**ALUMNO: César Blanco Fernández**

**PRIMERA PARTE:**

* 1. Hacer el estudio de:

1. STG\_FACTURAS\_FCT con conclusiones
2. Partimos de la base de que esta tabla ya está creada en la BBDD Stage, de tal manera que la información cargada procedente de la fuente:

DROP TABLE IF EXISTS STG\_PRODUCTOS\_CRM;

CREATE TABLE STG\_FACTURAS\_FCT AS

SELECT \* FROM FACTURADOR.FACTURAS;

ANALYZE TABLE STG\_FACTURAS\_FCT;

1. Revisamos el contenido de la tabla:

USE STAGE;

SELECT COUNT(\*) AS TOTAL\_REGITROS

, SUM(CASE WHEN LENGTH(TRIM(BILL\_REF\_NO)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_BILL\_REF\_NO

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(BILL\_REF\_NO)) <> 0 THEN BILL\_REF\_NO ELSE 0 END) AS TOTAL\_BILL\_REF\_NO\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(CUSTOMER\_ID)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_CUSTOMER\_ID

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(CUSTOMER\_ID)) <> 0 THEN CUSTOMER\_ID ELSE 0 END) AS TOTAL\_CUSTOMER\_ID\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(START\_DATE)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_START\_DATE

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(START\_DATE)) <> 0 THEN START\_DATE ELSE 0 END) AS TOTAL\_START\_DATE\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(END\_DATE)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_END\_DATE

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(END\_DATE)) <> 0 THEN END\_DATE ELSE 0 END) AS TOTAL\_END\_DATE\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(STATEMENT\_DATE)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_ESTATEMENT\_DATE

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(STATEMENT\_DATE)) <> 0 THEN STATEMENT\_DATE ELSE 0 END) AS TOTAL\_STATEMENT\_DATE\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(PAYMENT\_DATE)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_PAYMENT\_DATE

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(PAYMENT\_DATE)) <> 0 THEN PAYMENT\_DATE ELSE 0 END) AS TOTAL\_PAYMENT\_DATE\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(BILL\_CYCLE)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_BILL\_CYCLE

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(BILL\_CYCLE)) <> 0 THEN BILL\_CYCLE ELSE 0 END) AS TOTAL\_BILL\_CYCLE\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(AMOUNT)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_AMOUNT

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(AMOUNT)) <> 0 THEN AMOUNT ELSE 0 END) AS TOTAL\_AMOUNT\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(BILL\_METHOD)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_BILL\_METHOD

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(BILL\_METHOD)) <> 0 THEN BILL\_METHOD ELSE 0 END) AS TOTAL\_BILL\_METHOD\_DISTINTO

FROM STAGE.STG\_FACTURAS\_FCT;

# Visualización del contenido:

SELECT COUNT(\*) AS TOTAL\_REGITROS

, SUM(CASE WHEN LENGTH(TRIM(BILL\_CYCLE)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_BILL\_CYCLE

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(BILL\_CYCLE)) <> 0 THEN BILL\_CYCLE ELSE 0 END) AS TOTAL\_BILL\_CYCLE\_DISTINTOS

, BILL\_CYCLE, LENGTH(BILL\_CYCLE)

FROM STAGE.STG\_FACTURAS\_FCT

GROUP BY BILL\_CYCLE

ORDER BY LENGTH(BILL\_CYCLE) DESC;

SELECT COUNT(\*) AS TOTAL\_REGITROS

, SUM(CASE WHEN LENGTH(TRIM(AMOUNT)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_AMOUNT

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(AMOUNT)) <> 0 THEN AMOUNT ELSE 0 END) AS TOTAL\_AMOUNT\_DISTINTOS

, AMOUNT

FROM STAGE.STG\_FACTURAS\_FCT

GROUP BY AMOUNT;

SELECT COUNT(\*) AS TOTAL\_REGITROS

, SUM(CASE WHEN LENGTH(TRIM(BILL\_METHOD)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_BILL\_METHOD

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(BILL\_METHOD)) <> 0 THEN BILL\_METHOD ELSE 0 END) AS TOTAL\_BILL\_METHOD\_DISTINTOS

, BILL\_METHOD, LENGTH(BILL\_METHOD)

FROM STAGE.STG\_FACTURAS\_FCT

GROUP BY BILL\_METHOD

ORDER BY LENGTH(BILL\_METHOD) DESC;

Contenido de los campos:

|  |  |  |
| --- | --- | --- |
| **CAMPOS** | **VALOR** | **OBSERVACIÓN** |
| TOTAL\_REGISTROS | 420000 | Número total de registros |
| TOTAL\_BILL\_REF\_NO | 420000 | Identifcador de cada registro, coincide con el nº total de registros |
| TOTAL\_BILL\_REF\_NO\_DISTINTOS | 420000 | Ninguno se repite |
| TOTAL\_COSTUMER\_ID | 420000 | Identificador de cliente (Viene de la table de clientes) |
| TOTAL\_COSTUMER\_ID\_DISTINTOS | 20000 | Se repiten. Puede existir más de una factura asociada a un mismo cliente |
| TOTAL\_START\_DATE | 420000 | Fecha de comienzo |
| TOTAL\_START\_DATE\_DISTINTOS | 40 | Se repiten |
| TOTAL\_END\_DATE | 420000 | Fecha del final |
| TOTAL\_END\_DATE\_DISTINTOS | 20 | Se repiten |
| TOTAL\_STATEMENT\_DATE | 420000 | Fecha |
| TOTAL\_STATEMENT\_DATE\_DISTINTOS | 40 | Se repiten |
| TOTAL\_PAYMENT\_DATE | 420000 | Fecha de pagos total |
| TOTAL\_PAYMENT\_DATE\_DISTINTOS | 400 | Se repiten |
| TOTAL\_BILL\_CYCLE | 420000 | Coincide con el nº total de registros 🡪 No hay ninguna vacía |
| TOTAL\_BILL\_CYCLE\_DISTINTOS | 2 | Se repite con 2 posibles valores 🡪 Se puede usar como posible dimensión |
| TOTAL\_AMOUNT | 420000 | Coincide con el nº total de registros 🡪 No hay ninguna vacía |
| TOTAL\_AMOUNT\_DISTINTOS | 5604 | Se repiten (son valores numéricos) |
| TOTAL\_BILL\_METHOD | 420000 | Coincide con el nº total de registros 🡪 No hay ninguna vacía. Método de pago |
| TOTAL\_BILL\_METHOD\_DISTINTOS | 3 | Se repite con 3 posibles valores 🡪 Se puede usar como posible dimensión |

