Modelado de Bases de Datos para un Sistema Bancario

Principios esenciales y arquitectura de la información para sistemas financieros robustos.

Agenda: Pilares del Diseño de BBDD Bancarias

1. Entidades Fundamentales

Clientes, cuentas y transacciones: la base relacional.

2. Gestión de Transacciones

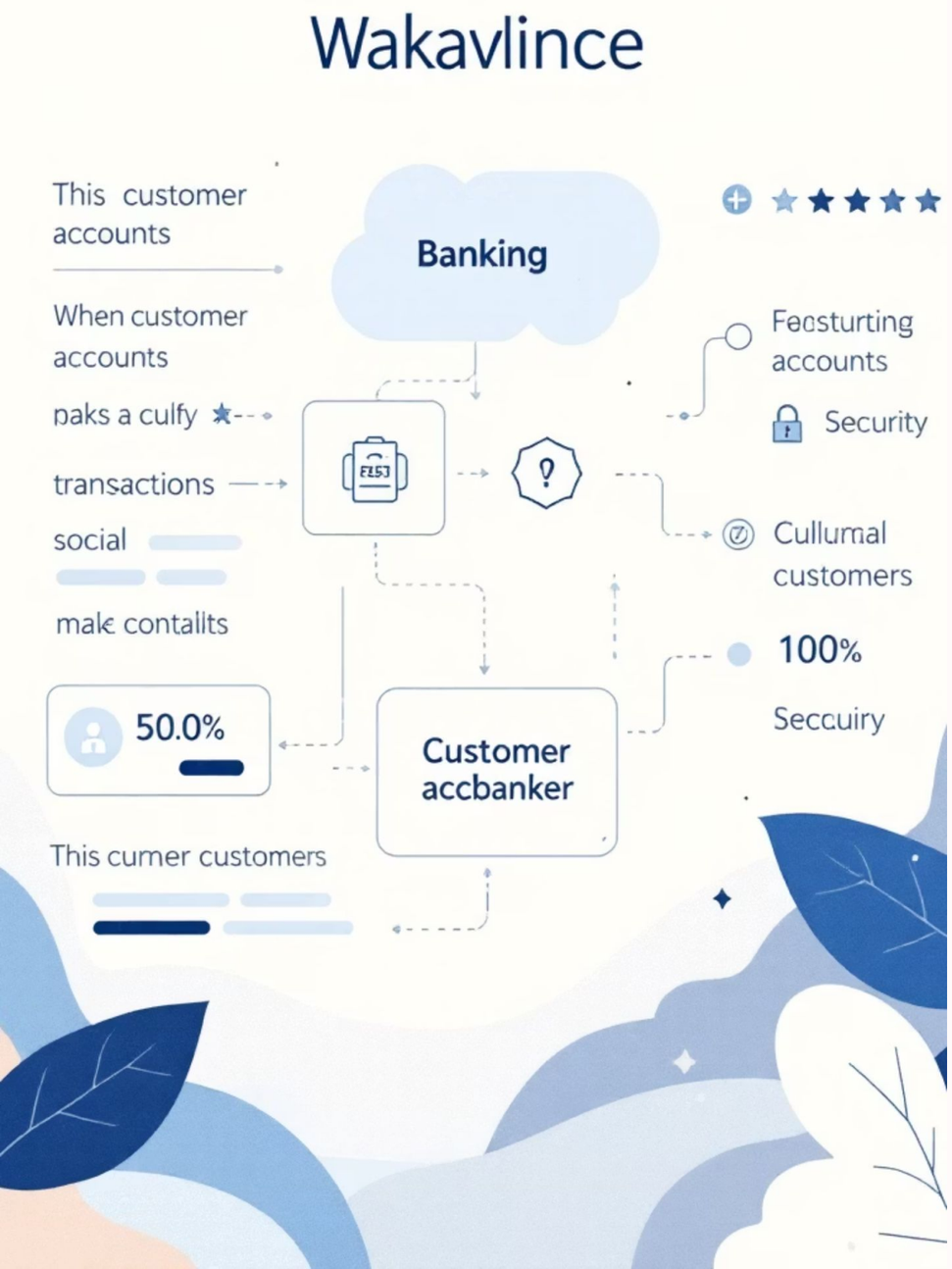
Modelado de movimientos financieros y el principio ACID.

