



inverza



Proyecto de Bases de Datos

Entrega final



Índice

1. Diseño conceptual

- 1.1. Enunciado
- 1.2. Esquema E/R
- 1.3. Atributos

2. Diseño lógico

- 2.1. Modelo relacional
- 2.2. Normalización 1ªFN
- 2.3. Normalización 2ªFN
- 2.4. Normalización 3ªFN

3. Diseño físico

- 3.1. Esquema de tablas
- 3.2. Capturas consultas
- 3.3. Capturas procedimientos
- 3.4. Capturas funciones
- 3.5. Capturas triggers

1. Diseño conceptual

—



1. Diseño conceptual

1.1. Enunciado

Se desea realizar una BBDD de “INVERZA”, un banco especializado en comercializar productos de inversión. Se desea guardar información de las sucursales, empleados, clientes y los diferentes tipos de cuentas bancarias. “INVERZA” está compuesto por sucursales localizadas por todo el país, de estas se desea guardar el *Id de la sucursal, dirección, población, CP, teléfonos y email*.

A las sucursales pertenecen personas, estas pueden ser empleados o clientes, una persona no puede ser empleado y cliente a la vez.

De las personas se desea guardar *DNI, nombre, dirección, población, CP, teléfono*. Las personas, independientemente de si son clientes o empleados, pueden pedir uno o más préstamos de los que se desea guardar el *código del préstamo, cuantía, fecha de emisión y el plazo de devolución en años*.

Los préstamos pueden ser de tipo personal o hipoteca.

De los préstamos de tipo personal se desea guardar el *motivo y la TAE*, mientras que de las hipotecas el *tipo de vivienda*. Las hipotecas a su vez pueden ser de tipo variable o tipo fijo, de las primeras se guardará además el *euribor y el diferencial fijo* mientras que de las segundas únicamente la *TAE*.



1. Diseño conceptual

1.1. Enunciado

De los empleados se desea guardar el *código de empleado*, *salario*, *fecha de inicio de contrato*, *antigüedad* y *cualificación*. Los empleados únicamente pueden obtener préstamos y tener una única cuenta corriente.

De cada sucursal 1 empleado es el director y es el jefe del resto de empleados, de este se desea guardar el *número de empleados a cargo*.

De los clientes se desea guardar *código del cliente*, *edad* y *e-mail*. Estos pueden crear como máximo 2 cuentas bancarias, una cuenta corriente y una de ahorro, sin embargo pueden ser autorizados en 1 o más cuentas. Los clientes realizan transacciones, de las que se desea *guardar código de transacción*, *cantidad*, *fecha* y *tipo*. Estas transacciones se ejecutan en las cuentas bancarias de las que son autorizados.

De las cuentas bancarias se desea guardar el *IBAN* y *fecha de apertura*. Estas pueden ser cuentas corrientes en cuyo caso se guarda el *saldo*, además estas pueden ser de tipo normal, oro o platinum según el saldo que tengan, de 1.000€ a 50.000€ la primera, de 51.000€ a 999.999€ la segunda y de a partir de 1.000.000 € la tercera.



1. Diseño conceptual

1.1. Enunciado

También existen las cuentas de ahorro de las que se desea guardar *aportaciones, valor actual y rentabilidad*. Estas pueden ser a plazo fijo y en cuyo caso se almacenará el interés y el plazo de retorno.

Las cuentas de ahorro también pueden ser cuentas de inversiones, compuesta por 1 o varios productos de inversión. "INVERZA" comercializa con acciones, bonos y fondos indexados, de las acciones se desea guardar el *nombre de la empresa, ISIN, ticker, cantidad, precio de compra, precio actual, capitalización, rentabilidad YTD y dividendos YTD*. Además las acciones pertenecen a índices de referencia de los que se desea guardar la siguiente información: *ticker, sector, capitalización, rentabilidad YTD*.

De los bonos se desea guardar *código de bono, rentabilidad y plazo de retorno*. Estos bonos son emitidos por países que a su vez tienen 1 o varios índices de referencia, se desea guardar de los países el *nombre y el tipo de país* (Desarrollado o Emergente).