1. Conclusiones:
2. El campo que relaciona la tabla de Facturas con la tabla de clientes es: CUSTOMER\_ID
3. La creación de la tabla de Facturas en ODS tendrá 2 dimensiones asociadas:

* ODS\_DM\_CICLOS\_FACTURACION asociada al campo BILL\_CYCLE
* ODS\_DM\_METODOS\_PAGO asociada al campo BILL\_METHOD

La motivación que nos ha llevado crear en la tabla de Facturas 2 dimensiones son:

* El contenido de los 2 campos mencionados es texto que se repite, por lo que crearemos dos dimensiones, de tal manera que el ID de cada dimensión sustituirá al valor de estos campos
* El mantenimiento automático de las tablas de dimensiones es fácil
* No tiene sentido volcar a otras tablas de dimensiones por ser numéricas: los campos de Fechas, IDs y otros campos que sean sólo numéricos.

1. STG\_CONTACTOS\_IVR con conclusiones
2. Partimos de la base de que esta tabla ya está creada en la BBDD Stage, de tal manera que la información cargada procedente de la fuente:

CREATE TABLE STG\_CONTACTOS\_IVR AS

SELECT \* FROM IVR.CONTACTOS;

ANALYZE TABLE STG\_CONTACTOS\_IVR;

1. Revisamos el contenido de la tabla:

SELECT COUNT(\*) AS TOTAL\_REGITROS

, SUM(CASE WHEN LENGTH(TRIM(ID)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_ID

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(ID)) <> 0 THEN ID ELSE 0 END) AS TOTAL\_ID\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(PHONE\_NUMBER)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_PHONE\_NUMBER

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(PHONE\_NUMBER)) <> 0 THEN PHONE\_NUMBER ELSE 0 END) AS TOTAL\_PHONE\_NUMBER\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(START\_DATETIME)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_START\_DATETIME

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(START\_DATETIME)) <> 0 THEN START\_DATETIME ELSE 0 END) AS TOTAL\_START\_DATETIME\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(END\_DATETIME)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_END\_DATETIME

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(END\_DATETIME)) <> 0 THEN END\_DATETIME ELSE 0 END) AS TOTAL\_END\_DATETIME\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(SERVICE)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_SERVICE

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(SERVICE)) <> 0 THEN SERVICE ELSE 0 END) AS TOTAL\_SERVICE\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(FLG\_TRANSFER)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_FLG\_TRANSFER

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(FLG\_TRANSFER)) <> 0 THEN FLG\_TRANSFER ELSE 0 END) AS TOTAL\_FLG\_TRANSFER\_DISTINTOS

, SUM(CASE WHEN LENGTH(TRIM(AGENT)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_AGENT

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(AGENT)) <> 0 THEN AGENT ELSE 0 END) AS TOTAL\_AGENT\_DISTINTOS

FROM STAGE.STG\_CONTACTOS\_IVR;

# Visualización del contenido

SELECT COUNT(\*) AS TOTAL\_REGITROS

, SUM(CASE WHEN LENGTH(TRIM(AGENT)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_AGENT

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(AGENT)) <> 0 THEN AGENT ELSE 0 END) AS TOTAL\_AGENT\_DISTINTOS

, AGENT

, LENGTH(AGENT)

FROM STAGE.STG\_CONTACTOS\_IVR

GROUP BY AGENT

ORDER BY LENGTH(AGENT) DESC;

SELECT COUNT(\*) AS TOTAL\_REGITROS

, SUM(CASE WHEN LENGTH(TRIM(FLG\_TRANSFER)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_FLG\_TRANSFER

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(FLG\_TRANSFER)) <> 0 THEN FLG\_TRANSFER ELSE 0 END) AS TOTAL\_FLG\_TRANSFER\_DISTINTOS

, FLG\_TRANSFER

FROM STAGE.STG\_CONTACTOS\_IVR

GROUP BY FLG\_TRANSFER;

SELECT COUNT(\*) AS TOTAL\_REGITROS

, SUM(CASE WHEN LENGTH(TRIM(SERVICE)) <> 0 THEN 1 ELSE 0 END) AS TOTAL\_SERVICE

, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(SERVICE)) <> 0 THEN SERVICE ELSE 0 END) AS TOTAL\_SERVICE\_DISTINTOS

, SERVICE, LENGTH(SERVICE)

FROM STAGE.STG\_CONTACTOS\_IVR

GROUP BY SERVICE

ORDER BY LENGTH(SERVICE) DESC;

Contenido de los campos:

|  |  |  |
| --- | --- | --- |
| **CAMPOS** | **VALOR** | **OBSERVACIÓN** |
| TOTAL\_REGISTROS | 202717 | Número total de registros |
| TOTAL\_ID | 202717 | Identifcador |
| TOTAL\_ID\_DISTINTOS | 150000 | Se repiten identificadores |
| TOTAL\_PHONE\_NUMBER | 185018 | Número de teléfono. Hay nº de teléfono vacíos |
| TOTAL\_PHONE\_NUMBER\_DISTINTOS | 18226 | Se repiten números de teléfono |
| TOTAL\_START\_DATETIME | 202717 | Fecha |
| TOTAL\_START\_DATETIME\_DISTINTOS | 201098 | Se repiten |
| TOTAL\_END\_DATETIME | 186535 | Fecha |
| TOTAL\_END\_DATETIME\_DISTINTOS | 183678 | Se repiten |
| TOTAL\_SERVICE | 202502 | Servicio. Hay algunos que están vacíos |
| TOTAL\_SERVICE\_DISTINTOS | 7 | Se repiten 🡪 Posible campo para dimensión |
| TOTAL\_FLG\_TRANSFER | 202717 | Campo booleano |
| TOTAL\_FLG\_TRANSFER\_DISTINTOS | 2 | Se repiten 🡪 Posible campo para dimensión |
| TOTAL\_AGENT | 194739 | Agente. Hay algunos que están vacíos |
| TOTAL\_AGENT\_DISTINTOS | 594 | Se repiten 🡪 Posible campo para dimensión |

1. Conclusiones:
2. La tabla de ‘Contactos’ está relacionada con la tabla de ‘Productos’ a través del campo: AGENT.
3. En ODS la tabla de ‘Contactos’ va a tener 3 dimensiones:

* ODS\_DM\_AGENTES asociado al campo AGENT
* ODS\_DM\_FLG\_TRANSFER asociado al campo FLG\_TRANSFER
* ODS\_DM\_DEPARTAMENTOS asociado al campo SERVICE

La motivación que nos ha llevado crear en la tabla de ‘Contactos’ 3 dimensiones son:

* El contenido de los 3 campos mencionados es texto que se repite, por lo que crearemos tres dimensiones, de tal manera que el ID de cada dimensión sustituirá al valor de estos campos mejorando el rendimiento de procesamiento.
* El campo booleano lo sacamos a otra dimensión, porque al sustituirlo por valores 0/1 tendrá mejor rendimiento que con el texto: True / False
* No tiene sentido volcar a otras tablas de dimensiones por ser numéricas: los campos de Fechas e IDs .
  1. Crear el modelo, las tablas, las FK y poblar los modelos:

1. FACTURAS

*El diagrama planteado es un modo, pero puedes plantear uno diferente*

*ver: Modelos\_Facturas\_y\_Llamadas.mwb*

1. Creación de las tablas del modelo de Facturas:

|  |  |  |  |
| --- | --- | --- | --- |
| **BBDD** | **Tabla** | **Campos** | **Tipo** |
| ODS | ODS\_HC\_CLIENTES | ID\_CLIENTE | INT |
| NOMBRE\_CLIENTE | VARCHAR(512) |
| APELLIDOS\_CLIENTE | VARCHAR(512) |
| NUMDOC\_CLIENTE | VARCHAR(24) |
| ID\_SEXO | INT |
| ID\_DIRECCION\_CLIENTE | INT |
| TELEFONO\_CLIENTE | BIGINT |
| EMAIL | VARCHAR(512) |
| FC\_NACIMIENTO | DATE |
| ID\_PROFESION | INT |
| ID\_COMPANYA | INT |
| FC\_INSERT | DATETIME |
| FC\_MODIFICACION | DATETIME |
|  |  |  |
| ODS\_HC\_FACTURAS | ID\_FACTURA | INT |
| ID\_CLIENTE | INT |
| FC\_INICIO | DATETIME |
| FC\_FIN | DATETIME |
| FC\_ESTADO | DATETIME |
| FC\_PAGO | DATETIME |
| ID\_CICLO\_FACTURACION | INT |
| ID\_METODO\_PAGO | INT |
| CANTIDAD | FLOAT(10,2) |
| FC\_INSERT | DATETIME |
| FC\_MODIFICACION | DATETIME |
|  |  |  |
| ODS\_DM\_METODOS\_PAGO | ID\_METODO\_PAGO | INT |
| DE\_METODO\_PAGO | VARCHAR(512) |
| FC\_INSERT | DATETIME |
| FC\_MODIFICACION | DATETIME |
|  |  |  |
| ODS\_DM\_CICLOS\_FACTURACION | ID\_CICLO\_FACTURACION | INT |
| DE\_CICLO\_FACTURACION | VARCHAR(512) |
| FC\_INSERT | DATETIME |
| FC\_MODIFICACION | DATETIME |

# CREACIÓN DE TABLAS DEL MODELO DE FACTURAS

USE ODS;

#ALTER TABLE ODS\_HC\_CLIENTES DROP FOREIGN KEY fk\_cl

SET FOREIGN\_KEY\_CHECKS=0;

DROP TABLE IF EXISTS ODS\_HC\_CLIENTES;

CREATE TABLE ODS\_HC\_CLIENTES

(ID\_CLIENTE INT NOT NULL PRIMARY KEY

, NOMBRE\_CLIENTE VARCHAR(512)

, APELLIDOS\_CLIENTE VARCHAR(512)

, NUMDOC\_CLIENTE VARCHAR(24)

, ID\_SEXO INT

, ID\_DIRECCION\_CLIENTE INT

, TELEFONO\_CLIENTE BIGINT

, EMAIL VARCHAR(512)

, FC\_NACIMIENTO DATE

, ID\_PROFESION INT

, ID\_COMPANYA INT

, FC\_INSERT DATETIME

, FC\_MODIFICACION DATETIME);

DROP TABLE IF EXISTS ODS\_HC\_FACTURAS;

CREATE TABLE ODS\_HC\_FACTURAS

(ID\_FACTURA INT NOT NULL PRIMARY KEY

, ID\_CLIENTE INT

, FC\_INICIO DATETIME

, FC\_FIN DATETIME

, FC\_ESTADO DATETIME

, FC\_PAGO DATETIME

, ID\_CICLO\_FACTURACION INT

, ID\_METODO\_PAGO INT

, CANTIDAD FLOAT(10,2)

, FC\_INSERT DATETIME

, FC\_MODIFICACION DATETIME);

DROP TABLE IF EXISTS ODS\_DM\_METODOS\_PAGO;

CREATE TABLE ODS\_DM\_METODOS\_PAGO

(ID\_METODO\_PAGO INT NOT NULL AUTO\_INCREMENT PRIMARY KEY

, DE\_METODO\_PAGO VARCHAR(512)

, FC\_INSERT DATETIME

, FC\_MODIFICACION DATETIME);

DROP TABLE IF EXISTS ODS\_DM\_CICLOS\_FACTURACION;

CREATE TABLE ODS\_DM\_CICLOS\_FACTURACION

(ID\_CICLO\_FACTURACION INT NOT NULL AUTO\_INCREMENT PRIMARY KEY

, DE\_CICLO\_FACTURACION VARCHAR(512)

, FC\_INSERT DATETIME

, FC\_MODIFICACION DATETIME);

1. Creación de FK asociadas al del modelo de Facturas:

# CREACION FK MODELO DE FACTURAS

USE ODS;

ALTER TABLE ODS\_HC\_FACTURAS ADD INDEX fk\_fac\_cli\_idx (ID\_CLIENTE ASC);

ALTER TABLE ODS\_HC\_FACTURAS ADD CONSTRAINT fk\_fac\_cli FOREIGN KEY(ID\_CLIENTE)

REFERENCES ODS\_HC\_CLIENTES(ID\_CLIENTE);

ALTER TABLE ODS\_HC\_FACTURAS MODIFY COLUMN ID\_METODO\_PAGO INT(10);

ALTER TABLE ODS\_HC\_FACTURAS ADD INDEX fk\_fac\_met\_idx (ID\_METODO\_PAGO ASC);

