PRÁCTICA DATA 101

Jaime Gómez Recasens

Índice

Primera parte

En esta primera parte haremos un estudio de las siguientes tablas:

- STG_PRODUCTOS_CRM
- STG_FACTURAS_FCT
- STG_CONTACTOS_IVR

Luego crearemos los modelos con la creación de las tablas, las claves secundarias y los poblaremos de datos. Los modelos que se van a crear son:

- Servicios
- Facturas
- Llamadas

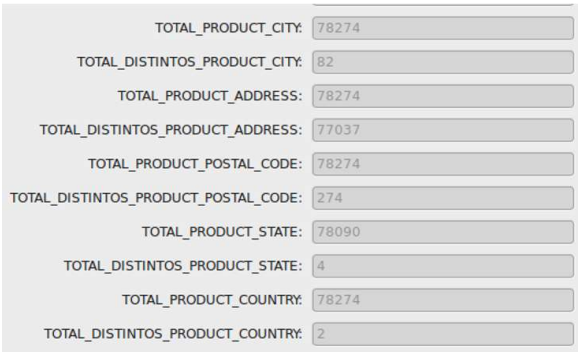
Estudio de tablas

La primera tabla que estudiamos es STG_PRODUCTOS_CRM. En esta tabla hay 78.495 registros. Los campos que la componen son: ID_PRODUCTO, ID_CLIENTE, NOMBRE_PRODUCTO, PUNTO_ACCESO, CANAL, CODIGO_AGENTE, FECHA_INICIO, FECHA_INSTALACION, FECHA_FIN, y DIRECCIÓN (engloba los campos CIUDAD_PRODUCTO, DIRECCION_PRODUCTO, CP_PRODUCTO, ESTADO_PRODUCTO, PAIS_PRODUCTO).

Hay que tener en cuenta que la dirección que consta en esta tabla es la especificada para la instalación del suministro y puede ser diferente a la del cliente. También hay que considerar que un cliente puede tener contratado de 0 a N productos y que, aproximadamente, un 10-14% de los registros contienen algún/os campo/s sin valor o con errores.

Si hacemos un estudio de la tabla en detalle para obtener todos los registros por campo y los registros distintos que hay esto es lo que obtenemos:

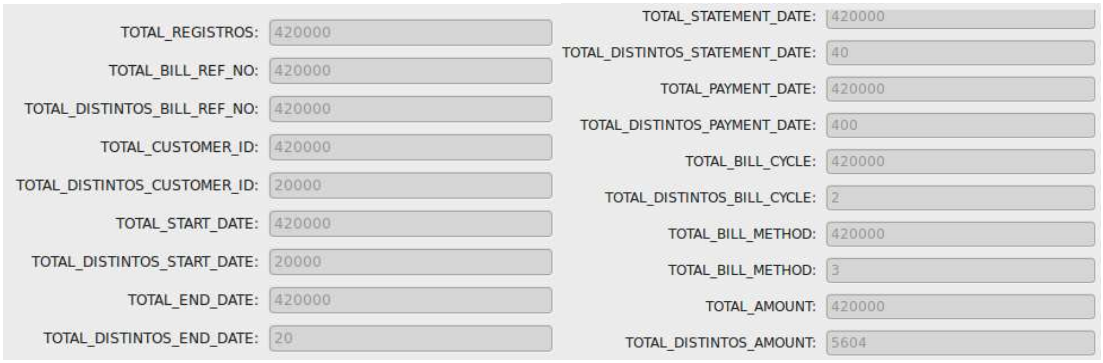
TOTAL_REGISTROS:	78495	TOTAL_DISTINTOS_CHANNEL:	5
TOTAL_PRODUCT_ID:	78495	TOTAL_AGENT_CODE:	42630
TOTAL_DISTINTOS_PRODUCT_ID:	78495	TOTAL_DISTINTOS_AGENT_CODE:	701
TOTAL_CUSTOMER_ID:	78495	TOTAL_START_DATE:	78495
TOTAL_DISTINTOS_CUSTOMER_ID:	8001	TOTAL_DISTINTOS_START_DATE:	8035
TOTAL_PRODUCT_NAME:	78495	TOTAL_INSTALL_DATE:	75363
TOTAL_DISTINTOS_PRODUCT_NAME:	8001	TOTAL_DISTINTOS_INSTALL_DATE:	75360
TOTAL_ACCESS_POINT:	78274	TOTAL_END_DATE:	46684
TOTAL_DISTINTOS_ACCESS_POINT:	78275	TOTAL_END_DATE:	46683
TOTAL_CHANNEL:	78274		



La siguiente tabla que estudiamos es STG_FACTURAS_FCT. En esta tabla hay 420.000 registros. Los campos que la componen son: ID_FACTURA, ID_CLIENTE, FECHA_INICIO, FECHA_FIN, FECHA_ESTADO, FECHA_PAGO, CICLO_FACTURA, CANTIDAD, METODO_PAGO.