1. Diseño conceptual

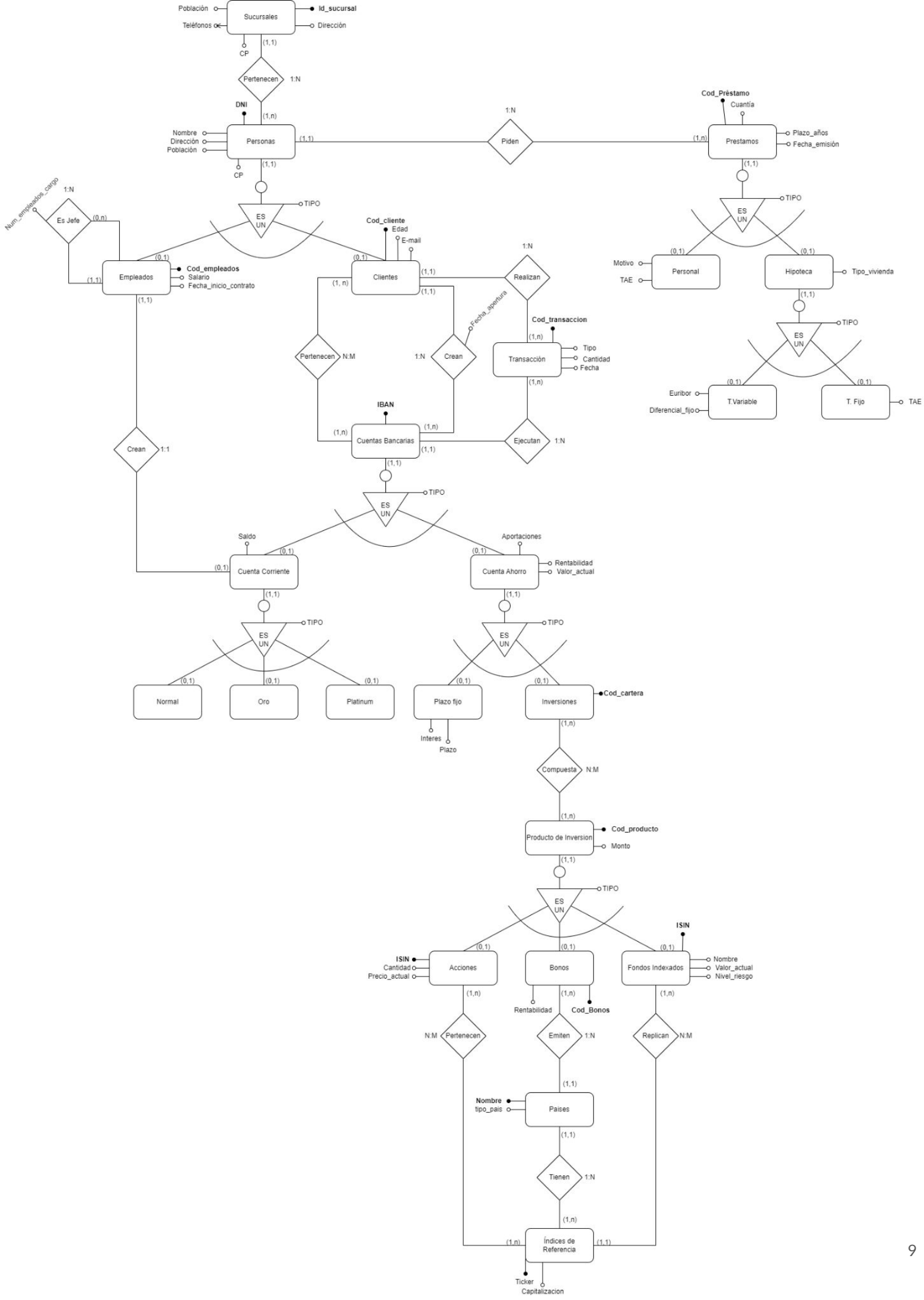
1.1. Enunciado

Por último de los fondos indexados se desea guardar *ISIN, nombre, valor actual, cantidad de participaciones, precio de compra, rentabilidad YTD, nivel de riesgo y capitalización*. Cada uno de los fondos indexados replican un solo índice de referencia.



1. Diseño conceptual

1.2. Esquema E/R





1. Diseño conceptual

1.3. Atributos

Sucursales: Id Sucursal, dirección, población, CP, teléfonos, E-mail.

Personas: DNI, nombre, dirección, población, CP, teléfono.

Préstamos: Cod Préstamo, cuantía, fecha emisión, plazo años.

Personal: Motivo, TAE.

Hipoteca: tipo vivienda.

T.Variable: Euribor, Diferencial Fijo.

T.Fijo: TAE.

Empleados: Cod Empleado, salario, fecha inicio contrato, antigüedad, cualificación.

Jefe: Núm empleados cargo.

Clientes: Cod Cliente, edad, E-mail.

Transacciones: Cod Transacción, cantidad, fecha, tipo.



1. Diseño conceptual

1.3. Atributos

Cuentas Bancarias: IBAN.

Cuentas Corrientes: Saldo.

Cuenta Ahorro: Aportaciones, Valor Actual, rentabilidad.

C.Plazo Fijo: Interés, plazo.

C.Inversiones: Código Cartera.

Producto Inversión: Cód Producto, monto.

Acciones: ISIN, Nombre Empresa, ticker, cantidad, precio compra, precio actual, capitalización, rentabilidad YTD, dividendos YTD.

Bonos: Cod Bono, rentabilidad, plazo.

Fondo Indexado: ISIN, nombre, valor actual, cantidad de participaciones, precio compra, rentabilidad YTD, nivel riesgo, capitalización.



1. Diseño conceptual

1.3. Atributos

Índice de referencia: Ticker, sector, capitalización, rentabilidad YTD.

Países: Nombre, tipo país.

2. Diseño lógico

—



2. Diseño lógico

2.1. Modelo relacional





2. Diseño lógico

2.2. Normalización 1ªFN

Trás analizar las tablas resultantes del modelo relacional, observamos que únicamente hay una tabla que no se encuentra en 1ºFN.

Recordemos que para que se encuentre en 1ºFN todos sus atributos deben ser atómicos, sin embargo, en la tabla "SUCURSALES" se encuentra el atributo "teléfonos" que es un atributo multivaluado. El proceso de normalizar esta tabla en 1ºFN nos daría lugar a otra tabla que llamaremos "TELÉFONOS".

Finalmente el resultado de normalizar hasta 1ºFN la tabla "SUCURSALES" sería el siguiente:

SUCURSALES(Id_sucursal, Direccion, Poblacion, CP, Email)

TELEFONOS (Id_sucursal(FK), Telefono)



2. Diseño lógico

2.3. Normalización 2ªFN

Trás analizar las tablas resultantes, observamos que todas ellas se encuentran en 2ºFN.

Recordemos que para que una tabla se encuentre en 2ºFN se deben cumplir 2 condiciones, la primera es que se encuentre en 1ºFN y la segunda es que todos los atributos dependan funcionalmente de la clave primaria al completo si esta fuera compuesta, lo que denominamos una dependencia funcional completa.

En el caso de las tablas resultantes, la mayoría tienen una clave primaria compuesta por un solo atributo, por lo que todas ellas se encuentran automáticamente en 2ºFN.

También nos encontramos con alguna tabla compuesta únicamente por una clave primaria y sin ningún atributo más, estas también se encuentran en 2ºFN automáticamente.

Por último, en las tablas que poseen una clave primaria compuesta y además tienen atributos, observamos que todos sus atributos dependen funcionalmente de la clave primaria en su totalidad.



2. Diseño lógico

2.4. Normalización 3ªFN

Trás analizar las tablas resultantes, observamos que únicamente las tablas “SUCURSALES” y “PERSONAS” no se encuentran en 3ºFN.

Recordemos que para que una tabla se encuentre en 3ºFN se deben cumplir 3 condiciones, la primera que se encuentre en 1ºFN, la segunda que se encuentre en 2ºFN, y por último que no existan dependencias funcionales transitivas, es decir, que no existan atributos que sean funcionalmente dependientes de otro atributo y no de la clave primaria.

Para normalizar las tablas “SUCURSALES” y “PERSONAS” debemos analizar sus dependencias funcionales:



2. Diseño lógico

2.4. Normalización 3ªFN

SUCURSALES(**Id_sucursal**, Direccion, Poblacion, CP, Email)

(**Id_sucursal**) ----- Direccion, Email.

(**CP**) ----- Poblacion.

PERSONAS(**DNI**, Id_sucursal(FK), Nombre, Direccion, Poblacion, CP, Telefono)

(**DNI**) ----- Id_sucursal, Nombre, Direccion, Telefono.

(**CP**) ----- Poblacion.