3. Seguridad y Auditoría

Trazabilidad, logs y cumplimiento normativo (GDPR, PCI DSS).

4. Arquitectura de Alto Rendimiento

Consideraciones sobre escalabilidad, particionamiento y consistencia.



Capítulo 1

Las Entidades Fundamentales

El corazón de cualquier sistema bancario reside en la correcta definición de sus entidades principales y la relación entre ellas.

Modelando la Relación Cliente-Cuenta

La base de datos debe reflejar con precisión cómo se relacionan los individuos (clientes) con los vehículos financieros (cuentas). Esta es a menudo una relación de muchos a muchos (M:N) resuelta con una tabla intermedia.

1

Entidad Cliente

Almacena datos personales sensibles (DNI/NIE, dirección, contacto). **Clave: Seguridad.**

2

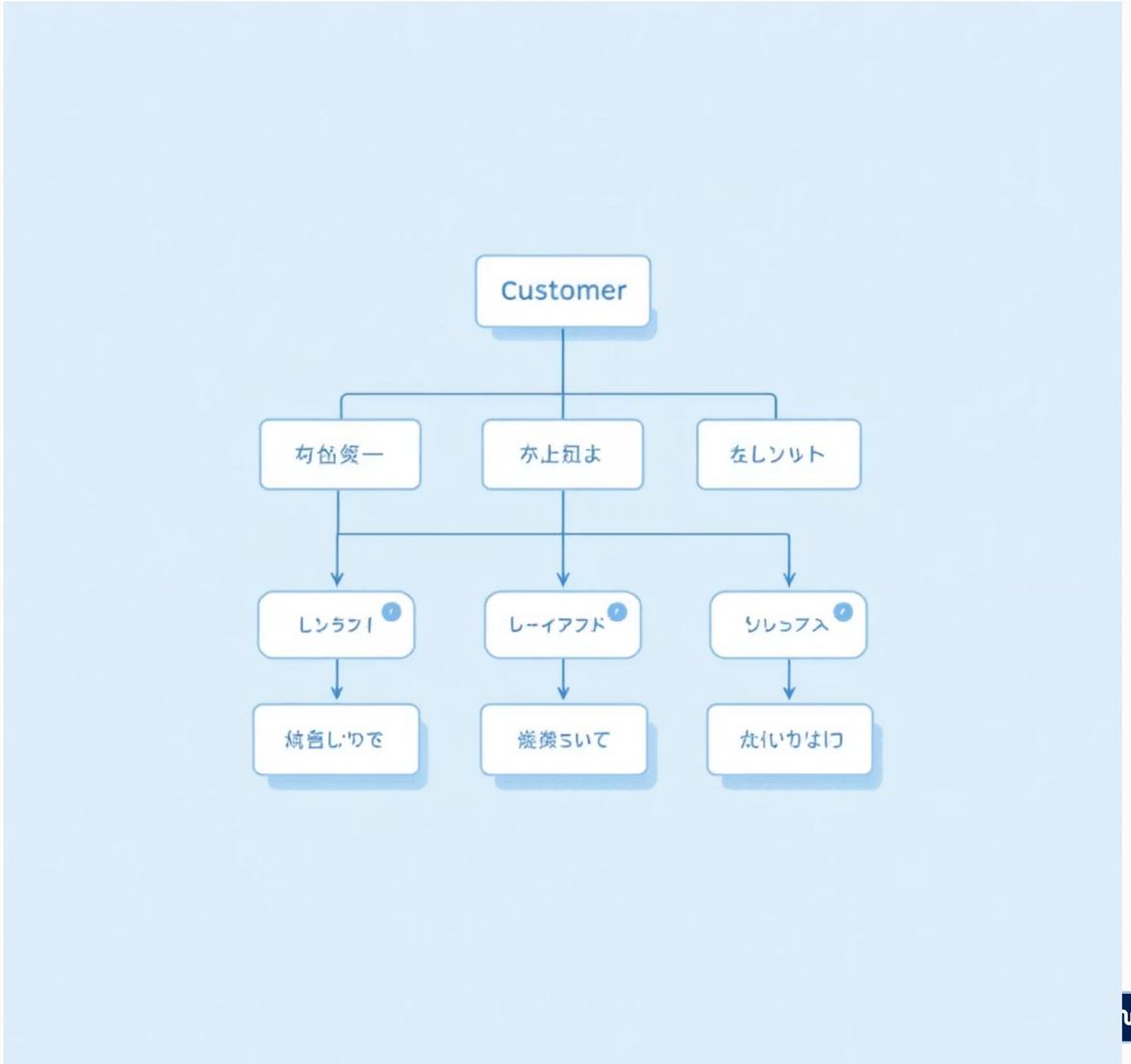
Entidad Cuenta

Define el producto financiero (Tipo de cuenta, saldo, divisa). **Clave: Consistencia.**

3

Tabla Intermedia (Cliente_Cuenta)

Resuelve la M:N. Permite que un cliente tenga N cuentas y una cuenta tenga N titulares. Incluye roles (titular, autorizado).



Capítulo 2

El Núcleo: Gestión de Transacciones

La tabla de transacciones es la más crítica y de mayor volumen. Cada registro debe ser inmutable y reflejar un movimiento financiero único.

Origen y Destino

Claves foráneas para identificar la cuenta emisora y la receptora. Es fundamental para el seguimiento.

Monto y Divisa

Almacenamiento preciso de valores decimales y la divisa (cuidado con la precisión flotante).

Timestamp y Estado

Registro del momento exacto de la operación y su estado (pendiente, completada, fallida).

Atomicidad (ACID)

Cada operación debe ser tratada como una unidad indivisible: o se completa por completo, o no se hace nada. Rollback esencial.