En esta tabla no hay ningún error y también tenemos que tener en cuenta que un cliente puede tener de 0 a N facturas.

Si hacemos un estudio de la tabla en detalle para obtener todos los registros por campo y los registros distintos que hay esto es lo que obtenemos:



La última tabla a estudiar es STG_CONTACTOS_IVR. Tiene 202.717 registros y los campos que la componen son: NUMERO_TELEFONO, FECHA_INICIO, FECHA_FIN, SERVICIO, FLG_TRANSFERIDO Y AGENTE.

Las consideraciones que debemos de tener son: el 16% de los registros contienen algún/os campo/s sin valor o con errores y un cliente puede tener de 0 a N contactos.

Si hacemos un estudio de la tabla en detalle para obtener todos los registros por campo y los registros distintos que hay esto es lo que obtenemos:

TOTAL_REGISTROS:	202717
TOTAL_PHONE_NUMBER:	185018
TOTAL_DISTINTOS_PHONE_NUMBER:	18226
TOTAL_START_DATETIME:	202717
TOTAL_DISTINTOS_START_DATETIME:	201098
TOTAL_END_DATETIME:	186535
TOTAL_DISTINTOS_END_DATETIME:	183678
TOTAL_SERVICE:	202502
TOTAL_DISTINTOS_SERVICE:	7
TOTAL_FLG_TRANSFER:	202717
TOTAL_DISTINTOS_FLG_TRANSFER:	2
TOTAL_AGENT:	194739
TOTAL_DISTINTOS_AGENT:	594

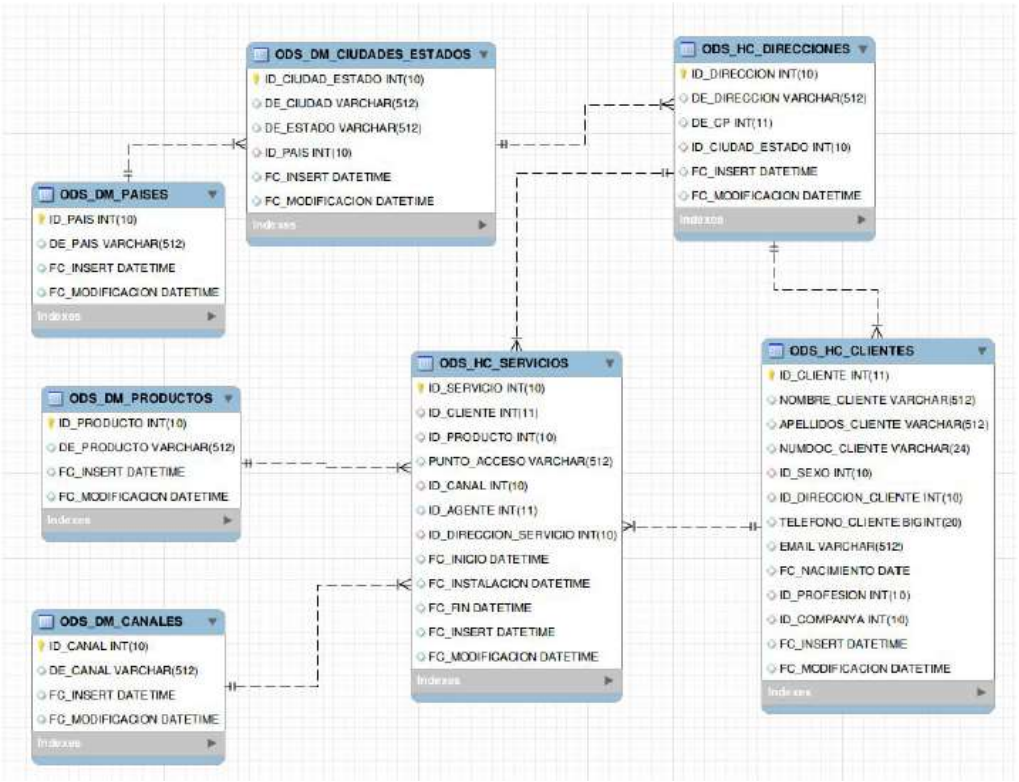
Creación de modelo

Una vez analizadas las tablas podemos crear 3 modelos: Servicios, facturas y llamadas. Vamos con el primero.

