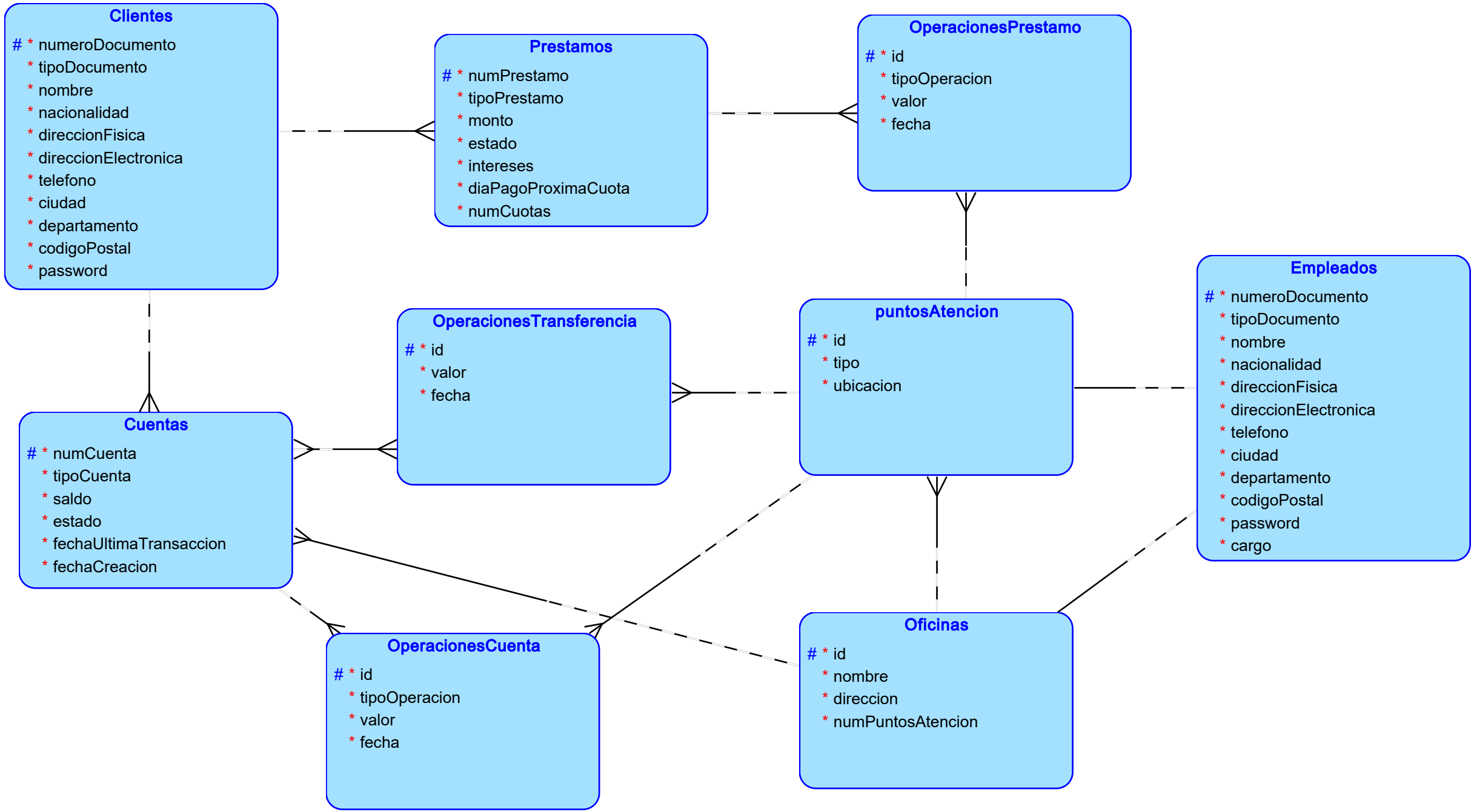