El Principio ACID en Entornos Transaccionales

El cumplimiento de ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) no es negociable en sistemas donde la integridad del saldo es la máxima prioridad.

Atomicidad

Todas o ninguna. Una transferencia debe restar de una cuenta y sumar a otra en una única operación lógica.

Consistencia

La transacción mueve la BBDD de un estado válido a otro, manteniendo todas las reglas de negocio (p.ej., el saldo nunca debe ser negativo sin crédito asociado).

Aislamiento (Isolation)

Las transacciones concurrentes no deben interferir entre sí. Evita problemas como lecturas sucias o perdidas.

Durabilidad

Una vez que una transacción ha sido confirmada, sus cambios son permanentes y sobreviven a fallos del sistema.

Capítulo 3

Seguridad, Logs y Auditoría Inmutable

La confianza en un sistema bancario depende de la capacidad de auditar cada cambio, incluso en el caso de fraude o errores de procesamiento.

Tabla de Eventos/Auditoría

Separar las transacciones de los eventos del sistema. Registrar quién, cuándo y qué se modificó. Es una tabla de **sólo inserción** (Append-Only).

Tokens y Autenticación

No almacenar contraseñas ni información de tarjetas en texto plano. Uso de hashing robusto (bcrypt) y cifrado de datos sensibles (cifrado en reposo y en tránsito).

Cumplimiento Normativo

Asegurar estructuras de datos que faciliten el cumplimiento de regulaciones como GDPR (derecho al olvido, seudónimos) y PCI DSS (protección de datos de tarjetas).

Capítulo 4

Desafíos de Escalabilidad y Alto Rendimiento

Los sistemas bancarios modernos manejan millones de transacciones por día. El diseño debe anticipar el crecimiento masivo de datos.

Particionamiento (Sharding)