Modelo SERVICIOS

A raíz del estudio de la tabla STG_PRODUCTOS_CRM llegamos a la conclusión de que el modelo Servicios se puede componer de 7 tablas. Tres de "hechos" y cuatro de "dimensiones". Las tablas son: ODS_HC_SERVICIOS, ODS_DM_PRODUCTOS, ODS_DM_CANALES, ODS_HC_DIRECCIONES, ODS_HC_CLIENTES, ODS_DM_CIUDADES_ESTADOS, ODS_DM_PRODUCTOS Y ODS_DM_CANALES. Las últimas 4 tablas ya se crearon para el modelo de Clientes por lo que aquí no las volveremos a crear.

Si dibujamos el modelo en un gráfico quedaría así:

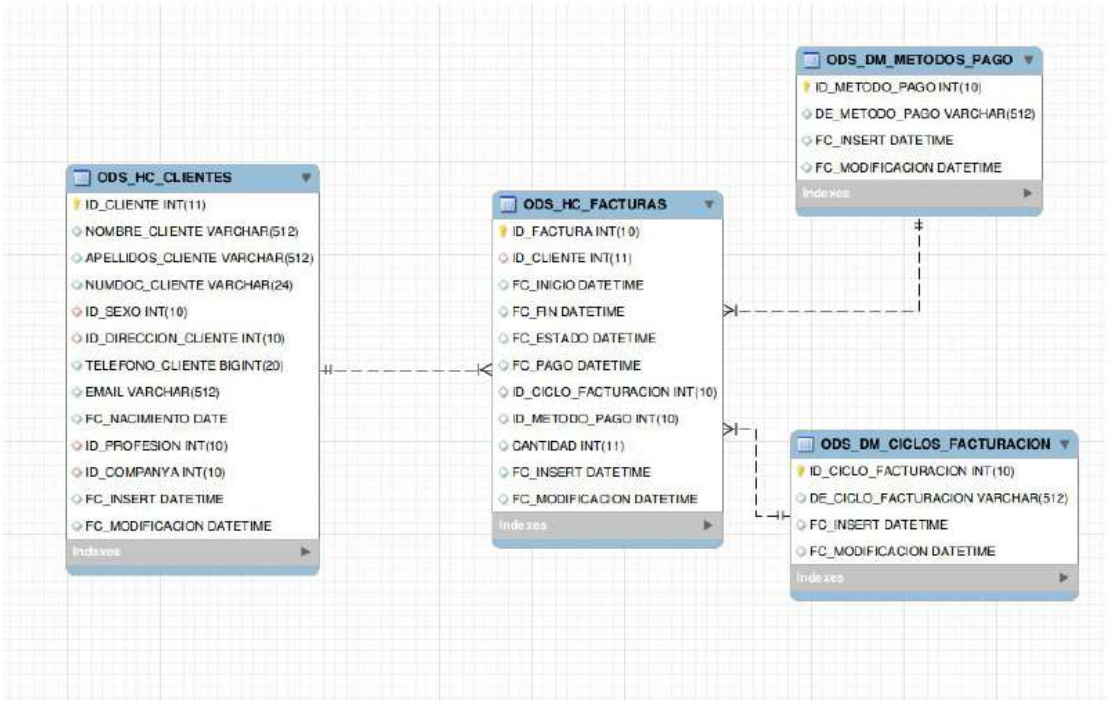


Una vez creado podemos proceder a crear las tablas, las claves y poblar el conjunto de datos. En el anejo 1 están todas las instrucciones SQL empleadas para ello.