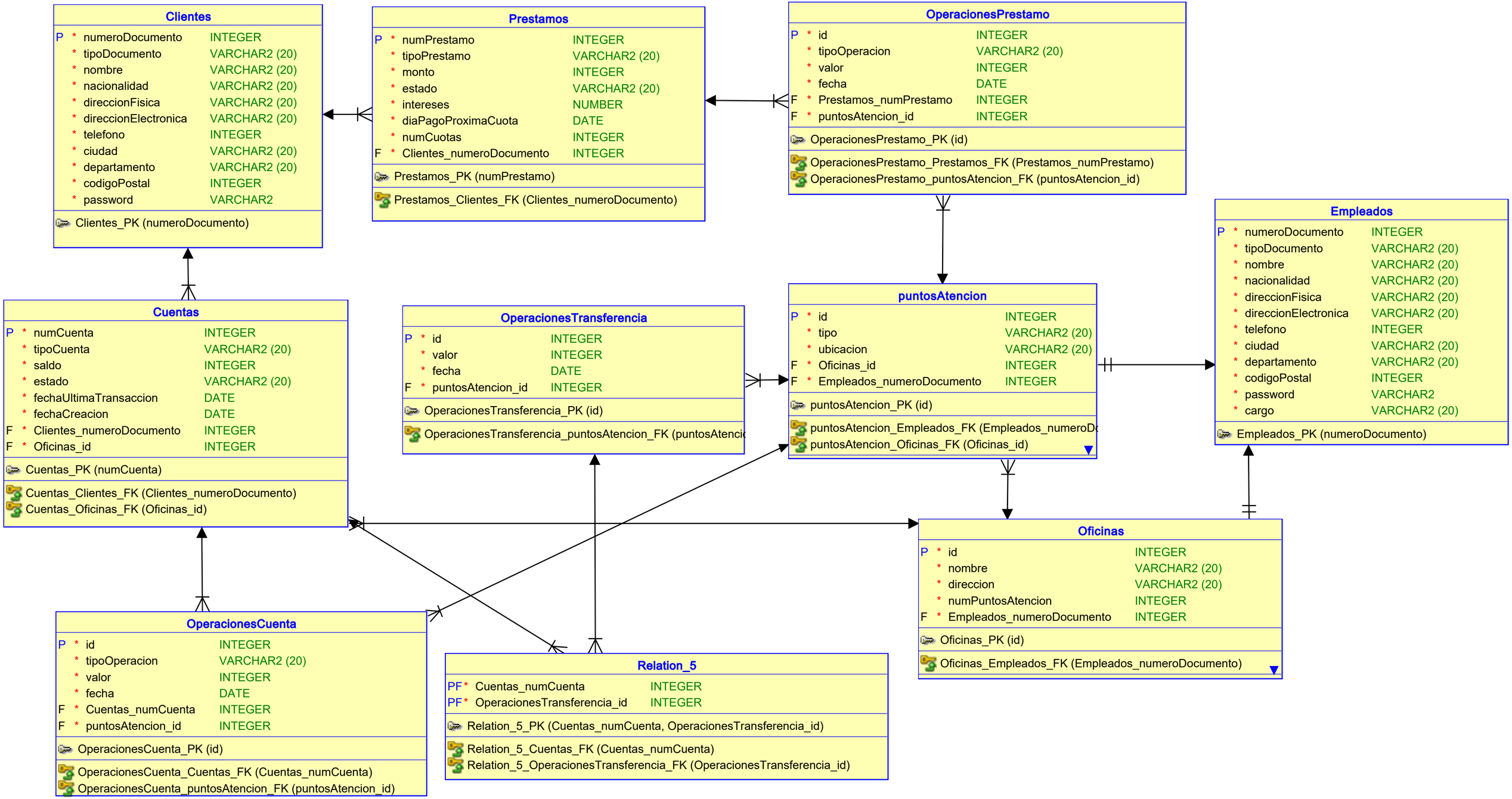

