1. Esquema lógico de la BB.DD SAKILA

El esquema lógico de la base de datos **Sakila** de MySQL está diseñado para modelar un sistema de gestión de alquiler de películas. Es una base de datos de ejemplo ampliamente utilizada para aprender y practicar conceptos de SQL. La base de datos sigue una estructura normalizada (3FN) y tiene tablas que representan entidades principales y sus relaciones.

A continuación, se describe el esquema lógico:

---

1. Entidades principales y su propósito

Tabla	Propósito
`film`	Almacena detalles de las películas.
`actor`	Lista de actores que aparecen en las películas.
`film_actor`	Relaciona actores con las películas en las que participaron (N:M).
`category`	Categorías de las películas (por ejemplo, acción, comedia).
`film_category`	Relaciona las películas con sus categorías (N:M).
`customer`	Información sobre los clientes que alquilan películas.
`store`	Información de las tiendas físicas donde se realizan los alquileres.
`staff`	Información sobre el personal que gestiona las tiendas y alquileres.
`inventory`	Representa las copias físicas de las películas disponibles en una tienda.
`rental`	Registro de los alquileres realizados por los clientes.
`payment`	Información de los pagos realizados por los alquileres.
`address`	Detalles de las direcciones asociadas a clientes, tiendas y empleados.
`city`	Ciudades relacionadas con las direcciones.
`country`	Países relacionados con las ciudades.
`language`	Idiomas disponibles para las películas.

---

2. Relaciones entre las tablas

El esquema lógico de la base de datos Sakila está compuesto por relaciones entre entidades. Aquí se destacan las más importantes:

Tablas relacionadas con películas

- `film`:
  - Relación **1:N** con `inventory` (una película puede tener múltiples copias en diferentes tiendas).
  - Relación **N:M** con `actor` a través de `film_actor`.
  - Relación **N:M** con `category` a través de `film_category`.
  - Relación **1:N** con `language` (un idioma principal por película).

Tablas relacionadas con clientes

- `customer`:
  - Relación **1:N** con `rental` (un cliente puede realizar múltiples alquileres).
  - Relación **1:N** con `payment` (un cliente puede realizar múltiples pagos).
  - Relación **N:1** con `store` (cada cliente está asociado a una tienda).

Tablas relacionadas con alquileres

- `rental`:
  - Relación **N:1** con `inventory` (un alquiler se asocia a una copia específica de una película).
  - Relación **N:1** con `customer` (un cliente realiza el alquiler).
  - Relación **N:1** con `staff` (un empleado gestiona el alquiler).

Tablas relacionadas con pagos

- `payment`:
  - Relación **N:1** con `rental` (un pago está asociado a un alquiler).
  - Relación **N:1** con `customer` (un cliente realiza el pago).
  - Relación **N:1** con `staff` (un empleado gestiona el pago).

Tablas relacionadas con localización

- `address`:
  - Relación **N:1** con `city` (cada dirección pertenece a una ciudad).
- `city`:
  - Relación **N:1** con `country` (cada ciudad pertenece a un país).

---

3. Esquema lógico en forma de diagrama textual

Resumen de las claves primarias (PK) y claves foráneas (FK) para comprender mejor las relaciones:

Tablas principales:

- `actor`
  - PK: `actor_id`
- `film`
  - PK: `film_id`
  - FK: `language_id` → `language.language_id`
- `customer`
  - PK: `customer_id`
  - FK: `store_id` → `store.store_id`
  - FK: `address_id` → `address.address_id`
- `store`

    - PK: `store_id`
    - FK: `manager_staff_id` → `staff.staff_id`
    - FK: `address_id` → `address.address_id`
- `rental`
    - PK: `rental_id`
    - FK: `inventory_id` → `inventory.inventory_id`
    - FK: `customer_id` → `customer.customer_id`
    - FK: `staff_id` → `staff.staff_id`

Tablas de relación:

- `film_actor`
    - PK: `actor_id`, `film_id`
    - FK: `actor_id` → `actor.actor_id`
    - FK: `film_id` → `film.film_id`
- `film_category`
    - PK: `film_id`, `category_id`
    - FK: `film_id` → `film.film_id`
    - FK: `category_id` → `category.category_id`

---

4. Características clave del diseño

1. Normalización:
    - El diseño está altamente normalizado (3FN), lo que reduce la redundancia y asegura consistencia.
2. Relaciones N:M:
    - Se modelan con tablas intermedias como `film_actor` y `film_category`.
3. Optimización para consultas transaccionales:
    - Aunque es normalizado, está diseñado para transacciones rápidas y eficientes.