ALTER TABLE ODS\_HC\_FACTURAS ADD CONSTRAINT fk\_fac\_met FOREIGN KEY(ID\_METODO\_PAGO)

REFERENCES ODS\_DM\_METODOS\_PAGO(ID\_METODO\_PAGO);

ALTER TABLE ODS\_HC\_FACTURAS MODIFY COLUMN ID\_CICLO\_FACTURACION INT(10);

ALTER TABLE ODS\_HC\_FACTURAS ADD INDEX fk\_fac\_cic\_idx (ID\_CICLO\_FACTURACION ASC);

ALTER TABLE ODS.ODS\_HC\_FACTURAS ADD CONSTRAINT fk\_fac\_cic\_dir FOREIGN KEY(ID\_CICLO\_FACTURACION)

REFERENCES ODS\_DM\_CICLOS\_FACTURACION(ID\_CICLO\_FACTURACION);

1. Poblamos el modelo de Facturas:

# POBLAMOS EL MODELO DE FACTURAS

# Carga Inicial:

USE ODS;

INSERT INTO ODS.ODS\_DM\_METODOS\_PAGO VALUES (-98, 'NO APLICA', NOW(), NOW());

INSERT INTO ODS.ODS\_DM\_METODOS\_PAGO VALUES (-99, 'DESCONOCIDO', NOW(), NOW());

COMMIT;

ANALYZE TABLE ODS\_DM\_METODOS\_PAGO;

INSERT INTO ODS.ODS\_DM\_CICLOS\_FACTURACION VALUES (-98, 'NO APLICA', NOW(), NOW());

INSERT INTO ODS.ODS\_DM\_CICLOS\_FACTURACION VALUES (-99, 'DESCONOCIDO', NOW(), NOW());

COMMIT;

ANALYZE TABLE ODS.DM\_CICLOS\_FACTURACION;

# OBSERVACIÓN: Ponemos índices negativos con la finalidad de identificar que esos # valores vienen de forma externa

# Actualizaciones periodicas

INSERT INTO ODS.ODS\_DM\_METODOS\_PAGO (DE\_METODO\_PAGO, FC\_INSERT, FC\_MODIFICACION)

SELECT DISTINCT UPPER(TRIM(STG\_FACTURAS\_FCT.BILL\_METHOD)), NOW(), NOW()

FROM STAGE.STG\_FACTURAS\_FCT

LEFT OUTER JOIN ODS.ODS\_DM\_METODOS\_PAGO

ON UPPER(TRIM(STG\_FACTURAS\_FCT.BILL\_METHOD)) = UPPER(TRIM(ODS.ODS\_DM\_METODOS\_PAGO.DE\_METODO\_PAGO))

WHERE STAGE.STG\_FACTURAS\_FCT.BILL\_METHOD IS NOT NULL

AND ODS.ODS\_DM\_METODOS\_PAGO.DE\_METODO\_PAGO IS NULL;

COMMIT;

INSERT INTO ODS.ODS\_DM\_CICLOS\_FACTURACION (DE\_CICLO\_FACTURACION, FC\_INSERT, FC\_MODIFICACION)

SELECT DISTINCT UPPER(TRIM(STG\_FACTURAS\_FCT.BILL\_CYCLE)), NOW(), NOW()

FROM STAGE.STG\_FACTURAS\_FCT

LEFT OUTER JOIN ODS.ODS\_DM\_CICLOS\_FACTURACION

ON UPPER(TRIM(STG\_FACTURAS\_FCT.BILL\_METHOD)) = UPPER(TRIM(ODS.ODS\_DM\_CICLOS\_FACTURACION.DE\_CICLO\_FACTURACION))

WHERE STAGE.STG\_FACTURAS\_FCT.BILL\_METHOD IS NOT NULL

AND ODS.ODS\_DM\_CICLOS\_FACTURACION.DE\_CICLO\_FACTURACION IS NULL;

COMMIT;

INSERT INTO ODS.ODS\_HC\_FACTURAS

(ID\_FACTURA,

ID\_CLIENTE

, FC\_INICIO

, FC\_FIN

, FC\_ESTADO

, FC\_PAGO

, ID\_CICLO\_FACTURACION

, ID\_METODO\_PAGO

, CANTIDAD

, FC\_INSERT

, FC\_MODIFICACION)

SELECT DISTINCT FACT.BILL\_REF\_NO

, FACT.CUSTOMER\_ID

, CASE WHEN LENGTH(TRIM(FACT.START\_DATE))<>0 THEN STR\_TO\_DATE(DATE\_FORMAT(FACT.START\_DATE,'%d/%m/%Y'),'%d/%m/%Y') ELSE STR\_TO\_DATE('31/12/9999','%d/%m/%Y') END AS FC\_INICIO

, CASE WHEN LENGTH(TRIM(FACT.END\_DATE))<>0 THEN STR\_TO\_DATE(DATE\_FORMAT(FACT.END\_DATE,'%d/%m/%Y'),'%d/%m/%Y') ELSE STR\_TO\_DATE('31/12/9999','%d/%m/%Y') END AS FC\_FIN

, CASE WHEN LENGTH(TRIM(FACT.STATEMENT\_DATE))<>0 THEN STR\_TO\_DATE(DATE\_FORMAT(FACT.STATEMENT\_DATE,'%d/%m/%Y'),'%d/%m/%Y') ELSE STR\_TO\_DATE('31/12/9999','%d/%m/%Y') END AS FC\_ESTADO

, CASE WHEN LENGTH(TRIM(FACT.PAYMENT\_DATE))<>0 THEN STR\_TO\_DATE(DATE\_FORMAT(FACT.PAYMENT\_DATE,'%d/%m/%Y'),'%d/%m/%Y') ELSE STR\_TO\_DATE('31/12/9999','%d/%m/%Y') END AS FC\_PAGO

, CICLOS.ID\_CICLO\_FACTURACION

, FACT.AMOUNT

, METODOS.ID\_METODO\_PAGO

, NOW()

, NOW()

FROM STAGE.STG\_FACTURAS\_FCT AS FACT

LEFT OUTER JOIN ODS.ODS\_HC\_FACTURAS ON

UPPER(TRIM(FACT.BILL\_REF\_NO)) = UPPER(TRIM(ODS.ODS\_HC\_FACTURAS.ID\_FACTURA))

