Introducción a los Diagramas Entidad–Relación (ER) con la base de datos *Sakila*

Departamento de Informática

1. Objetivo del documento

Este texto introduce los conceptos fundamentales del modelado Entidad–Relación (ER) usando como caso real la base de datos **Sakila** (videoclub).

Introducción a la base de datos *Sakila*

La base de datos **Sakila** es un modelo de ejemplo desarrollado por **MySQL** con fines educativos. Su propósito es mostrar cómo se estructura y gestiona un sistema de información realista mediante un modelo entidad–relación completo. El dominio elegido es el de un **videoclub internacional** que ofrece servicios de **alquiler de películas** en varias tiendas distribuidas por diferentes países.

El objetivo de Sakila no es servir como una base de datos de producción, sino como una herramienta de aprendizaje y demostración. Permite estudiar cómo se definen las **entidades**, las **relaciones**, las **claves primarias y foráneas**, y cómo se representan los distintos tipos de cardinalidad (1:1, 1:N, N:M). También resulta idónea para practicar consultas SQL, la normalización, el diseño físico y la integridad referencial.

1. Contexto del modelo

Sakila reproduce de forma simplificada la operativa de un videoclub: los clientes alquilan películas, los empleados gestionan los pagos y las devoluciones, y cada tienda mantiene su propio inventario. Las películas están descritas con información detallada —título, año, idioma, duración, clasificación, actores y categorías— y se distribuyen entre distintas tiendas ubicadas en diversas ciudades y países.

El modelo está completamente normalizado e incluye los principales elementos que intervienen en el negocio:

- **Clientes** y sus datos de contacto.

- **Empleados** y **tiendas** físicas.
- **Películas**, **idiomas**, **categorías** y **actores**.
- **Inventario**, **alquileres** y **pagos**.

2. Objetivo pedagógico

Esta base de datos es ideal para introducir el **modelo Entidad–Relación (ER)** porque contiene todos los casos que un diseñador debe dominar:

- Entidades fuertes y débiles.
- Relaciones de los tres tipos posibles: 1:1, 1:N y N:M.
- Claves primarias simples y compuestas.
- Tablas intermedias con atributos propios.

Gracias a esta riqueza estructural, Sakila permite observar cómo un modelo conceptual se traduce al modelo relacional y cómo las relaciones se implementan mediante claves foráneas.

3. Estructura general

La base de datos contiene aproximadamente dieciséis tablas, organizadas en tres bloques principales:

1. **Bloque de Catálogo:** contiene la información relacionada con las películas y su clasificación. Incluye tablas como: `film`, `actor`, `category`, `language`, `film_actor`, `film_category`.
2. **Bloque de Operaciones del videoclub:** representa la actividad diaria del negocio, es decir, los alquileres, los pagos y la gestión de clientes y empleados. Incluye tablas como: `inventory`, `rental`, `payment`, `store`, `staff`, `customer`.
3. **Bloque de Ubicación geográfica:** almacena la información sobre direcciones, ciudades y países, que se utiliza para vincular tiendas, clientes y empleados con sus ubicaciones. Incluye tablas como: `address`, `city`, `country`.

Cada bloque está conectado con los demás mediante **claves foráneas**, formando una red de relaciones que permite representar la totalidad de la información necesaria para la gestión de un videoclub.

2. Conceptos básicos

2.1 Entidad

Una **entidad** es un tipo de objeto del mundo real sobre el que queremos guardar información. En Sakila, ejemplos de entidades son: `film` (película), `actor`, `customer` (cliente), `store` (tienda), `rental` (alquiler), `payment` (pago), `address` (dirección), `city`, `country`, `language`.

2.2 Atributo

Un **atributo** es un dato que describe a la entidad. Por ejemplo, en `film`: `title` (título), `length` (duración), `release_year` (año), `rating`; en `actor`: `first_name`, `last_name`.

2.3 Clave primaria (PK)

La **clave primaria** identifica *de manera única* cada fila de una tabla. No puede repetirse ni ser nula. Ejemplos: `film(film_id)`, `actor(actor_id)`, `customer(customer_id)`.

2.4 Clave foránea (FK)

Una **clave foránea** es un atributo en una tabla que referencia la clave primaria de otra tabla, estableciendo una relación entre ambas. Ejemplos:

- `rental.customer_id` → `customer(customer_id)`.
- `inventory.film_id` → `film(film_id)`.

3. Entidades fuertes y débiles

- **Fuertes** (existen por sí mismas): `film`, `actor`, `category`, `customer`, `staff`, `store`, `rental`, `payment`, `address`, `city`, `country`, `language`.
- **Débiles / puente** (dependen de otras): `film_actor`, `film_category`. En Sakila, `inventory` depende lógicamente de `film` y `store` (es una copia física de una película en una tienda), aunque tenga PK propia.