Observamos que en ambas tablas existe la misma dependencia funcional transitiva de “CP” y “Población”, de manera que únicamente necesitamos crear una tabla “POBLACIONES” que almacene el “CP” y la “Poblacion” para que ambas tablas se encuentren en 3ªFN, la tablas normalizadas quedarían de la siguiente manera:



2. Diseño lógico

2.4. Normalización 3ªFN

SUCURSALES(**Id_sucursal**, Direccion, CP(FK), Email)

POBLACIONES(**CP**, Poblacion)

PERSONAS(**DNI**, Id_sucursal(FK), Nombre, Direccion, CP(FK), Telefono)



Modelo relacional normalizado



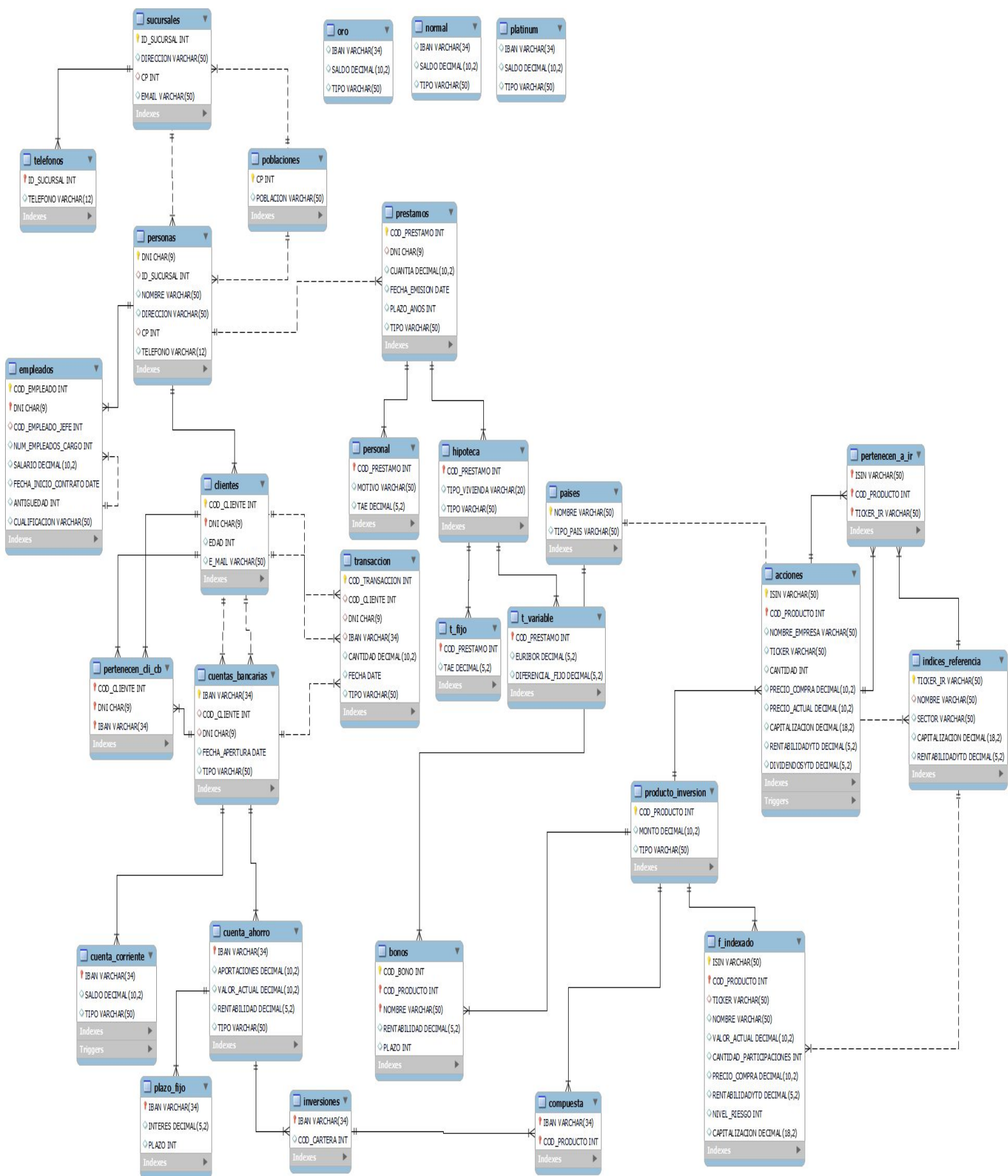
3. Diseño físico





3. Diseño físico

3.1. Esquema de tablas





3. Diseño físico

3.1. Capturas consultas

```

517 -- 1.Consulta para obtener la cantidad total de préstamos personales.
518 * SELECT COUNT(*) AS "Préstamos personales" FROM PERSONAL;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Préstamos personales
8

```

520 -- 2.Consulta para obtener el nombre de todas las personas que han realizado al menos una transacción con un monto superior a 1000.
521 * SELECT DISTINCT PE.NOMBRE
522 FROM PERSONAS PE
523 INNER JOIN CLIENTES C ON PE.DNI = C.DNI
524 INNER JOIN TRANSACCION T ON C.COD_CLIENTE = T.COD_CLIENTE
525 WHERE T.CANTIDAD > 1000;
526

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

NOMBRE
María García

```

527 -- 3.Consulta para obtener la cantidad total de préstamos personales y hipotecarios.
528 * SELECT
529 (SELECT COUNT(*) FROM PERSONAL) as "Préstamos personales",
530 (SELECT COUNT(*) FROM HIPOTECA) as "Hipotecas";
531

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Préstamos personales	Hipotecas
8	4

```

532 -- 4 Consulta para obtener el nombre y la dirección de todos los empleados que tienen más de 5 años de antigüedad en la empresa.
533 * SELECT PE.NOMBRE, PE.DIRECCION
534 FROM PERSONAS PE
535 INNER JOIN EMPLEADOS E ON PE.DNI = E.DNI
536 WHERE E.ANTIGUEDAD > 5;
537

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

NOMBRE	DIRECCION
Javier Torres	Calle Xàtiva, 16

```

538 -- 5 Consulta para obtener la suma total de las cuantías de los préstamos hipotecarios.
539 * SELECT ROUND(SUM(CUANTIA)) AS "Suma cuantía hipotecas" FROM PRESTAMOS WHERE TIPO = 'HIPOTECA';
540

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Suma cuantía hipotecas
255000

```

541 -- 6 Consulta para obtener el promedio de las cuantías de los préstamos personales.
542 * SELECT ROUND(AVG(CUANTIA)) AS "Media cuantía prestamos personales" FROM PRESTAMOS WHERE TIPO = 'PERSONAL';
543

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Media cuantía prestamos personales
17500

```

544 -- 7 Consulta para obtener el total de préstamos hipotecarios por plazo (en años).
545 • SELECT PR.PLAZO_ANOS, COUNT(*) AS "Total Hipotecas"
546 FROM HIPOTECA H
547 INNER JOIN PRESTAMOS PR ON H.COD_PRESTAMO = PR.COD_PRESTAMO
548 GROUP BY PR.PLAZO_ANOS;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	PLAZO_ANOS	Total Hipotecas
▶	5	1
	10	1
	8	1
	15	1