---

2. Determinar la tabla de hechos

Determinar la **tabla de hechos** al desnormalizar la base de datos Sakila implica identificar la entidad central que captura eventos o transacciones principales en el sistema. En un modelo en estrella, las tablas de hechos suelen contener datos cuantitativos o métricas que se pueden analizar, mientras que las tablas de dimensiones describen el contexto de esos hechos.

---

1. Pasos para determinar la tabla de hechos

a. Identificar los eventos principales

En la base de datos Sakila, hay dos eventos principales que pueden ser considerados tablas de hechos:

1. **Rentas de películas** (`rental`): Representa cada transacción en la que un cliente alquila una película.
2. **Pagos realizados** (`payment`): Representa cada transacción de pago asociada a un alquiler.

Ambos eventos contienen datos cuantitativos:

- **`rental`:** Fechas de alquiler y devolución, películas alquiladas, cliente, empleado, tienda.
- **`payment`:** Monto pagado, cliente, alquiler relacionado.

b. Decidir el nivel de granularidad

La granularidad de la tabla de hechos debe reflejar la menor unidad de análisis que deseas:

- Si te interesa analizar cada alquiler individual, la tabla `rental` sería tu tabla de hechos.
- Si el análisis se enfoca en pagos individuales realizados por los clientes, la tabla `payment` será más apropiada.

En muchos casos, puedes combinar ambos niveles, creando una tabla de hechos unificada que incluya datos tanto de `rental` como de `payment`.

---

2. Propuesta de tabla de hechos

Tabla de hechos: `rental_payment_fact`

Una tabla de hechos unificada que combine `rental` y `payment` podría incluir lo siguiente:

Columna	Descripción
`rental_id`	Identificador único del alquiler.
`payment_id`	Identificador único del pago.
`rental_date`	Fecha y hora del alquiler.
`return_date`	Fecha y hora de devolución.
`film_id`	Identificador de la película alquilada.
`customer_id`	Identificador del cliente que realizó el alquiler.
`staff_id`	Empleado que gestionó el alquiler.
`store_id`	Tienda donde se realizó el alquiler.
`payment_date`	Fecha y hora del pago.
`amount`	Monto del pago.

SQL para construir la tabla de hechos:

```sql
CREATE TABLE rental_payment_fact AS
SELECT
    r.rental_id,
    p.payment_id,
    r.rental_date,
    r.return_date,
    i.film_id,
    r.customer_id,
    r.staff_id,
    s.store_id,
    p.payment_date,
    p.amount
FROM
    rental r
LEFT JOIN payment p ON r.rental_id = p.rental_id
LEFT JOIN inventory i ON r.inventory_id = i.inventory_id
LEFT JOIN store s ON i.store_id = s.store_id;
```

---

3. Determinación de las dimensiones

La tabla de hechos está relacionada con varias dimensiones que proporcionan el contexto necesario para análisis:

1. **Dimensión Cliente (`customer_dim`)**:
    - Contiene detalles del cliente (`customer_id`, nombre, email, tienda asociada).
2. **Dimensión Película (`film_dim`)**:
    - Detalles de las películas (`film_id`, título, categoría, actores).
3. **Dimensión Tiempo (`time_dim`)**:
    - Fechas específicas de análisis (`rental_date`, `return_date`, `payment_date`).
4. **Dimensión Tienda (`store_dim`)**:
    - Información sobre las tiendas (`store_id`, ubicación, gerente).
5. **Dimensión Empleado (`staff_dim`)**:
    - Detalles del personal (`staff_id`, nombre, correo electrónico).

---

4. Resumen

La tabla de hechos debe reflejar el evento principal que deseas analizar, y en el caso de Sakila, puede ser:

- **`rental`** (si te interesa analizar los alquileres).
- **`payment`** (si te interesa analizar los pagos).
- Una **combinación de ambas** (si buscas un análisis completo).

El diseño final dependerá de tus requisitos analíticos y las métricas clave que deseas calcular.

3. Pasos para desnormalizar la BB.DD SAKILA

La desnormalización es un proceso en el cual se combinan tablas previamente normalizadas para optimizar el rendimiento en consultas analíticas. Esto implica crear redundancia controlada al juntar datos relacionados en una sola tabla, con el objetivo de facilitar la construcción de un modelo en estrella, que es común en sistemas de inteligencia empresarial (BI).

En el modelo en estrella, tenemos una **tabla central o de hechos** que almacena las métricas (hechos numéricos como ventas, ingresos, etc.) y **tablas de dimensiones** que describen los contextos de esos hechos (como clientes, productos, tiempo, etc.).