INNER JOIN ODS.ODS\_DM\_CICLOS\_FACTURACION AS CICLOS ON

CASE WHEN LENGTH(TRIM(FACT.BILL\_CYCLE))<>0 THEN UPPER(TRIM(FACT.BILL\_CYCLE)) ELSE 'DESCONOCIDO' END = CICLOS.DE\_CICLO\_FACTURACION

INNER JOIN ODS.ODS\_DM\_METODOS\_PAGO AS METODOS ON

CASE WHEN LENGTH(TRIM(FACT.BILL\_METHOD))<>0 THEN UPPER(TRIM(FACT.BILL\_METHOD)) ELSE 'DESCONOCIDO' END = METODOS.DE\_METODO\_PAGO

WHERE FACT.BILL\_REF\_NO IS NOT NULL

AND ODS.ODS\_HC\_FACTURAS.ID\_FACTURA IS NULL;

COMMIT;

ANALYZE TABLE ODS.ODS\_HC\_FACTURAS;

1. LLAMADAS

*El diagrama planteado es un modo, pero puedes plantear uno diferente*

*ver: Modelos\_Facturas\_y\_Llamadas.mwb*

1. Creación de las tablas del modelo de Llamadas:

|  |  |  |  |
| --- | --- | --- | --- |
| **BBDD** | **Tablas** | **Campo** | **Tipo** |
| ODS | ODS\_HC\_LLAMADAS | ID\_LLAMADA | INT |
| ID\_IVR | INT |
| TELEFONO\_LLAMADA | BIGINT(20) |
| FC\_INICIO\_LLAMADA | DATETIME |
| FC\_FIN\_LLAMADA | DATETIME |
| ID\_DEPARTAMENTO\_CC | INT |
| ID\_FLG\_TRANSFERIDO | INT |
| ID\_AGENTE\_CC | INT |
| FC\_INSERT | DATETIME |
| FC\_MODIFICACION | DATETIME |
|  |  |  |
| ODS\_DM\_DEPARTAMENTOS\_CC | ID\_DEPARTAMENTO\_CC | INT |
| DE\_DEPARTAMENTO\_CC | VARCHAR(512) |
| FC\_INSERT | DATETIME |
| FC\_MODIFICACION | DATETIME |
|  |  |  |
| ODS\_DM\_AGENTES\_CC | ID\_AGENTE\_CC | INT |
| DE\_AGENTE\_CC | VARCHAR(512) |
| FC\_INSERT | DATETIME |
| FC\_MODIFICACION | DATETIME |
|  |  |  |
| ODS\_DM\_FLG\_TRANSFER | ID\_FLG\_TRANSFERIDO | INT |
| DE\_FLG\_TRANSFERIDO | VARCHAR(512) |
| FC\_INSERT | DATETIME |
| FC\_MODIFICACION | DATETIME |

#Creamos las tablas del modelo de LLAMADAS

# CREACIÓN DE TABLAS DEL MODELO DE LLAMADAS

USE ODS;

#ALTER TABLE ODS\_HC\_CLIENTES DROP FOREIGN KEY fk\_cl

SET FOREIGN\_KEY\_CHECKS=0;

DROP TABLE IF EXISTS ODS\_HC\_LLAMADAS;

CREATE TABLE ODS\_HC\_LLAMADAS

(ID\_LLAMADA INT NOT NULL AUTO\_INCREMENT PRIMARY KEY

, ID\_IVR INT

, TELEFONO\_LLAMADA BIGINT(20)

/\*, ID\_CLIENTE INT\*/

, FC\_INICIO\_LLAMADA DATETIME

, FC\_FIN\_LLAMADA DATETIME

, ID\_DEPARTAMENTO\_CC INT

, ID\_FLG\_TRANSFERIDO INT

, ID\_AGENTE\_CC INT

, FC\_INSERT DATETIME

, FC\_MODIFICACION DATETIME);

DROP TABLE IF EXISTS ODS\_DM\_DEPARTAMENTOS\_CC;

CREATE TABLE ODS\_DM\_DEPARTAMENTOS\_CC

(ID\_DEPARTAMENTO\_CC INT NOT NULL AUTO\_INCREMENT PRIMARY KEY

, DE\_DEPARTAMENTO\_CC VARCHAR(512)

, FC\_INSERT DATETIME

, FC\_MODIFICACION DATETIME);

DROP TABLE IF EXISTS ODS\_DM\_AGENTES\_CC;

CREATE TABLE ODS\_DM\_AGENTES\_CC

(ID\_AGENTE\_CC INT NOT NULL AUTO\_INCREMENT PRIMARY KEY

, DE\_AGENTE\_CC VARCHAR(512)

, FC\_INSERT DATETIME

, FC\_MODIFICACION DATETIME);

DROP TABLE IF EXISTS ODS\_DM\_FLG\_TRANSFER;

CREATE TABLE ODS\_DM\_FLG\_TRANSFER

(ID\_FLG\_TRANSFERIDO INT NOT NULL AUTO\_INCREMENT PRIMARY KEY

, DE\_FLG\_TRANSFERIDO VARCHAR(512)

, FC\_INSERT DATETIME

, FC\_MODIFICACION DATETIME);

1. Creación de FK asociadas al del modelo de Llamadas:

# CREACION FK MODELO DE LLAMADAS

USE ODS;

ALTER TABLE ODS\_HC\_LLAMADAS MODIFY COLUMN ID\_DEPARTAMENTO\_CC INT(10);

ALTER TABLE ODS\_HC\_LLAMADAS ADD INDEX fk\_lla\_dep\_idx (ID\_DEPARTAMENTO\_CC ASC);

ALTER TABLE ODS\_HC\_LLAMADAS ADD CONSTRAINT fk\_lla\_dep FOREIGN KEY(ID\_DEPARTAMENTO\_CC)

REFERENCES ODS\_DM\_DEPARTAMENTOS\_CC(ID\_DEPARTAMENTO\_CC);

ALTER TABLE ODS\_HC\_LLAMADAS MODIFY COLUMN ID\_AGENTE\_CC INT(10);

ALTER TABLE ODS\_HC\_LLAMADAS ADD INDEX fk\_lla\_age\_idx (ID\_AGENTE\_CC ASC);

ALTER TABLE ODS\_HC\_LLAMADAS ADD CONSTRAINT fk\_lla\_age FOREIGN KEY(ID\_AGENTE\_CC)

REFERENCES ODS\_DM\_AGENTES\_CC(ID\_AGENTE\_CC);

ALTER TABLE ODS\_HC\_LLAMADAS MODIFY COLUMN ID\_FLG\_TRANSFERIDO INT(10);