```

550 -- 8 Consulta para obtener el nombre y dirección de todas las sucursales junto con el nombre de la población a la que pertenecen.
551 • SELECT S.DIRECCION, P.POBLACION
552 FROM SUCURSALES S
553 INNER JOIN POBLACIONES P ON S.CP = P.CP;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	DIRECCION	POBLACION
▶	Calle Mayor, 1	Madrid
	Calle Atocha, 34	Madrid
	Calle Serrano, 56	Madrid
	Avenida Diagonal, 123	Barcelona
	Calle Aragón, 321	Barcelona

```

556 -- 9 Consulta para obtener el nombre y la dirección de todas las personas junto con el nombre de la población a la que pertenecen.
557 • SELECT PE.NOMBRE, PE.DIRECCION, PO.POBLACION
558 FROM PERSONAS PE
559 INNER JOIN POBLACIONES PO ON PE.CP = PO.CP;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	NOMBRE	DIRECCION	POBLACION
▶	Sonia Ortiz	Calle Larios, 6	Málaga
	Antonio Morales	Avenida Andalucía, 67	Málaga
	Juan Pérez	Calle Mayor, 2	Madrid
	Carmen Gutiérrez	Calle Sierpes, 34	Sevilla
	María García	Calle Atocha, 35	Madrid

```

561 -- 10 Consulta para obtener el nombre y el teléfono de todas las personas que trabajan en cada sucursal.
562 • SELECT PE.NOMBRE AS "NOMBRE EMPLEADO", PE.TELEFONO, S.ID_SUCURSAL
563 FROM PERSONAS PE
564 INNER JOIN EMPLEADOS E ON PE.DNI = E.DNI
565 INNER JOIN SUCURSALES S ON PE.ID_SUCURSAL = S.ID_SUCURSAL;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	NOMBRE EMPLEADO	TELEFONO	ID_SUCURSAL
▶	Sonia Ortiz	601234567	10
	Antonio Morales	602345678	11
	Carmen Gutiérrez	603456789	12
	Javier Torres	690123456	9


```

566 -- 11 Consulta para obtener la cantidad total de cuentas bancarias por población.
567 * SELECT PO.POBACION, COUNT(*) AS "Total Cuentas Bancarias"
568 FROM CUENTAS_BANCARIAS CB
569 INNER JOIN CLIENTES C ON CB.COD_CLIENTE = C.COD_CLIENTE
570 INNER JOIN PERSONAS PE ON C.DNI = PE.DNI
571 INNER JOIN POBLACIONES PO ON PE.CP = PO.CP
572 GROUP BY PO.POBACION;

```

Result Grid Filter Rows: Export: Wrap Cell Content:

	POBACION	Total Cuentas Bancarias
▶	Madrid	3
	Barcelona	3
	Valencia	2

```

574 -- 12 Consulta para obtener la cantidad total de cuentas de ahorro.
575 * SELECT COUNT(*) AS "CUENTAS AHORRO" FROM CUENTA_AHORRO;

```

Result Grid Filter Rows: Export: Wrap Cell Content:

	CUENTAS AHORRO
▶	4

```

577 -- 13 Consulta para obtener la cantidad total de préstamos personales por población.
578 * SELECT PO.POBACION, COUNT(*) AS "Préstamos personales"
579 FROM PERSONAL P
580 INNER JOIN PRESTAMOS PR ON P.COD_PRESTAMO = PR.COD_PRESTAMO
581 INNER JOIN PERSONAS PE ON PR.DNI = PE.DNI
582 INNER JOIN POBLACIONES PO ON PE.CP = PO.CP
583 GROUP BY PO.POBACION;

```

Result Grid Filter Rows: Export: Wrap Cell Content:

	POBACION	Préstamos personales
▶	Madrid	3
	Barcelona	3
	Valencia	2

```

585 -- 14 Consulta para obtener el valor total en cuentas de ahorro.
586 * SELECT SUM(VALOR_ACTUAL) AS "VALOR TOTAL CUENTA AHORRO" FROM CUENTA_AHORRO;

```

Result Grid Filter Rows: Export: Wrap Cell Content:

	VALOR TOTAL CUENTA AHORRO
▶	39390.00

```

588 -- 15 Consulta los clientes que tienen más de una cuenta bancaria.
589 * SELECT c.*, COUNT(pcc.IBAN) AS Numero_de_Cuentas
590 FROM CLIENTES c
591 JOIN PERTENECEN_CLI_CB pcc ON c.COD_CLIENTE = pcc.COD_CLIENTE AND c.DNI = pcc.DNI
592 GROUP BY c.COD_CLIENTE, c.DNI
593 HAVING COUNT(pcc.IBAN) > 1;

```

Result Grid Filter Rows: Export: Wrap Cell Content:

	COD_CLIENTE	DNI	EDAD	E_MAIL	Numero_de_Cuentas
▶	1	12345678A	25	juan.perez@email.com	2
	2	23456789B	32	maria.garcia@email.com	2
	6	67890123F	50	ana.sanchez@email.com	2
	8	89012345H	42	teresa.lopez@email.com	2

```

595 -- 16 Consulta las sucursales que tienen una cantidad de empleados superior o igual a la media de empleados por sucursal.
596 * SELECT s.ID_SUCURSAL, s.DIRECCION, s.CP, s.EMAIL, COUNT(e.COD_EMPLEADO) AS Numero_de_Empleados
597 FROM SUCURSALES s
598 JOIN PERSONAS p ON s.ID_SUCURSAL = p.ID_SUCURSAL
599 JOIN EMPLEADOS e ON p.DNI = e.DNI
600 GROUP BY s.ID_SUCURSAL, s.DIRECCION, s.CP, s.EMAIL
601 HAVING COUNT(e.COD_EMPLEADO) >= (
602     SELECT AVG(Numero_de_Empleados) AS Media_Empleados
603     FROM (
604         SELECT p.ID_SUCURSAL, COUNT(e.COD_EMPLEADO) AS Numero_de_Empleados
605         FROM EMPLEADOS e
606         JOIN PERSONAS p ON e.DNI = p.DNI
607         GROUP BY p.ID_SUCURSAL
608     ) AS EmpleadosPorSucursal
609 );

```

ID_SUCURSAL	DIRECCION	CP	EMAIL	Numero_de_Empleados
10	Calle Larios, 5	29001	sucursal10@banco.es	1
11	Avenida Andalucía, 66	29002	sucursal11@banco.es	1
12	Calle Serpientes, 33	41001	sucursal12@banco.es	1
9	Calle Xàtiva, 15	46003	sucursal09@banco.es	1

```

611 -- 17 Consulta los préstamos con una cuantía superior al promedio de cuantía de todos los préstamos.
612 * SELECT P.*
613 FROM PRESTAMOS P
614 WHERE P.CUANTIA > (
615     SELECT AVG(CUANTIA)
616     FROM PRESTAMOS
617 );
618