Modelo FACTURAS

A raíz del estudio de la tabla STG_FACTURAS_FCT llegamos a la conclusión de que el modelo Facturas se puede componer de 4 tablas. Dos de “hechos” y dos de “dimensiones”. Las tablas son: ODS_HC_FACTURAS, ODS_DM_METODOS_PAGO, ODS_DM_CICLOS_FACTURACION y ODS_HC_CLIENTES. Esta última tabla ya se creó para el modelo de Clientes por lo que aquí no las volveremos a crear.

La representación gráfica del modelo quedaría así:

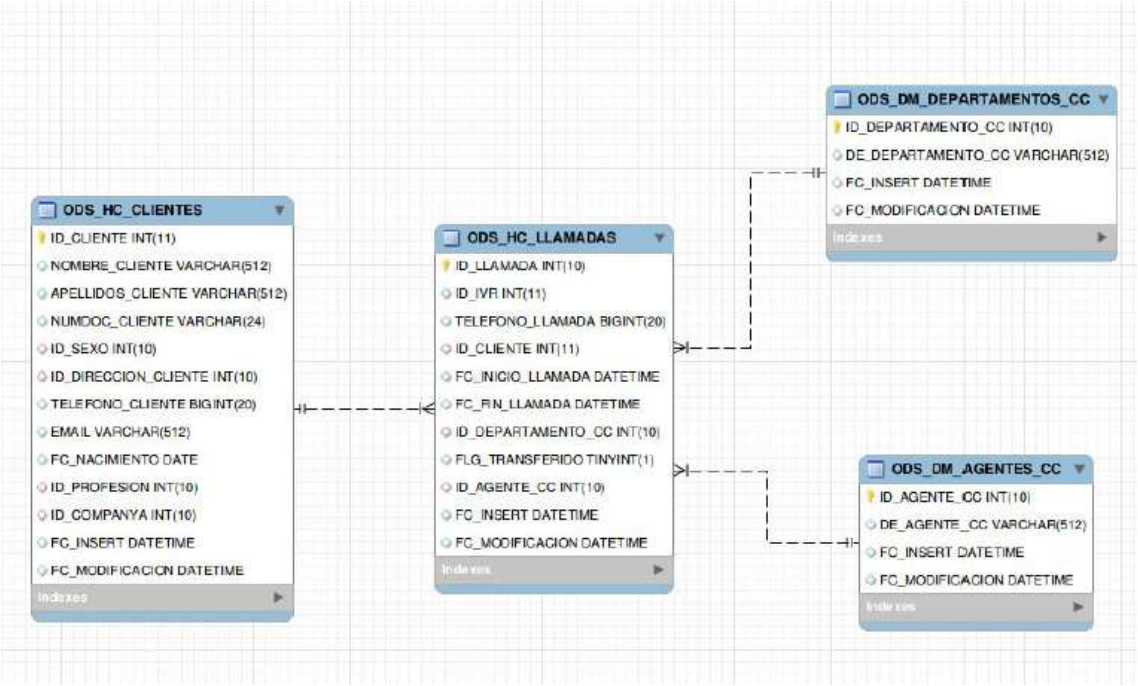


Una vez creado podemos proceder a crear las tablas, las claves y poblar el conjunto de datos. En el anejo 2 están todas las instrucciones SQL empleadas para ello.

Modelo LLAMADAS

A raíz del estudio de la tabla STG_CONTACTOS_IVR creamos el modelo Llamadas con cuatro tablas. Dos de “hechos” y dos de “dimensiones”. Las tablas son: ODS_HC_LLAMADAS, ODS_DM_DEPARTAMENTOS_CC, ODS_DM_AGENTES_CC y ODS_HC_CLIENTES. Esta última tabla ya se creó para el modelo de Clientes por lo que aquí no las volveremos a crear.