ALTER TABLE ODS\_HC\_LLAMADAS ADD INDEX fk\_lla\_transf\_idx (ID\_FLG\_TRANSFERIDO ASC);

ALTER TABLE ODS\_HC\_LLAMADAS ADD CONSTRAINT fk\_lla\_transf FOREIGN KEY(ID\_FLG\_TRANSFERIDO)

REFERENCES ODS\_DM\_FLG\_TRANSFER(ID\_FLG\_TRANSFERIDO);

1. Poblamos el modelo de Llamadas:

# POBLAMOS TABLAS DE MODELO DE LLAMADAS

USE ODS;

INSERT INTO ODS.ODS\_DM\_AGENTES\_CC VALUES (-98,'NO APLICA', NOW(), NOW());

INSERT INTO ODS.ODS\_DM\_AGENTES\_CC VALUES (-99,'DESCONOCIDO', NOW(), NOW());

COMMIT;

ANALYZE TABLE ODS\_DM\_AGENTES\_CC;

INSERT INTO ODS\_DM\_FLG\_TRANSFER VALUES (-98,'NO APLICA', NOW(), NOW());

INSERT INTO ODS.ODS\_DM\_FLG\_TRANSFER VALUES (-99,'DESCONOCIDO', NOW(), NOW());

COMMIT;

ANALYZE TABLE ODS\_DM\_FLG\_TRANSFER;

INSERT INTO ODS\_DM\_DEPARTAMENTOS\_CC VALUES (-98,'NO APLICA', NOW(), NOW());

INSERT INTO ODS.ODS\_DM\_DEPARTAMENTOS\_CC VALUES (-99,'DESCONOCIDO', NOW(), NOW());

COMMIT;

ANALYZE TABLE ODS\_DM\_DEPARTAMENTOS\_CC;

# Actualización periódica de las tablas

INSERT INTO ODS.ODS\_DM\_AGENTES\_CC (DE\_AGENTE\_CC

, FC\_INSERT

, FC\_MODIFICACION)

SELECT DISTINCT UPPER(TRIM(STAGE.STG\_CONTACTOS\_IVR.AGENT)), NOW(), NOW()

FROM STAGE.STG\_CONTACTOS\_IVR

LEFT OUTER JOIN ODS.ODS\_DM\_AGENTES\_CC

ON UPPER(TRIM(STAGE.STG\_CONTACTOS\_IVR.AGENT)) =

UPPER(TRIM(ODS.ODS\_DM\_AGENTES\_CC.DE\_AGENTE\_CC))

WHERE LENGTH (TRIM(STAGE.STG\_CONTACTOS\_IVR.AGENT))<>0

AND ODS.ODS\_DM\_AGENTES\_CC.DE\_AGENTE\_CC IS NULL;

COMMIT;

INSERT INTO ODS.ODS\_DM\_FLG\_TRANSFER (DE\_FLG\_TRANSFERIDO

, FC\_INSERT

, FC\_MODIFICACION)

SELECT DISTINCT UPPER(TRIM(STAGE.STG\_CONTACTOS\_IVR.FLG\_TRANSFER)), NOW(), NOW()

FROM STAGE.STG\_CONTACTOS\_IVR

LEFT OUTER JOIN ODS.ODS\_DM\_FLG\_TRANSFER

ON UPPER(TRIM(STAGE.STG\_CONTACTOS\_IVR.FLG\_TRANSFER)) =

UPPER(TRIM(ODS.ODS\_DM\_FLG\_TRANSFER.DE\_FLG\_TRANSFERIDO))

WHERE LENGTH (TRIM(STAGE.STG\_CONTACTOS\_IVR.FLG\_TRANSFER))<>0

AND ODS.ODS\_DM\_FLG\_TRANSFER.DE\_FLG\_TRANSFERIDO IS NULL;

COMMIT;

INSERT INTO ODS.ODS\_DM\_DEPARTAMENTOS\_CC (DE\_DEPARTAMENTO\_CC

, FC\_INSERT

, FC\_MODIFICACION)

SELECT DISTINCT UPPER(TRIM(STAGE.STG\_CONTACTOS\_IVR.SERVICE)), NOW(), NOW()

FROM STAGE.STG\_CONTACTOS\_IVR

LEFT OUTER JOIN ODS.ODS\_DM\_DEPARTAMENTOS\_CC

ON UPPER(TRIM(STAGE.STG\_CONTACTOS\_IVR.SERVICE)) =

UPPER(TRIM(ODS.ODS\_DM\_DEPARTAMENTOS\_CC.DE\_DEPARTAMENTO\_CC))

WHERE LENGTH (TRIM(STAGE.STG\_CONTACTOS\_IVR.SERVICE))<>0

AND ODS.ODS\_DM\_DEPARTAMENTOS\_CC.DE\_DEPARTAMENTO\_CC IS NULL;

COMMIT;

INSERT INTO ODS.ODS\_HC\_LLAMADAS (ID\_IVR

, TELEFONO\_LLAMADA

, FC\_INICIO\_LLAMADA

, FC\_FIN\_LLAMADA

, ID\_DEPARTAMENTO\_CC

, ID\_FLG\_TRANSFERIDO

, ID\_AGENTE\_CC

, FC\_INSERT

, FC\_MODIFICACION)

SELECT CON.ID

, CASE WHEN LENGTH(TRIM(CON.PHONE\_NUMBER))<>0

THEN CON.PHONE\_NUMBER ELSE 999999999 END AS TELEFONO\_LLAMADA

, CASE WHEN LENGTH(TRIM(CON.START\_DATETIME))<>0

THEN STR\_TO\_DATE(DATE\_FORMAT(CON.START\_DATETIME,'%d/%m/%Y'),'%d/%m/%Y')

ELSE STR\_TO\_DATE('31/12/9999','%d/%m/%Y') END AS FC\_INICIO\_LLAMADA

, CASE WHEN LENGTH(TRIM(CON.END\_DATETIME))<>0

THEN STR\_TO\_DATE(DATE\_FORMAT(CON.END\_DATETIME,'%d/%m/%Y'),'%d/%m/%Y')

ELSE STR\_TO\_DATE('31/12/9999','%d/%m/%Y') END AS FC\_FIN\_LLAMADA

, DEP.ID\_DEPARTAMENTO\_CC

, TRANS.ID\_FLG\_TRANSFERIDO

, AGENT.ID\_AGENTE\_CC

, NOW()