```

COD_PRESTAMO	DNI	CUANTIA	FECHA_EMISION	PLAZO_ANOS	TIPO
2	23456789B	50000.00	2022-02-28	10	HIPOTECA
4	45678901D	75000.00	2022-04-23	15	HIPOTECA
8	89012345H	100000.00	2022-08-31	20	HIPOTECA
*	NULL	NULL	NULL	NULL	NULL

```

619 -- 18 Consulta las cuentas de ahorro con un valor actual superior a 5000.
620 * SELECT *
621 FROM CUENTA_AHORRO
622 WHERE VALOR_ACTUAL > 5000;
623

```

IBAN	APORTACIONES	VALOR_ACTUAL	RENTABILIDAD	TIPO
ES0012345678012345678901	5000.00	5200.00	4.00	PLAZO_FIJO
ES0012345678034567890123	10000.00	10200.00	2.00	INVERSIONES
ES0012345678056789012345	15000.00	15750.00	5.00	PLAZO_FIJO
ES0012345678078901234567	8000.00	8240.00	3.00	INVERSIONES
*	NULL	NULL	NULL	NULL

```

624 -- 19 Consulta las cuentas bancarias que no tienen ninguna transacción registrada.
625 * SELECT CB.*
626 FROM CUENTAS_BANCARIAS CB
627 LEFT JOIN TRANSACCION T ON CB.IBAN = T.IBAN
628 WHERE T.COD_TRANSACCION IS NULL;

```

IBAN	COD_CLIENTE	DNI	FECHA_APERTURA	TIPO
------	-------------	-----	----------------	------

```

630 -- 20 Consulta la cantidad total de empleados por sucursal.
631 * SELECT S.ID_SUCURSAL, S.DIRECCION, COUNT(P.DNI) AS NUM_EMPLEADOS
632 FROM SUCURSALES S
633 LEFT JOIN PERSONAS P ON S.ID_SUCURSAL = P.ID_SUCURSAL
634 WHERE P.DNI IN (SELECT DNI FROM EMPLEADOS)
635 GROUP BY S.ID_SUCURSAL, S.DIRECCION
636 ORDER BY S.ID_SUCURSAL;

```

Result Grid Filter Rows: Export: Wrap Cell Content:

	ID_SUCURSAL	DIRECCION	NUM_EMPLEADOS
▶	9	Calle Xàtiva, 15	1
	10	Calle Larios, 5	1
	11	Avenida Andalucía, 66	1
	12	Calle Sierpes, 33	1

```

638 -- 21 Consulta la cantidad total de préstamos de cada tipo en la tabla de préstamos.
639 * SELECT TIPO, COUNT(*) AS NUM_PRESTAMOS
640 FROM PRESTAMOS
641 GROUP BY TIPO;

```

Result Grid Filter Rows: Export: Wrap Cell Content:

	TIPO	NUM_PRESTAMOS
▶	PERSONAL	4
	HIPOTECA	4

```

643 -- 22 Consulta las sucursales que no tienen teléfono registrado.
644 * SELECT S.*
645 FROM SUCURSALES S
646 LEFT JOIN TELEFONOS T ON S.ID_SUCURSAL = T.ID_SUCURSAL
647 WHERE T.TELEFONO IS NULL;

```

Result Grid Filter Rows: Export: Wrap Cell Content:

	ID_SUCURSAL	DIRECCION	CP	EMAIL
--	-------------	-----------	----	-------

```

649 -- 23 Consulta el promedio de cuantía de los préstamos personales según su motivo.
650 * SELECT P.MOTIVO, ROUND(AVG(PR.CUANTIA)) AS PROMEDIO_CUANTIA
651 FROM PERSONAL P
652 INNER JOIN PRESTAMOS PR ON P.COD_PRESTAMO = PR.COD_PRESTAMO
653 GROUP BY P.MOTIVO;

```

Result Grid Filter Rows: Export: Wrap Cell Content:

	MOTIVO	PROMEDIO_CUANTIA
▶	Compra de coche	10000
	Compra de vivienda	50000
	Reformas en el hogar	25000
	Consolidación de deudas	75000
	Vacaciones	15000
	Inversión en negocio	30000

```

655 -- 24 Consulta la cantidad de clientes por tipo de cuenta bancaria.
656 * SELECT CB.TIPO, COUNT(*) AS NUM_CLIENTES
657 FROM CUENTAS_BANCARIAS CB
658 INNER JOIN CLIENTES C ON CB.COD_CLIENTE = C.COD_CLIENTE
659 GROUP BY CB.TIPO;

```

Result Grid Filter Rows: Export: Wrap Cell Content:

	TIPO	NUM_CLIENTES
▶	CUENTA_CORRIENTE	4
	CUENTA_AHORRO	4

661 -- 25 Consulta la suma de aportaciones por tipo de cuenta de ahorro.

```
662 * SELECT CA.TIPO, SUM(CA.APORTACIONES) AS TOTAL_APORTACIONES
663 FROM CUENTA_AHORRO CA
664 GROUP BY CA.TIPO;
```

Result Grid Filter Rows: Export: Wrap Cell Content:

TIPO	TOTAL_APORTACIONES
PLAZO_FIJO	20000.00
INVERSIONES	18000.00

666 -- 26 Consulta la rentabilidad promedio de los fondos indexados por nivel de riesgo.

```
667 * SELECT FI.NIVEL_RIESGO, AVG(FI.RENTABILIDADYTD) AS PROMEDIO_RENTABILIDAD
668 FROM F_INDEXADO FI
669 GROUP BY FI.NIVEL_RIESGO;
```

Result Grid Filter Rows: Export: Wrap Cell Content:

NIVEL_RIESGO	PROMEDIO_RENTABILIDAD
5	5.000000
4	4.000000
3	3.500000
6	6.000000
7	5.500000

671 -- 27 Consulta la cantidad total de empleados por cualificación.

```
672 * SELECT E.CUALIFICACION, COUNT(*) AS NUM_EMPLEADOS
673 FROM EMPLEADOS E
674 GROUP BY E.CUALIFICACION;
```

Result Grid Filter Rows: Export: Wrap Cell Content:

CUALIFICACION	NUM_EMPLEADOS
Director	1
Subdirector	2
Empleado	1

676 -- 28 Consulta el monto total invertido en cada tipo de producto de inversión.

```
677 * SELECT PI.TIPO, SUM(PI.MONTO) AS TOTAL_INVERSION
678 FROM PRODUCTO_INVERSION PI
679 GROUP BY PI.TIPO;
```