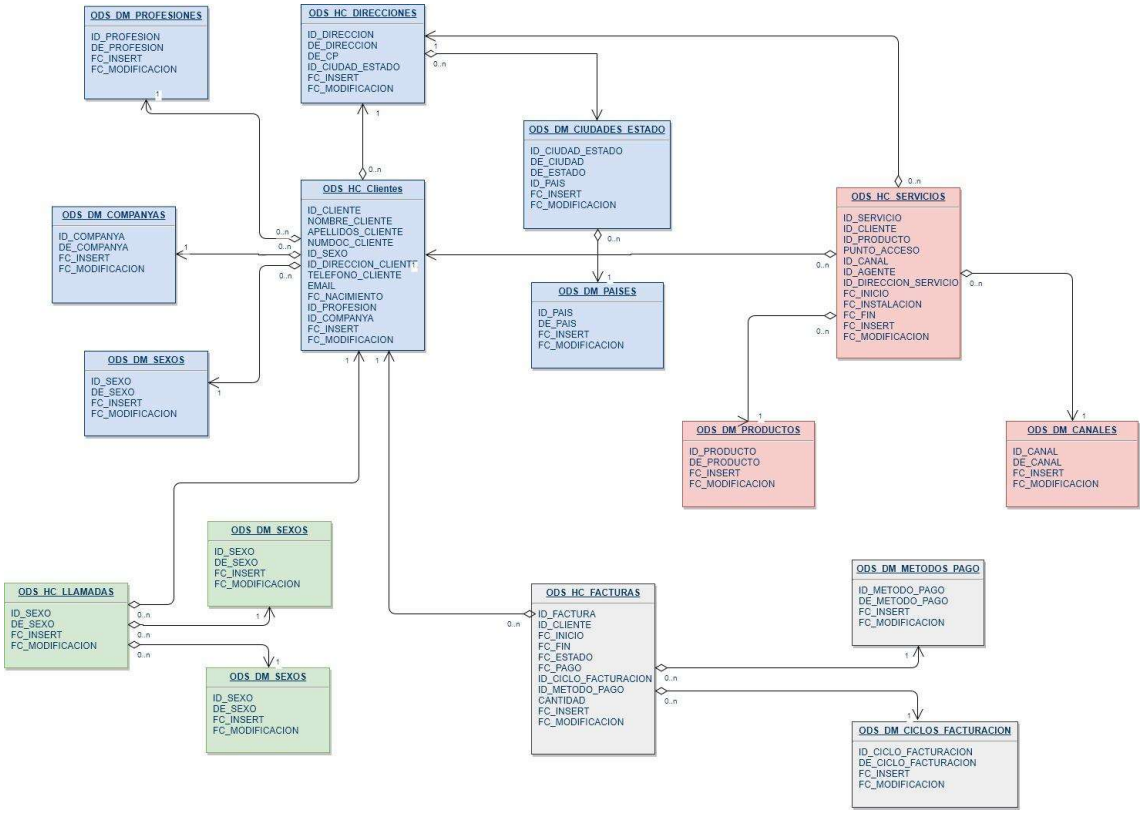
El diagrama del modelo quedaría así:



Una vez creado podemos proceder a crear las tablas, las claves y poblar el conjunto de datos. En el anejo 3 están todas las instrucciones SQL empleadas para ello.

Segunda parte

En este punto dibujamos nuestro modelo ODS completo:



En cuanto al número de registros que contiene cada tabla estos son los datos que obtenemos:

- ODS_HC_CLIENTES: 20.001 registros.
- ODS_DM_SEXOS: 4 registros.
- ODS_DM_COMPANYAS: 395 registros.
- ODS_DM_PROFESIONES: 197 registros.
- ODS_HC_DIRECCIONES: 95.769 registros.
- ODS_DM_CIUDADES_ESTADOS: 158 registros.
- ODS_DM_PAISES: 3 registros.
- ODS_HC_SERVICIOS: 78.495 registros.
- ODS_DM_PRODUCTOS: 8 registros.
- ODS_DM_CANALES: 6 registros.
- ODS_HC_FACTURAS: 420.000 registros.
- ODS_DM_METODOS_PAGO: 5 registros.
- ODS_DM_CICLOS_FACTURACION: 4 registros.
- ODS_HC_LLAMADAS: 202.717 registros.
- ODS_DM_DEPARTAMENTO_CC: 8 registros.
- ODS_DM_AGENTES_CC: 595 registros.

Respecto a la pregunta de por qué no se han separado en el modelo de DIRECCIONES los campos CIUDADES y ESTADOS y se han dejado en la misma tabla yo creo que el motivo es para no dividir las jerarquías y los niveles de las jerarquías en múltiples dimensiones.

Si se crean dimensiones extras para niveles de jerarquías, se complican las sentencias que se deberán realizar para el consumo de los datos del data warehouse. Además, al ser una dimensión innecesaria, se aumenta considerablemente el espacio ocupado (ya que se generarían millones de registros con el campo de la dimensión.