, NOW()

FROM STAGE.STG\_CONTACTOS\_IVR AS CON

INNER JOIN ODS.ODS\_DM\_DEPARTAMENTOS\_CC AS DEP

ON CASE WHEN LENGTH(TRIM(CON.SERVICE))<>0 THEN UPPER(TRIM(CON.SERVICE))

ELSE 'DESCONOCIDO' END = DEP.DE\_DEPARTAMENTO\_CC

INNER JOIN ODS.ODS\_DM\_FLG\_TRANSFER AS TRANS

ON CASE WHEN LENGTH(TRIM(CON.FLG\_TRANSFER))<>0

THEN UPPER(TRIM(CON.FLG\_TRANSFER))

ELSE 'DESCONOCIDO' END = TRANS.DE\_FLG\_TRANSFERIDO

INNER JOIN ODS.ODS\_DM\_AGENTES\_CC AS AGENT

ON CASE WHEN LENGTH(TRIM(CON.AGENT))<>0

THEN UPPER(TRIM(CON.AGENT))

ELSE 'DESCONOCIDO' END = AGENT.DE\_AGENTE\_CC;

COMMIT;

ANALYZE TABLE ODS.ODS\_HC\_LLAMADAS;

**SEGUNDA PARTE**

**2.1.** Adjunta tu diagrama de ODS completo con el número de registros que contiene cada tabla

*ver: Modelo\_Completo.mwb*

|  |  |
| --- | --- |
| **TABLA** | **NUM\_REGISTROS** |
| ODS.ODS\_DM\_AGENTES\_CC | 595 |
| ODS.ODS\_DM\_CICLOS\_FACTURACION | 4 |
| ODS.ODS\_DM\_METODOS\_PAGO | 5 |
| ODS.ODS\_DM\_CANALES | 6 |
| ODS.ODS\_DM\_CIUDADES\_ESTADOS | 159 |
| ODS.ODS\_DM\_COMPANYAS | 385 |
| ODS.ODS\_DM\_FLG\_TRANSFER | 4 |
| ODS.ODS\_DM\_PAISES | 3 |
| ODS.ODS\_DM\_PRODUCTOS | 8 |
| ODS.ODS\_DM\_PROFESIONES | 197 |
| ODS.ODS\_DM\_DEPARTAMENTOS\_CC | 8 |
| ODS.ODS\_DM\_SEXOS | 4 |
| ODS.ODS\_HC\_CLIENTES | 17547 |
| ODS.ODS\_HC\_LLAMADAS | 202717 |
| ODS.ODS\_HC\_DIRECCIONES | 113262 |
| ODS.ODS\_HC\_FACTURAS | 420000 |
| ODS.ODS\_HC\_SERVICIOS | 78467 |

SELECT 'ODS\_DM\_AGENTES\_CC' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_DM\_AGENTES\_CC

UNION ALL

SELECT 'ODS.ODS\_DM\_CICLOS\_FACTURACION' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_DM\_CICLOS\_FACTURACION

UNION ALL

SELECT 'ODS.ODS\_DM\_METODOS\_PAGO' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_DM\_METODOS\_PAGO

UNION ALL

SELECT 'ODS.ODS\_DM\_CANALES' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_DM\_CANALES

UNION ALL

SELECT 'ODS.ODS\_DM\_CIUDADES\_ESTADOS' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_DM\_CIUDADES\_ESTADOS

UNION ALL

SELECT 'ODS.ODS\_DM\_COMPANYAS' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_DM\_COMPANYAS

UNION ALL

SELECT 'ODS.ODS\_DM\_FLG\_TRANSFER' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_DM\_FLG\_TRANSFER

UNION ALL

SELECT 'ODS.ODS\_DM\_PAISES' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_DM\_PAISES

UNION ALL

SELECT 'ODS.ODS\_DM\_PRODUCTOS' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_DM\_PRODUCTOS

UNION ALL

SELECT 'ODS.ODS\_DM\_PROFESIONES' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_DM\_PROFESIONES

UNION ALL

SELECT 'ODS.ODS\_DM\_DEPARTAMENTOS\_CC' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_DM\_DEPARTAMENTOS\_CC

UNION ALL

SELECT 'ODS.ODS\_DM\_SEXOS' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_DM\_SEXOS

UNION ALL

SELECT 'ODS.ODS\_HC\_CLIENTES' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_HC\_CLIENTES

UNION ALL

SELECT 'ODS.ODS\_HC\_LLAMADAS' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_HC\_LLAMADAS

UNION ALL

SELECT 'ODS.ODS\_HC\_DIRECCIONES' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_HC\_DIRECCIONES

UNION ALL

SELECT 'ODS.ODS\_HC\_FACTURAS' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_HC\_FACTURAS

UNION ALL

SELECT 'ODS.ODS\_HC\_SERVICIOS' AS TABLA, COUNT(\*) AS NUM\_REGISTROS FROM ODS.ODS\_HC\_SERVICIOS

**2.2.** ¿Por qué en el modelo de DIRECCIONES dejo en la misma tabla las CIUDADES y los ESTADOS y no los separo en dos tablas distintas para ser más estricta con la jerarquía:

PAIS → ESTADOS → CIUDADES → DIRECCIONES

Se deja en la misma tabla, porque la relación de CIUDAD/ESTADO/PAIS no se repite, al tenerlo de esta manera sólo habrá que acceder una vez y a una sola tabla para obtener la información.

Además, esta tabla se puede mantener de forma automática y con el tiempo será estable por lo que no requerirá de pesadas actualizaciones que penalicen en el proceso de mantenimiento de dicha tabla.

**2.3.** Separar el campo DE\_DIRECCION de la tabla de direcciones en dos campos: NOMBRE\_VIA y NUM\_VIA

SELECT DE\_DIRECCION

, LEFT(TRIM(DE\_DIRECCION), INSTR(DE\_DIRECCION, ' ') - 1) AS NUM\_VIA

, RIGHT(TRIM(DE\_DIRECCION), LENGTH(TRIM(DE\_DIRECCION)) - INSTR(DE\_DIRECCION, ' ')) AS NOMBRE\_VIA

FROM ODS.ODS\_HC\_DIRECCIONES;

**TERCERA PARTE**

La realidad es que si hubiésemos aplicado el “Data Management”, muchas de las acciones que hemos tenido que realizar nos las hubiésemos evitado porque deberían estar controladas de otra forma.