Clientes
* numeroDocumento
* tipoDocumento
* nombre
* nacionalidad
* direccionFisica
* direccionElectronica
* telefono
* ciudad
* departamento
* codigoPostal
* password
Prestamos
* numPrestamo
* tipoPrestamo
* monto
* estado
* intereses
* diaPagoProximaCuota
* numCuotas
OperacionesPrestamo
* id
* tipoOperacion
* valor
* fecha
Empleados
* numeroDocumento
* tipoDocumento
* nombre
* nacionalidad
* direccionFisica
* direccionElectronica
* telefono
* ciudad
* departamento
* codigoPostal
* password
* cargo
OperacionesTransferencia
* id
* valor
* fecha
puntosAtencion
* id
* tipo
* ubicacion
Cuentas
* numCuenta
* tipoCuenta
* saldo
* estado
* fechaUltimaTransaccion
* fechaCreacion
OperacionesCuenta
* id
* tipoOperacion
* valor
* fecha
Oficinas
* id
* nombre
* direccion
* numPuntosAtencion


Clientes
P * numeroDocumento INTEGER
* tipoDocumento VARCHAR2 (20)
* nombre VARCHAR2 (20)
* nacionalidad VARCHAR2 (20)
* direccionFisica VARCHAR2 (20)
* direccionElectronica VARCHAR2 (20)
* telefono INTEGER
* ciudad VARCHAR2 (20)
* departamento VARCHAR2 (20)
* codigoPostal INTEGER
* password VARCHAR2
Clientes_PK (numeroDocumento)
Prestamos
P * numPrestamo INTEGER
* tipoPrestamo VARCHAR2 (20)
* monto INTEGER
* estado VARCHAR2 (20)
* intereses NUMBER
* diaPagoProximaCuota DATE
* numCuotas INTEGER
F * Clientes_numeroDocumento INTEGER
Prestamos_PK (numPrestamo)
Prestamos_Clientes_FK (Clientes_numeroDocumento)
OperacionesPrestamo
P * id INTEGER
* tipoOperacion VARCHAR2 (20)
* valor INTEGER
* fecha DATE
F * Prestamos_numPrestamo INTEGER
F * puntosAtencion_id INTEGER
OperacionesPrestamo_PK (id)
OperacionesPrestamo_Prestamos_FK (Prestamos_numPrestamo)
OperacionesPrestamo_puntosAtencion_FK (puntosAtencion_id)
Empleados
P * numeroDocumento INTEGER
* tipoDocumento VARCHAR2 (20)
* nombre VARCHAR2 (20)
* nacionalidad VARCHAR2 (20)
* direccionFisica VARCHAR2 (20)
* direccionElectronica VARCHAR2 (20)
* telefono INTEGER
* ciudad VARCHAR2 (20)
* departamento VARCHAR2 (20)
* codigoPostal INTEGER
* password VARCHAR2
* cargo VARCHAR2 (20)
Empleados_PK (numeroDocumento)
Cuentas
P * numCuenta INTEGER
* tipoCuenta VARCHAR2 (20)
* saldo INTEGER
* estado VARCHAR2 (20)
* fechaUltimaTransaccion DATE
* fechaCreacion DATE
F * Clientes_numeroDocumento INTEGER
F * Oficinas_id INTEGER
Cuentas_PK (numCuenta)
Cuentas_Clientes_FK (Clientes_numeroDocumento)
Cuentas_Oficinas_FK (Oficinas_id)
OperacionesTransferencia
P * id INTEGER
* valor INTEGER
* fecha DATE
F * puntosAtencion_id INTEGER
OperacionesTransferencia_PK (id)
OperacionesTransferencia_puntosAtencion_FK (puntosAtenci
puntosAtencion
P * id INTEGER
* tipo VARCHAR2 (20)
* ubicacion VARCHAR2 (20)
F * Oficinas_id INTEGER
F * Empleados_numeroDocumento INTEGER
puntosAtencion_PK (id)
puntosAtencion_Empleados_FK (Empleados_numeroD
puntosAtencion_Oficinas_FK (Oficinas_id)
Oficinas
P * id INTEGER
* nombre VARCHAR2 (20)
* direccion VARCHAR2 (20)
* numPuntosAtencion INTEGER
F * Empleados_numeroDocumento INTEGER
Oficinas_PK (id)
Oficinas_Empleados_FK (Empleados_numeroDocumento)
OperacionesCuenta
P * id INTEGER
* tipoOperacion VARCHAR2 (20)
* valor INTEGER
* fecha DATE
F * Cuentas_numCuenta INTEGER
F * puntosAtencion_id INTEGER
OperacionesCuenta_PK (id)
OperacionesCuenta_Cuentas_FK (Cuentas_numCuenta)
OperacionesCuenta_puntosAtencion_FK (puntosAtencion_id)
Relation_5
PF* Cuentas_numCuenta INTEGER
PF* OperacionesTransferencia_id INTEGER
Relation_5_PK (Cuentas_numCuenta, OperacionesTransferencia_id)
Relation_5_Cuentas_FK (Cuentas_numCuenta)
Relation_5_OperacionesTransferencia_FK (OperacionesTransferencia_id)