Tercera parte

En este punto indicamos lo que habríamos hecho en caso de aplicar "Data Management".

Data Quality

Si aplicásemos "Data Quality" a este modelo yo me centraría sobre todo en la limpieza y enriquecimiento del dato:

- Buscaría herramientas para determinar y separar elementos de un campo situándolo en su lugar correspondiente.
- Estandarizaría formatos: para adecuar los datos a un formato esperado. Por ejemplo: CIF, CP, números de teléfono, etc.
- Intentaría corregir posibles datos en los datos reemplazando los elementos erróneos por su valor correcto. Por ejemplo: nombres de calle que han cambiado, CP que no se corresponden con la calle, etc.
- Enriquecimiento de datos para añadir datos que no existían. Ejemplo: Sr. Javier Fernández => sexo = varón.

También buscaría casar datos para detectar duplicados, crear relaciones entre dos fuentes de datos que no tienen campos de unión entre sí e intentar detectar unidades familiares y/o corporativas.

Una vez hecho el paso anterior podríamos intentar fusionar registros consolidando así los datos.

Master Data

Si aplicásemos un MDM el mayor cambio radicaría en los diferentes tipos de datos que emplean uno y otro. La gestión de datos maestros se aplica solamente a las entidades y no a los datos de transacciones, mientras que un "*data warehouse*" incluye datos que son a la vez transaccionales y no transaccionales en su naturaleza. Así, podríamos decir que un MDM solo afecta a los datos que existen en tablas dimensionales y no en tablas de hechos, mientras que en un entorno de data warehouse se incluyen tanto las tablas dimensionales como las tablas de hechos.

De esta forma, al aplicar MDM tendríamos que descartar las bases de datos de hechos.

Data Modeling & Design

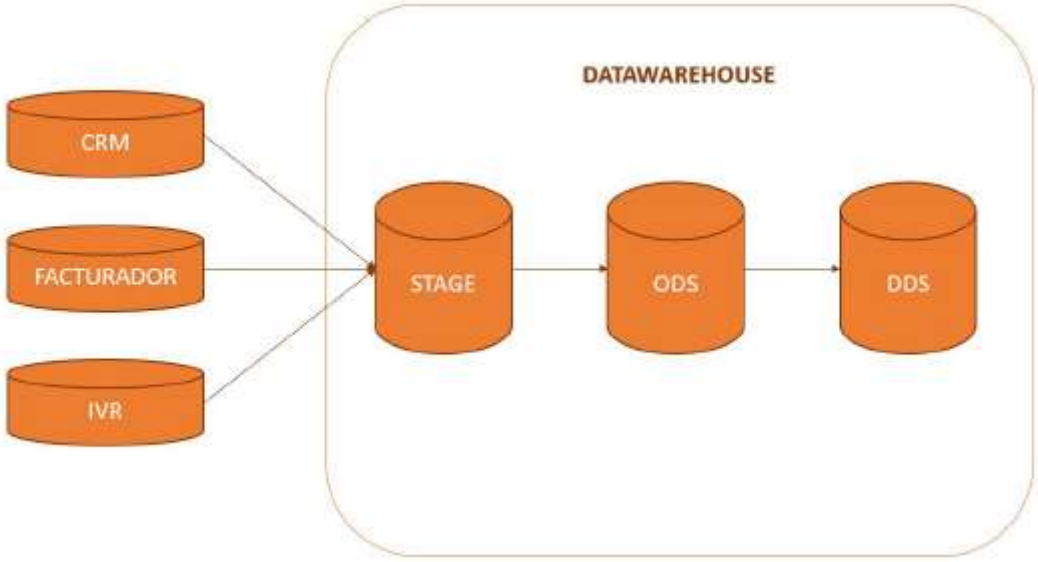
A la hora de realizar el modelado de datos y diseño de la base de datos habría que cumplir estos requisitos:

- Dominio: ¿cuál es el dominio que esta solución necesita abordar?.
- Funcionalidad: ¿cuál es la funcionalidad requerida? ¿cuáles son los casos de uso relacionados con estos datos?
- Entidades: crear las entidades de la base de datos modelando las entidades principales del sistema.
- Relaciones: definir cuáles y cómo están relacionadas las entidades definidas en el punto anterior.

- Diseño formal: una vez definidas las entidades y sus relaciones ya podemos construir un modelo o un diagrama de relaciones de entidades.