Usaremos la base de datos **Sakila** como ejemplo para construir un modelo en estrella desnormalizado. La base Sakila es una base de datos de ejemplo que modela un sistema de alquiler de películas.

---

1. Identificar las tablas en Sakila

Las principales tablas en Sakila son:

- **Tablas de hechos:**
  - `rental` (información de los alquileres de películas).
  - `payment` (información de los pagos realizados).
- **Tablas de dimensiones:**
  - `film` (detalles de las películas).
  - `customer` (información de los clientes).
  - `staff` (empleados del sistema).
  - `store` (tiendas).
  - `category` (categorías de películas).
  - `actor` (actores de las películas).

2. Identificar las relaciones

La base de datos Sakila está completamente normalizada. Por ejemplo:

- **`rental`** está relacionada con:
  - `customer` (quién alquiló).
  - `inventory` (qué inventario fue alquilado, relacionado con una película específica).
  - `staff` (quién gestionó el alquiler).
- **`payment`** está relacionada con:
  - `rental` (el alquiler asociado al pago).
  - `customer` (quién pagó).
  - `staff` (quién gestionó el pago).
- **`film`** está relacionada con:
  - `category` (a través de `film_category`).

- `actor` (a través de `film_actor`).

3. Construcción del modelo en estrella

Para desnormalizar y construir el modelo en estrella, necesitamos:

1. **Tabla de hechos**:
   - Una combinación de datos de las tablas `rental` y `payment`. Esta tabla contendrá métricas como el monto pagado, la fecha de alquiler y otros datos importantes.
2. **Tablas de dimensiones**:
   - **Dimensión Cliente:** Combinar la tabla `customer` con datos relacionados como la tienda donde el cliente está registrado (`store`).
   - **Dimensión Película:** Combinar la tabla `film` con sus actores (`film_actor`) y categorías (`film_category`).
   - **Dimensión Tiempo:** Crear una tabla de fechas basada en las fechas en `rental_date` o `payment_date`.

Ejemplo de desnormalización

Tabla de hechos: rental_payment_fact

```sql
CREATE TABLE rental_payment_fact AS
SELECT
    r.rental_id,
    p.payment_id,
    r.rental_date,
    r.return_date,
    p.payment_date,
    p.amount AS payment_amount,
    r.customer_id,
    i.film_id,
    r.staff_id,
    c.store_id
FROM
    rental r
JOIN payment p ON r.rental_id = p.rental_id
JOIN customer c ON r.customer_id = c.customer_id
JOIN inventory i ON r.inventory_id = i.inventory_id;
```

Dimensión Cliente: customer_dim

```sql
CREATE TABLE customer_dim AS
SELECT
    c.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    c.email,
    c.active,
    c.create_date,
    c.store_id,
    s.manager_staff_id AS store_manager_id
FROM
    customer c
JOIN store s ON c.store_id = s.store_id;
```

Dimensión Película: film_dim

```sql
CREATE TABLE film_dim AS
SELECT
    f.film_id,
    f.title,
    f.description,
    f.release_year,
    f.rental_rate,
    f.length AS film_length,
    GROUP_CONCAT(DISTINCT a.actor_id) AS actor_ids,
    GROUP_CONCAT(DISTINCT a.first_name, ' ', a.last_name) AS actor_names,
    GROUP_CONCAT(DISTINCT c.category_id) AS category_ids,
    GROUP_CONCAT(DISTINCT cat.name) AS category_names
FROM
    film f
LEFT JOIN film_actor fa ON f.film_id = fa.film_id
LEFT JOIN actor a ON fa.actor_id = a.actor_id
LEFT JOIN film_category fc ON f.film_id = fc.film_id
LEFT JOIN category cat ON fc.category_id = cat.category_id
GROUP BY
    f.film_id;
```

Dimensión Tiempo: time_dim

```sql
CREATE TABLE time_dim AS
SELECT DISTINCT
    DATE(r.rental_date) AS date,
    EXTRACT(YEAR FROM r.rental_date) AS year,
    EXTRACT(MONTH FROM r.rental_date) AS month,
    EXTRACT(DAY FROM r.rental_date) AS day,
    EXTRACT(DAYOFWEEK FROM r.rental_date) AS day_of_week
FROM
    rental r;
```

Consideraciones

1. **Performance:** La desnormalización puede mejorar el rendimiento de consultas analíticas, pero aumentará el almacenamiento necesario.
2. **Consistencia:** Asegúrate de sincronizar correctamente las tablas desnormalizadas si los datos cambian.
3. **Herramientas de BI:** Al usar herramientas como Tableau o Power BI, un modelo en estrella desnormalizado simplifica mucho la creación de informes.