Análisis y revisión:

Para el modelo lógico quitamos varias cosas que no eran necesarias y creamos unas entidades que creímos eran mejores. Por ejemplo, creamos Clientes y Empleados, además para cada producto creamos una entidad de operación como: operacionCuentas, operacionTransferencias y operacionPrestamo. Esto tiene más sentido a los anteriores modelos que teníamos, ya que había muchas entidades como Sistemas de manejo que en realidad no hacían nada, teníamos retiro, consignación y transferencia como entidades, estas las transformamos por las operaciones, puesto que era más sencillo de guardar la información y tenia más sentido, borramos la entidad solicitudPrestamo porque no hacía nada, también borramos cdt porque no era necesaria en la aplicación. Borramos cajero automático ya que era un tipo de punto de atención. Con estos cambios obtuvimos mejores resultados y también más simples a lo que teníamos antes.

Esto nos dio unas tablas diferentes a las que teníamos en la anterior entrega. En la primera entrega nos dieron muchas tablas que no eran necesarias y no tenían información importante. En esta entrega tenemos tablas con Primary Keys claras y con las Foreign key necesarias para en lazar cosas como el cliente y sus productos, puntos de atención con oficinas, etc. Quitamos algunas PK que eran múltiples ya que hacía muy difícil implementación en código y las dejamos sencillas.

Cada relación que obtuvimos está en su nivel de normalización de Boyce-Codd lo cual asegura su integridad y eficiencia. Para cumplir con el FN 1, se garantiza que ninguna tabla tenga atributos en forma de lista, es decir, todos los atributos son atómicos. Esto se logra mediante el uso de clases de asociación y/o Foreign Keys (FKs). Asimismo, para cumplir con el FN2, se verifica la ausencia de dependencias parciales entre atributos primos y no primos dentro de las relaciones. La separación de entidades desde el UML contribuye en gran medida a este logro. Por otro lado, las llaves primarias están bien pensadas y se componen estrictamente de los atributos necesarios para no generar dependencias parciales entre los atributos. Por otro lado, para el FN3, se asegura la inexistencia de dependencias transitivas entre atributos no primos dentro de las relaciones. Nuevamente, la separación de entidades desde el UML desempeña un papel importante en este aspecto. Además, en general, los atributos no primos no tienen mucha relación por lo que no hay dependencias. En cuanto a la BCNF, se garantiza que cada relación tenga una única llave candidata que se convierte en Primary Key (PK). Esto se logra mediante la creación de atributos de identificación solo cuando es necesario y al aprovechar las reglas de negocio para establecer la PK. Como resultado, no hay varias llaves candidatas que se superpongan, lo que cumple con los criterios de la BCNF.

Documentación de java Spring:

Se creo una carpeta llamada “modelo” donde pusimos todas las entidades de la aplicación, después hicimos otra carpeta llamada “repositorio” acá fue donde pedimos directamente a cada tabla de la base de datos usando sql diferentes cosas como insertar nuevos datos usando “INSERT INTO…” o conseguir información como una oficina usando el id de esta. Luego creamos una carpeta “controller” donde están todos los controllers de las diferentes entidades, en donde podíamos pedir por ejemplo la lista de oficinas o crear nuevas oficinas y guardarlas. Por último, creamos toda la parte del front usando archivos html. Tenemos diferentes archivos como “clientes.html” donde se muestra la tabla completa de los clientes que existan, además hay templates para crear un nuevo cliente, etc.