Result Grid Filter Rows: Export: Wrap Cell Content:

TIPO	TOTAL_INVERSION
ACCIONES	51500.00
BONOS	50000.00
F_INDEXADO	28500.00

681 -- 29 Consulta las acciones con una rentabilidad YTD superior al promedio de rentabilidad YTD de todas las acciones.

```
682 * SELECT A.*
683 FROM ACCIONES A
684 WHERE A.RENTABILIDADYTD > (
685     SELECT AVG(A2.RENTABILIDADYTD)
686     FROM ACCIONES A2
687 );
```

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:

ISIN	COD_PRODUCTO	NOMBRE_EMPRESA	TICKER	CANTIDAD	PRECIO_COMPRA	PRECIO_ACTUAL	CAPITALIZACION	RENTABILIDADYTD	DIVIDENDOSYTD
US0378331005	1	Apple Inc.	AAPL	100	145.00	155.00	2500000000000.00	8.00	1.50
US38259P5089	10	Alphabet Inc.	GOOGL	50	2200.00	2300.00	12000000000000.00	7.50	0.00
US5949181045	7	Microsoft Corporation	MSFT	150	250.00	280.00	20000000000000.00	10.00	2.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

689 -- 30 Consulta el número total de transacciones (ingresos y retiradas) por cada cliente.

```
690 * SELECT C.COD_CLIENTE, C.DNI, COUNT(T.COD_TRANSACCION) AS NUM_TRANSACCIONES
691 FROM CLIENTES C
692 INNER JOIN TRANSACCION T ON C.COD_CLIENTE = T.COD_CLIENTE
693 GROUP BY C.COD_CLIENTE, C.DNI;
```

Result Grid Filter Rows: Export: Wrap Cell Content:

COD_CLIENTE	DNI	NUM_TRANSACCIONES
1	12345678A	2
2	23456789B	2
3	34567890C	1
4	45678901D	1
5	56789012E	1
6	67890123F	2
7	78901234G	1
8	89012345H	2



3. Diseño físico

3.1. Capturas procedimientos

```

647 -- PROCEDIMIENTO 1- El siguiente procedimiento almacenado buscará una cuenta corriente por IBAN y realizará un depósito en la cuenta.
    Si el IBAN no se encuentra, el procedimiento almacenado mostrará un mensaje de error. Si el monto del depósito es negativo, también
    mostrará un mensaje de error. De lo contrario, actualizará la cuenta bancaria con el nuevo saldo y registrará la transacción.
648 DELIMITER //
649 DROP PROCEDURE IF EXISTS INGRESAR_DINERO;
650 //CREATE PROCEDURE INGRESAR_DINERO(P_IBAN VARCHAR(34),P_CANTIDAD DECIMAL(10,2))
651 BEGIN
652     DECLARE SALDO_ACTUAL DECIMAL(10,2);
653
654     -- Introduzco el saldo actual en la variable SALDO_ACTUAL
655     SELECT SALDO INTO SALDO_ACTUAL
656     FROM CUENTA_CORRIENTE
657     WHERE IBAN = P_IBAN;
658
659     -- Verificar si la cantidad es negativa
660     IF P_CANTIDAD < 0 THEN
661         SELECT 'Error: la cantidad no puede ser negativa.' AS 'ERROR';
662     -- Verificar si la cuenta existe
663     ELSEIF SALDO_ACTUAL IS NULL THEN
664         SELECT 'Error: no se encontró la cuenta.' AS 'ERROR';
665     ELSE
666         -- Todo está bien, realizar la transacción
667         UPDATE CUENTA_CORRIENTE
668         SET SALDO = SALDO + P_CANTIDAD
669         WHERE IBAN = P_IBAN;
670
671         SELECT CONCAT_WS(" ", 'Depósito realizado con éxito.', 'El saldo actualizado es', (SALDO_ACTUAL+P_CANTIDAD), 'EUROS.') AS 'MENSAJE';
672     END IF;
673 END //
674 DELIMITER ;

```

```

676 -- MENSAJE ERROR SALDO NEGATIVO
677 CALL INGRESAR_DINERO('ES0012345678001234567890',-100);
678

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

ERROR
Error: la cantidad no puede ser negativa.

```

682 -- INGRESO CORRECTO
683 CALL INGRESAR_DINERO('ES0012345678001234567890',100);
684

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

MENSAJE
Depósito realizado con éxito. El saldo actualizad...

```

685 -- COMPROBACION
686 SELECT * FROM CUENTA_CORRIENTE
687 WHERE IBAN = 'ES0012345678001234567890';
688

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

IBAN	SALDO	TIPO
ES0012345678001234567890	600.00	NORMAL
NULL	NULL	NULL

```

689 -- PROCEDIMIENTO 2- El siguiente procedimiento almacenado buscará una cuenta corriente por IBAN y realizará un retiro en la cuenta. Si
    el IBAN no se encuentra, el procedimiento almacenado mostrará un mensaje de error. Si el monto del depósito es mayor al saldo ,
    también mostrará un mensaje de error. De lo contrario, actualizará la cuenta bancaria con el nuevo saldo y registrará la transacción.
690
691 DELIMITER //
692 • DROP PROCEDURE IF EXISTS RETIRAR_DINERO;
693 • //CREATE PROCEDURE RETIRAR_DINERO(P_IBAN VARCHAR(34),P_CANTIDAD DECIMAL(10,2))
694 • BEGIN
695     DECLARE SALDO_ACTUAL DECIMAL(10,2);
696
697     -- Introduzco el saldo actual en la variable SALDO_ACTUAL
698     SELECT SALDO INTO SALDO_ACTUAL
699     FROM CUENTA_CORRIENTE
700     WHERE IBAN = P_IBAN;
701
702     -- Verificar si la cantidad es negativa
703     • IF P_CANTIDAD > SALDO_ACTUAL THEN
704         SELECT 'Error: no existen fondos suficientes.' AS 'ERROR';
705     -- Verificar si la cuenta existe
706     ELSEIF SALDO_ACTUAL IS NULL THEN
707         SELECT 'Error: no se encontró la cuenta.' AS 'ERROR';
708     ELSE
709         -- Todo está bien, realizar la transacción
710         UPDATE CUENTA_CORRIENTE
711         SET SALDO = SALDO - P_CANTIDAD
712         WHERE IBAN = P_IBAN;
713
714         SELECT CONCAT_WS(" ", 'Retiro realizado con éxito.', 'El saldo actualizado es', (SALDO_ACTUAL-P_CANTIDAD), 'EUROS.') AS 'MENSAJE';
715     END IF;
716 • END //
717 DELIMITER ;
718
719 • -- MENSAJE ERROR SALDO INCORRECTO
720 CALL RETIRAR_DINERO('ES0012345678001234567890',1400);
721
722 -- MENSAJE ERROR CUENTA NO ENCONTRADA
723 • CALL RETIRAR_DINERO('999912345678001234567890',100);
724
725 -- RETIRO CORRECTO
726 • CALL RETIRAR_DINERO('ES0012345678001234567890',100);
727
728 -- COMPROBACION
729 • SELECT * FROM CUENTA_CORRIENTE
730 WHERE IBAN = 'ES0012345678001234567890';
731

```

```

719 -- MENSAJE ERROR SALDO INCORRECTO
720 CALL RETIRAR_DINERO('ES0012345678001234567890',1400);
721

```

Result Grid Filter Rows: Export: Wrap Cell Content:

ERROR

Error: no existen fondos suficientes.