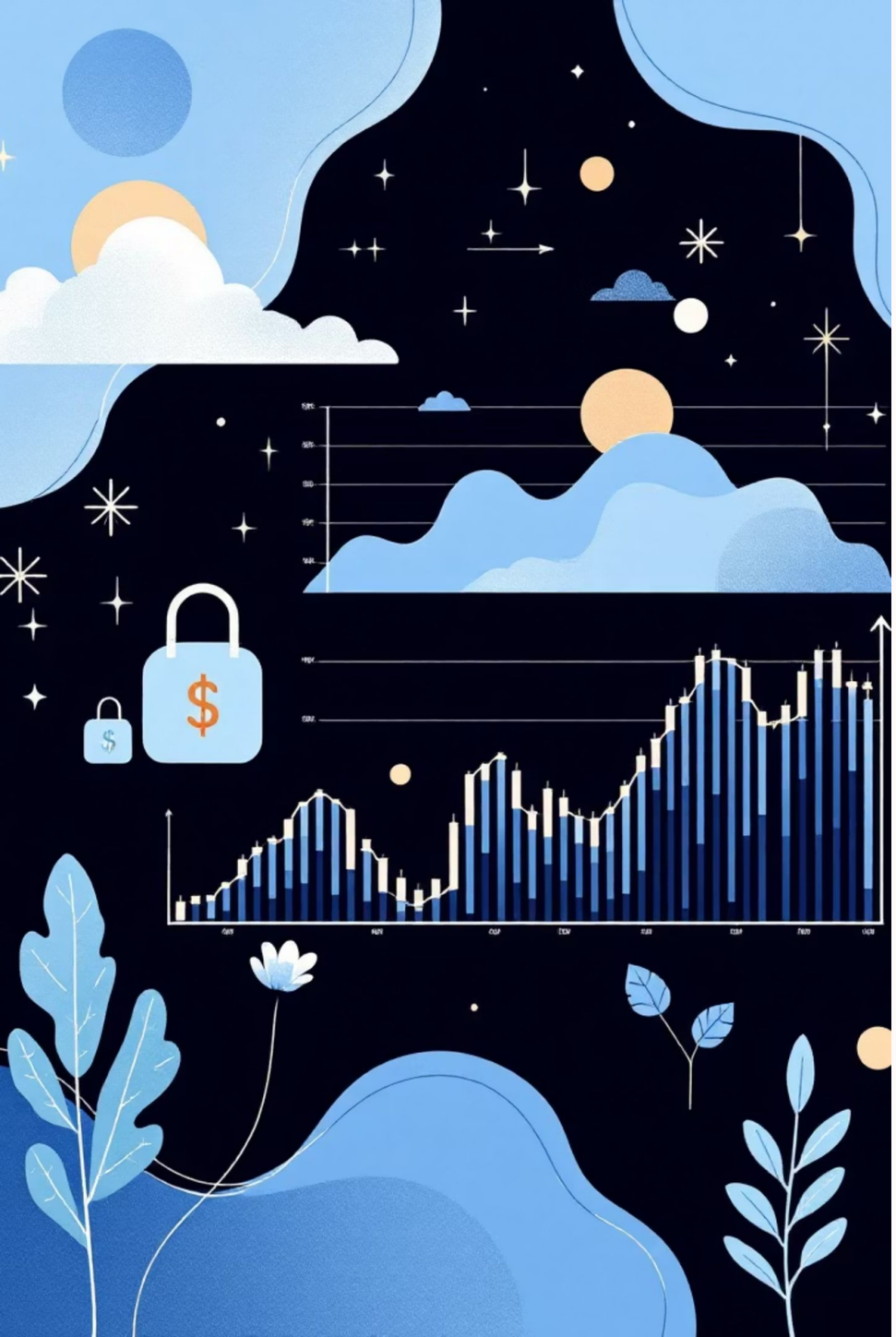
Dividir la tabla de transacciones horizontalmente (p.ej., por región o por hash de cuenta) para distribuir la carga entre múltiples servidores.

Replicación y Alta Disponibilidad

Utilizar réplicas (Maestro-Esclavo) para balancear la carga de lectura. Esencial para la continuidad del negocio (DR) y la baja latencia.

Caché y Materialización

El saldo de una cuenta es un valor frecuentemente consultado. Almacenar saldos en una capa de caché (Redis, Memcached) o como vistas materializadas para evitar sumar todas las transacciones en cada consulta.



Conclusiones y Próximos Pasos

Finalmente, el diseño de base de datos bancaria exige un equilibrio entre la velocidad de acceso, la integridad de los datos (ACID) y la trazabilidad (Auditoría).

Integridad Primero

Priorizar las garantías ACID sobre la optimización prematura. La pérdida de datos es inaceptable.

Diseño Inmutable

Adoptar un modelo de "registro de eventos" (event sourcing) donde los saldos se calculan y las transacciones nunca se eliminan.

Separación de Cargas

Separar las BBDD OLTP (Transaccionales) de las BBDD OLAP (Análisis/Reporting) para mantener el rendimiento.

¿Preguntas sobre el particionamiento de datos o la implementación de las reglas ACID en frameworks ORM?