Explica qué habrías hecho diferente centrándote en las “patas”:

* Data Quality

*Nota: Además de las obvias que nos han salido al crear ODS que hay que describirlas, ¿se te ocurre alguna otra normalización?*

* Master Data
* Data Modeling & Design (me refiero a cómo están definidas las tablas en origen)

¿Aconsejarías algún cambio en los sistemas origen extra teniendo en cuenta el resto de disciplinas del Data Governance?

1. **Mejoras en Data Quality (mejora de la calidad del dato)**

Todas las transformaciones para estandarizar los datos podrían ser llevadas a cabo desde las tablas operacionales, estas son:

* Empleo de identificadores (ID) en datos estándares como: ciudades, estados, países; lo cual nos permite sustituir los textos por dichos identificadores.
* Formato de las fechas: uso de un mismo formato en todas las tablas
* Inicialización de datos vacíos
* Validación del tipo de dato
* Validación para datos obligatorios

1. **Mejoras en el Data Master (optimización de datos maestros)**

Identificación de cuales son los datos maestros, para realizar la carga diaria de las tablas auxiliares para la correcta explotación de los mismos. En ellos se realizaran trabajos de: detección de errores, análisis y validación.

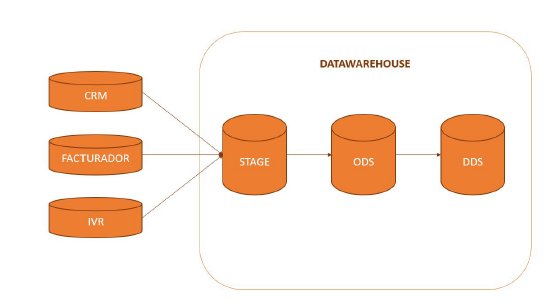
1. **Mejoras en el Data Modeling & Design (optimización del diseño de recogida de datos):**

Las tablas se relacionarían a través de los Id’s que tengan en común:

|  |  |  |  |
| --- | --- | --- | --- |
| **TABLA** | **CAMPOS** | **TABLA** | **CAMPOS** |
| CLIENTES | CUSTOMER\_ID | CITY | ID\_CITY |
| FIRST\_NAME | DE\_CITY |
| LAST\_NAME | FC\_INIT |
| IDENTIFIED\_DOC | FC\_MODIF |
| GENDER |  |  |
| ID\_DIRECCION | STATE | ID\_STATE |
| PHONE | DE\_STATE |
| EMAIL | FC\_INIT |
| BIRTHDAY | FC\_MODIF |
| ID\_PROFESION |  |  |
| ID\_COMPANY | COUNTRY | ID\_COUNTRY |
| FC\_INIT | DE\_COUNTRY |
| FC\_MODIF | FC\_INIT |
|  |  | FC\_MODIF |
| DIRECCION | ID\_DIRECCION |  |  |
| TIPO\_VIA | PROFESSION | ID\_PROFESSION |
| NOMBRE\_DIRECCION | DE\_PROFESSION |
| NRO\_DIRECCION | FC\_INIT |
| ID\_CITY | FC\_MODIF |
| ID\_STATE |  |  |
| ID\_COUNTRY | COMPANY | ID\_COMPANY |
| POSTAL\_CODE | DE\_COMPANY |
| FC\_INIT | FC\_INIT |
| FC\_MODIF | FC\_MODIF |
|  |  |  |  |
| PRODUCTOS\_REF | ID\_PRODUCTO | AGENT | ID\_AGENT |
| PRODUCT\_NAME | DE\_AGENT |
| ID\_DIRECCION | FC\_INIT |
|  |  | FC\_MODIF |
| PRODUCTO | ID\_PRODUCTO |  |  |
| CUSTOMER\_ID |  |  |
| ACCESS\_POINT |  |  |
| CHANNEL |  |  |
| START\_DATE |  |  |
| INSTALL\_DATE |  |  |
| END\_DATE |  |  |
| ID\_AGENT |  |  |
|  |  |  |  |
| FACTURAS | ID\_FACTURA | BILL\_CYCLE | ID\_BILL\_CYCLE |
| BILL\_REF\_NO | DE\_BILL\_CYCLE |
| CUSTOMER\_ID | FC\_INIT |
| START\_DATE | FC\_MODIF |
| END\_DATE |  |  |
| STATEMENT\_DATE | BILL\_METHOD | ID\_BILL\_METHOD |
| PAYMENT\_DATE | DE\_BILL\_METHOD |
| AMOUNT | FC\_INIT |
| ID\_BILL\_CYCLE | FC\_MODIF |
| ID\_BILL\_METHOD |  |  |
|  |  |  |  |
| CONTACTOS | ID\_CONTACTO | SERVER | ID\_SERVER |
| PHONE\_NUMBER | DE\_SERVER |
| START\_DATE | FC\_INIT |
| END\_DATE | FC\_MODIF |
| ID\_SERVER |  |  |
| ID\_FLG\_TRANSFER | FLG\_TRANSFER | ID\_TRANSFER |
| ID\_AGENT | DE\_TRANSFER |
|  |  | FC\_INIT |
|  |  | FC\_MODIF |

**CUARTA PARTE**

Después de todo lo visto nuestro ecosistema quedaría así:



¿Lo dejarías así o plantearías otro diseño mejorado? Si es que sí, esbózalo.

**QUINTA PARTE**

Escribe tus propias reglas o mandamientos de un DataWarehouse

Con el objetivo de la gestión de los datos como un activo preciado por la información y el conocimiento que se pueden adquirir de ellos, sería recomendable el empleo de la disciplina de Data Management desde la ingesta de los datos por los sistemas operacionales.

Dispondríamos de un primer bloque de Sistemas Operacionales donde se llevarían a cabo las disciplinas de:

1. Data Governance: con la finalidad de tener localizado en todo momento los datos, garantizando así su trazabilidad
2. Data Architecture: para diseñar una arquitectura lógica del proceso
3. Data Storage: con el objetivo de que las aplicaciones guarden los datos de una manera adecuada, es decir, que existan unos procesos de normalizado de los datos para ser guardados con el mismo formato.

Al final de todo el proceso obtenemos una BBDD con una la calidad que estamos buscando.

Dispondríamos de un segundo bloque en el cual se volcaría toda la información sin apenas realizar ningún tipo de tratamiento sobre los datos para, a continuación ser agrupados de manera adecuada en DDS. A partir de DDS, se repartiría la información demandada por los distintos departamentos.