```

722 -- MENSAJE ERROR CUENTA NO ENCONTRADA
723 CALL RETIRAR_DINERO('999912345678001234567890',100);
724

```

Result Grid Filter Rows: Export: Wrap Cell Content:

ERROR

Error: no se encontró la cuenta.

```

725 -- RETIRO CORRECTO
726 CALL RETIRAR_DINERO('ES0012345678001234567890',100);
727

```

Result Grid Filter Rows: Export: Wrap Cell Content:

MENSAJE

Retiro realizado con éxito. El saldo actualizado ...

```

728 -- COMPROBACION
729 SELECT * FROM CUENTA_CORRIENTE
730 WHERE IBAN = 'ES0012345678001234567890';

```

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:

	IBAN	SALDO	TIPO
▶	ES0012345678001234567890	400.00	NORMAL
*	NULL	NULL	NULL

```

733 -- PROCEDIMIENTO 3 (CURSOR)-Este procedimiento se encarga de crear tres nuevas tablas: Normal, Oro y Platinum. Estas tablas se
    utilizarán para reorganizar las cuentas corrientes existentes en la base de datos en función del tipo de cuenta.
734 DELIMITER //
735 • DROP PROCEDURE IF EXISTS PROCEDIMIENTO3;
736 • // CREATE PROCEDURE PROCEDIMIENTO3()
737 BEGIN
738     -- DECLARO VARIABLES
739     DECLARE FIN_TABLA BOOLEAN DEFAULT FALSE;
740     DECLARE V_IBAN VARCHAR(34);
741     DECLARE V_SALDO DECIMAL(10,2);
742     DECLARE V_TIPO VARCHAR(50);
743
744     -- DECLARO CURSOR
745     DECLARE C CURSOR FOR SELECT IBAN,SALDO,TIPO FROM CUENTA_CORRIENTE;
746
747     -- DECLARO HANDLER
748     DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET FIN_TABLA = TRUE;
749
750     -- CREO TABLAS
751     DROP TABLE IF EXISTS NORMAL;
752     CREATE TABLE NORMAL(IBAN VARCHAR(34),SALDO DECIMAL(10, 2),TIPO VARCHAR(50));
753     DROP TABLE IF EXISTS ORO;
754     CREATE TABLE ORO(IBAN VARCHAR(34),SALDO DECIMAL(10, 2),TIPO VARCHAR(50));
755     DROP TABLE IF EXISTS PLATINUM;
756     CREATE TABLE PLATINUM(IBAN VARCHAR(34),SALDO DECIMAL(10, 2),TIPO VARCHAR(50));
757
758     -- ABRO CURSOR
759     OPEN C;
760
761     -- RECORRO CON CURSOR
762     WHILE (FIN_TABLA=FALSE) DO
763         FETCH C INTO V_IBAN,V_SALDO,V_TIPO;
764         IF(FIN_TABLA=FALSE) THEN
765             IF(V_TIPO = 'NORMAL') THEN
766                 INSERT INTO NORMAL VALUES (V_IBAN,V_SALDO,V_TIPO);
767             ELSEIF (V_TIPO = 'ORO') THEN
768                 INSERT INTO ORO VALUES (V_IBAN,V_SALDO,V_TIPO);
769             ELSEIF (V_TIPO = 'PLATINUM') THEN
770                 INSERT INTO PLATINUM VALUES (V_IBAN,V_SALDO,V_TIPO);
771             END IF;
772         END IF;
773     END WHILE;
774
775     -- CIERRO CURSOR
776     CLOSE C;
777 END; //
778 DELIMITER ;
780 • CALL PROCEDIMIENTO3();
781
782 • SELECT * FROM NORMAL;
783 • SELECT * FROM ORO;
784 • SELECT * FROM PLATINUM;
785

```


```
780 • CALL PROCEDIMIENTO3();
781
782 • SELECT * FROM NORMAL;
```

Result Grid   Filter Rows: Export:  Wrap Cell Content: 

	IBAN	SALDO	TIPO
▶	ES0012345678001234567890	400.00	NORMAL
	ES0012345678067890123456	600.00	NORMAL





```
783 • SELECT * FROM ORO;
```

Result Grid   Filter Rows: Export:  Wrap Cell Content: 

	IBAN	SALDO	TIPO
▶	ES0012345678023456789012	200.00	ORO



```
783 • SELECT * FROM ORO;
```

Result Grid   Filter Rows: Export:  Wrap Cell Content: 

	IBAN	SALDO	TIPO
▶	ES0012345678023456789012	200.00	ORO





3. Diseño físico

3.1. Capturas funciones


```

791 -- FUNCION 1- Esta función recibe el código del cliente y retorna el saldo total de todas sus cuentas corrientes.
792 DELIMITER //
793 DROP FUNCTION IF EXISTS SALDO_TOTAL_CLIENTE;
794 //CREATE FUNCTION SALDO_TOTAL_CLIENTE(P_CODCLIENTE INT) RETURNS DECIMAL(10, 2)
795 DETERMINISTIC
796 BEGIN
797     DECLARE SALDO_TOTAL DECIMAL(10, 2);
798     SELECT SUM(SALDO) INTO SALDO_TOTAL
799     FROM CUENTA_CORRIENTE CC
800     INNER JOIN PERTENECEN_CLI_CB PCCB ON CC.IBAN = PCCB.IBAN
801     WHERE PCCB.COD_CLIENTE = P_CODCLIENTE;
802
803     RETURN SALDO_TOTAL;
804 END//
805 DELIMITER ;
806
807 SELECT SALDO_TOTAL_CLIENTE(1) AS 'SALDO TOTAL';

