServicio) ON
DELETE CASCADE
);
```


UIB