---

4. Por qué esta TABLA DE HECHOS

Incluir múltiples eventos en una tabla de hechos, como los **alquileres** (`rental`) y los **pagos** (`payment`) en el caso de la base de datos **Sakila**, tiene sentido en algunos contextos analíticos. Esto se debe a que ambos eventos están estrechamente relacionados y compartir una tabla de hechos unificada puede simplificar el modelo y las consultas analíticas. Aquí están las razones clave:

---

1. Relación Natural entre los Eventos

- En Sakila, cada pago (`payment`) está asociado a un alquiler (`rental`).
    - Un cliente alquila una película (evento de alquiler).
    - El cliente realiza un pago por el alquiler (evento de pago).

Esta relación directa permite que los dos eventos se combinen en una sola tabla de hechos, ya que ambos se refieren al mismo proceso de negocio.

---

2. Simplificación del Modelo

- Una tabla de hechos unificada evita duplicación de datos y simplifica las consultas, ya que las métricas clave (como el monto pagado) y las dimensiones relacionadas (como cliente, película, tienda, etc.) pueden estar disponibles en un solo lugar.
- Por ejemplo, en lugar de consultar dos tablas separadas para obtener detalles de un alquiler y su pago asociado, puedes consultar una única tabla.

---

3. Granularidad Consistente

- Una tabla de hechos unificada tiene una granularidad consistente: **un registro por cada pago realizado**.
    - Si un cliente realiza múltiples pagos por diferentes alquileres, habrá múltiples registros en la tabla de hechos, cada uno representando un par único de alquiler-pago.

Esto asegura que las métricas sean fáciles de agregar y analizar.

---

4. Flexibilidad para Análisis

Combinar ambos eventos en una tabla de hechos permite realizar diferentes tipos de análisis desde una sola fuente de datos:

- **Análisis de ingresos:** Sumar los pagos (`amount`) por cliente, tienda, película, etc.
- **Análisis de rendimiento de alquileres:** Contar cuántos alquileres se realizaron en un periodo determinado.
- **Análisis combinados:** Relacionar métricas de pago con detalles del alquiler, como qué tipo de películas generan más ingresos.

---

5. Reducción de Consultas Complejas

Si se usan tablas de hechos separadas, los análisis que involucran tanto datos de alquileres como de pagos requieren combinaciones (`JOINs`) entre las tablas de hechos. Esto puede aumentar la complejidad y el tiempo de ejecución de las consultas.

Al unificar los eventos, se evita esta necesidad, ya que la información relevante está disponible en una sola tabla.

Ejemplo: Tabla de Hechos Combinada

Supongamos que un cliente alquila tres películas y realiza un único pago por todas ellas:

Alquileres en `rental`

rental_id	customer_id	film_id	rental_date	return_date
1	101	201	2025-01-01 10:00:00	2025-01-03 10:00:00
2	101	202	2025-01-01 10:00:00	2025-01-03 10:00:00
3	101	203	2025-01-01 10:00:00	2025-01-03 10:00:00

Pago en `payment`

payment_id	rental_id	customer_id	payment_date	amount
501	1	101	2025-01-01 10:30:00	10.00

Tabla de Hechos Combinada

rental_id	payment_id	customer_id	film_id	rental_date	return_date	payment_date	amount
1	501	101	201	2025-01-01 10:00:00	2025-01-03 10:00:00	2025-01-01 10:30:00	10.00
2	501	101	202	2025-01-01 10:00:00	2025-01-03 10:00:00	2025-01-01 10:30:00	10.00
3	501	101	203	2025-01-01 10:00:00	2025-01-03 10:00:00	2025-01-01 10:30:00	10.00

---

6. Consideraciones al Combinar Eventos

- **Espacio de almacenamiento:** Una tabla de hechos combinada puede ser más grande, ya que duplica datos relacionados con los pagos para cada alquiler.
- **Consistencia:** Asegúrate de que la relación entre los eventos sea clara y consistente para evitar duplicados o datos incorrectos.
- **Granularidad adecuada:** Define el nivel de detalle necesario para el análisis y construye la tabla de hechos en consecuencia.

En resumen, combinar los eventos **alquiler** y **pago** en la tabla de hechos es una decisión práctica que simplifica el análisis, asegura consistencia y reduce la complejidad del modelo y las consultas.