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

SALDO TOTAL
400.00

Result
Grid

```

809 -- FUNCION 2 Esta función recibe como parámetros el código y DNI del cliente y un rango de fechas (fecha de inicio y fecha de fin). La
función luego cuenta el número de transacciones realizadas por ese cliente durante el rango de fechas especificado y devuelve este
número.
810 DELIMITER //
811 DROP FUNCTION IF EXISTS CONTAR_TRANSACCIONES;
812 //CREATE FUNCTION CONTAR_TRANSACCIONES(P_DNI CHAR(9), FECHA_INICIO DATE, FECHA_FIN DATE) RETURNS INT
813 DETERMINISTIC
814 BEGIN
815     DECLARE TOTAL_TRANSACCIONES INT;
816
817     SELECT COUNT(*)
818     INTO TOTAL_TRANSACCIONES
819     FROM TRANSACCION
820     WHERE DNI = P_DNI
821     AND FECHA BETWEEN FECHA_INICIO AND FECHA_FIN;
822
823     RETURN TOTAL_TRANSACCIONES;
824 END //
825 DELIMITER ;
827 SELECT CONTAR_TRANSACCIONES('12345678A', '2023-01-11', '2023-02-10') AS TRANSACCIONES;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

TRANSACCIONES
2

Result
Grid



3. Diseño físico

3.1. Capturas triggers

```

834 -- TIGRETÓN 1 - Este tigrón actualizará la rentabilidad YTD cada vez que se actualice el precio actual de las acciones.
835 DELIMITER //
836 * DROP TRIGGER IF EXISTS ACTUALIZAR_RENTABILIDAD_YTD;
837 * // CREATE TRIGGER ACTUALIZAR_RENTABILIDAD_YTD
838 BEFORE UPDATE ON ACCIONES
839 FOR EACH ROW
840 BEGIN
841     IF NEW.PRECIO_ACTUAL != OLD.PRECIO_ACTUAL THEN
842         SET NEW.RENTABILIDADYTD = ((NEW.PRECIO_ACTUAL * 100) / NEW.PRECIO_COMPRA) - 100;
843     END IF;
844 END//
845 DELIMITER ;
847 * SELECT * FROM ACCIONES;
848
849 * UPDATE ACCIONES
850 SET PRECIO_ACTUAL = PRECIO_ACTUAL + 23
851 WHERE ISIN = 'US0378331005';

```

Result Grid										
Filter Rows:										
Edit: Export/Import: Wrap Cell Content:										
	ISIN	COD_PRODUCTO	NOMBRE_EMPRESA	TICKER	CANTIDAD	PRECIO_COMPRA	PRECIO_ACTUAL	CAPITALIZACION	RENTABILIDADYTD	DIVIDENDOSYTD
▶	US0231351067	4	Amazon.com, Inc.	AMZN	20	3300.00	3400.00	1500000000000.00	6.00	0.00
	US0378331005	1	Apple Inc.	AAPL	100	145.00	178.00	2500000000000.00	22.76	1.50
	US30303M1027	13	Facebook, Inc.	FB	200	330.00	350.00	900000000000.00	5.00	0.50
	US38259P5089	10	Alphabet Inc.	GOOGL	50	2200.00	2300.00	1200000000000.00	7.50	0.00
	US5949181045	7	Microsoft Corporation	MSFT	150	250.00	280.00	2000000000000.00	10.00	2.00
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```

853 -- TIGRETÓN 2 - Este tigrón actualizará la tabla transacciones insertando una nueva fila cada vez que haya una retirada o ingreso de
dinero en una cuenta corriente.
854 DELIMITER //
855 DROP TRIGGER IF EXISTS ACTUALIZAR_TRANSACCIONES;
856 //CREATE TRIGGER ACTUALIZAR_TRANSACCIONES
857 AFTER UPDATE ON CUENTA_CORRIENTE
858 FOR EACH ROW
859 BEGIN
860     DECLARE V_COD_TRANSACCION INT;
861     DECLARE V_COD_CLIENTE INT;
862     DECLARE V_DNI CHAR(9);
863
864     -- Damos valor a la variable V_COD_TRANSACCION
865     SELECT (MAX(COD_TRANSACCION)+1) INTO V_COD_TRANSACCION FROM TRANSACCION;
866     -- Damos valor a la variable V_COD_CLIENTE
867     SELECT COD_CLIENTE INTO V_COD_CLIENTE FROM CUENTAS_BANCARIAS WHERE IBAN = NEW.IBAN;
868     -- Damos valor a la variable V_DNI
869     SELECT DNI INTO V_DNI FROM CUENTAS_BANCARIAS WHERE IBAN = NEW.IBAN;
870
871     -- Comprobamos si el saldo ha disminuido, lo que implica una retirada
872     IF NEW.SALDO < OLD.SALDO THEN
873         INSERT INTO TRANSACCION VALUES ( V_COD_TRANSACCION,V_COD_CLIENTE,V_DNI,NEW.IBAN,(OLD.SALDO - NEW.SALDO),CURRENT_DATE(),'RETIRADA' );
874     END IF;
875
876     -- Comprobamos si el saldo ha aumentado, lo que implica un ingreso
877     IF NEW.SALDO > OLD.SALDO THEN
878         INSERT INTO TRANSACCION VALUES (V_COD_TRANSACCION,V_COD_CLIENTE,V_DNI,NEW.IBAN,(NEW.SALDO - OLD.SALDO),CURRENT_DATE(),'INGRESO');
879     END IF;
880 END//
881 DELIMITER ;
882 -- INGRESAMOS DINERO USANDO NUESTRO PROCEDIMIENTO Y SEGUIDAMENTE COMPROBAMOS SI SE HA ACTUALIZADO LA TABLA TRANSACCION
883 CALL INGRESAR_DINERO('ES0012345678001234567890',100);
884

```

Result Grid Filter Rows: Export: Wrap Cell Content:

MENSAJE

Depósito realizado con éxito. El saldo actualizad...

886 SELECT * FROM TRANSACCION;

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:

	COD_TRANSACCION	COD_CLIENTE	DNI	IBAN	CANTIDAD	FECHA	TIPO
1	1	1	12345678A	ES0012345678001234567890	500.00	2023-01-11	INGRESO
2	2	2	23456789B	ES0012345678012345678901	1000.00	2023-02-20	INGRESO
3	3	3	34567890C	ES0012345678023456789012	300.00	2023-02-28	RETIRADA
4	4	4	45678901D	ES0012345678034567890123	750.00	2023-03-15	INGRESO
5	5	5	56789012E	ES0012345678045678901234	200.00	2023-03-01	RETIRADA
6	6	6	67890123F	ES0012345678056789012345	600.00	2023-03-18	INGRESO
7	7	7	78901234G	ES0012345678067890123456	900.00	2023-03-05	RETIRADA



inverza



Tu inversión inteligente

Walter Martín Lopes

Samuel Fernández Díaz

Alberto Rivero Núñez