4. Tipos de relaciones (cardinalidades)

4.1 Relación 1:N (uno a muchos)

Un registro de A se asocia con muchos de B, pero cada B con exactamente uno de A. Ejemplos:

- `country (1)` → `city (N)`.
- `city (1)` → `address (N)`.
- `store (1)` → `staff (N)`.
- `customer (1)` → `rental (N)`.
- `rental (1)` → `payment (N)`.
- `film (1)` → `inventory (N)`.

4.2 Relación 1:1 (uno a uno)

Cada registro de A se asocia con *exactamente uno* de B, y viceversa. En Sakila, el ejemplo didáctico es:

- `store` ↔ `staff` (manager): cada tienda tiene un único gerente (referenciado por `store.manager_staff_id`); para imponer realmente 1:1 se requiere una **restricción UNIQUE** sobre `store.manager_staff_id` (y coherencia con `staff.store_id`).

Nota: Muchos escenarios 1:1 se modelan en la práctica como 1:N con una restricción `UNIQUE` en la FK del lado N.

4.3 Relación N:M (muchos a muchos)

Si una película puede tener *muchos* actores y un actor puede participar en *muchas* películas, no se puede guardar correctamente con una sola FK. Piénsalo por un momento y convéncete de ello.

Necesidad de tabla intermedia:

a) Sin tabla intermedia, sólo podríamos registrar "un actor por película" (FK en `film`) o "una película por actor" (FK en `actor`), lo cual es incorrecto.

b) La tabla intermedia (`film_actor`) contiene **dos FKs**, una a cada entidad principal, y convierte la N:M en **dos relaciones 1:N** (`film` → `film_actor` y `actor` → `film_actor`).

c) Para **evitar duplicados** (el mismo actor repetido para la misma película), la PK suele ser **compuesta** por ambas FKs:

```
PRIMARY KEY (film_id, actor_id)
```

Atributos en tablas intermedias (muy importante). Las tablas intermedias *no sólo* guardan las FKs. **Pueden y suelen tener atributos propios**, por ejemplo, la tabla `film_actor` representa la tabla intermedia entre `film` y `actor` y tiene estos atributos:

- `film_actor.rol` (*principal, secundario, cameo*),
- `film_actor.orden_facturacion` (posición en los créditos),
- `film_actor.fecha_contrato` (cuándo se incorporó),
- `film_actor.cache` (importe pactado),
- `last_update` (técnico; Sakila lo incluye a menudo).

En Sakila, `film_actor` y `film_category` ya incluyen un atributo técnico `last_update`. En un sistema real añadiríamos los campos de negocio que correspondan a la relación.

Resumen de N:M en Sakila.

- `film` $\leftrightarrow$ `actor` $\Rightarrow$ `film_actor(film_id, actor_id, last_update, ...)`
- `film` $\leftrightarrow$ `category` $\Rightarrow$ `film_category(film_id, category_id, last_update, ...)`

5. Patrón de mapeo ER $\rightarrow$ relacional

1. **Relación 1:N** $\rightarrow$ FK en la tabla del lado N que hace referencia a la PK en la tabla del 1.
2. **Relación 1:1** $\rightarrow$ FK con `UNIQUE` en el lado que almacena la referencia (para "forzar" 1:1).
3. **Relación N:M** $\rightarrow$ **tabla intermedia** con FKs a ambas tablas y **PK compuesta** (o PK artificial + `UNIQUE(fk1,fk2)`).
4. Añadir **atributos de la relación** (si existen) a la **tabla intermedia**.

6. Sakila: mini–mapa de algunas relaciones

```
country (1) -> city (N) -> address (N) -> customer (N)
                                    \
                                     -> staff (N)
store (1) -> customer (N)
store (1) -> inventory (N) -> rental (N) -> payment (N)
```

```
film (1) -> inventory (N)
film (N) -> film_actor <- (N) actor
film (N) -> film_category <- (N) category
```

7. Tarea práctica (para entregar)

Actividad: Genera el ER en MySQL Workbench a partir del caso Sakila y documenta tus decisiones.

7.1 Pasos en MySQL Workbench

1. Abre MySQL Workbench.
2. "Database" -> "Reverse Engineer"
3. Selecciona solamente las tablas.

7.2 Entregables

- Diagrama EER (captura en PDF o PNG) donde se vean claramente las tablas y las relaciones.
- Documento breve (1–2 páginas) respondiendo:
    1. ¿Cuáles son las **claves primarias** de cada tabla?
    2. ¿Qué **claves foráneas** existen y qué relación representan (1:1, 1:N, N:M)?
    3. En las N:M, ¿qué **atributos adicionales** tienen las **tablas intermedias** y por qué?
    4. ¿Cómo se garantiza que no haya **duplicados** en la tabla intermedia?
    5. ¿Recuerdas la base de datos tienda_online? ¿Qué relaciones N:M y 1:N tenía?

Recordatorio clave sobre N:M: la tabla intermedia es el *lugar correcto* para poner los datos que describen la *relación* (no a los extremos). Ejemplo: en `film_actor`, el *rol* o el *orden de facturación* describen "la participación de ese actor en esa película", no al actor ni a la película por separado.