Cuarta parte

Yo no propondría ningún cambio en el modelo ya que el que hemos diseñado es el más sencillo y el que permite un mayor escalado. Además sigue el mandamiento DRY de no crear más copias de las estrictamente necesarias por lo que lo dejaría tal cual está.



Quinta parte

Yo solo seguiría un mandamiento a la hora de crear y diseñar un data warehouse y es: "**Don't repeat yourself**" o "no te repitas", o simplemente DRY. Es un principio que defiende la necesidad de reducir o eliminar todo tipo de duplicidades. Este principio, aplicado a la ingeniería del software, fue formulado por Andrew Hunt y David Thomas en **"The Pragmatic Programmer**".

El principio DRY debe ser valorado en todas y cada una de las decisiones que tomamos durante el ciclo de vida de un proyecto de datawarehousing). Hay que procurar evitar las duplicidades en la extracción de la información, en el tratamiento de estos datos, en el modelo de datos, en la arquitectura de la solución, en los patrones que seguimos, en las herramientas de explotación, en los informes que realizamos y en la estructura de cada uno de ellos, en la documentación, etc.

Por supuesto, y muy lamentablemente, a veces son inevitables las duplicidades. De hecho, paradójicamente, un data warehouse es la duplicación de datos que ya teníamos en otro lugar.

Los datos, indudablemente, parece que los estamos repitiendo en distintos sitios. Los datos que ya teníamos en los sistemas de origen los duplicamos temporalmente en la staging, los volvemos a duplicar en un modelo normalizado y finalmente los duplicamos en el modelo dimensional que será accedido por la capa de presentación. Como toda duplicación, debe justificarse, y en este caso se valora que los usos que se hacen en cada una de las áreas son distintos y claramente diferenciados:

- El data warehouse, en contraposición al sistema operacional de origen, tiene un enfoque analítico y cubre unos requerimientos que el operacional no alcanza (facilidad de uso, tiempo de respuesta, visión histórica…). Es decir, aunque los datos sean los mismos, el DW y el operacional son cosas distintas.
- El área de staging es una duplicidad evidente. Son los mismos datos que el sistema de origen. Es necesario justificarlo: Es temporal, es "fácil" duplicarlo, y evita que los procesos de carga del DWH (que pueden llegar a ser largos) afecten negativamente al operacional.
- El modelo normalizado es el modelo de datos DRY por excelencia. En un modelo 3NF, cada dato está una y solo una vez. Si hemos aceptado que necesitamos un DWH, y queremos una única "fuente de la verdad", y somos buenos discípulos de DRY, hemos de defender a capa y espada el modelo normalizado, con la ayuda de Codd.
- El modelo dimensional es un mal necesario. En el estado actual de la técnica, y con el volumen de información que gestionan las empresas, el modelo normalizado no cubre los requerimientos que justifican la existencia del DWH (facilidad de uso y tiempos de respuesta, fundamentalmente). El modelo dimensional será el único que será accedido por las herramientas de explotación. En este sentido, es muy distinto al modelo normalizado. Además, crear un modelo dimensional a partir de un modelo normalizado es una tarea casi trivial que se justifica fácilmente.

NOTA: Esta información está sacada de esta página Web.[1]

[1] https://www.businessintelligence.info/definiciones/lo-mas-importante-para-crear-datawarehouse-dry.html

Anejo 1 – Creación Modelo Servicios sentencias SQL

Sentencias SQL empleadas para la creación del modelo servicios:

Creación tablas SERVICIOS

```
/*Tablas:
        - ODS_HC_DIRECCIONES
        - ODS_HC_CLIENTES
        - ODS_DM_CIUDADES_ESTADOS
        - ODS_DM_PAISES
Ya creadas en modelo clientes */
USE ODS;

DROP TABLE IF EXISTS ODS_DM_CANALES;
CREATE TABLE ODS_DM_CANALES
(ID_CANAL INT(10) UNSIGNED AUTO_INCREMENT PRIMARY KEY
, DE_CANAL VARCHAR(512)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME);

DROP TABLE IF EXISTS ODS_HC_SERVICIOS;
CREATE TABLE ODS_HC_SERVICIOS
(ID_SERVICIO INT(10) UNSIGNED AUTO_INCREMENT PRIMARY KEY
, ID_CLIENT INT(11)
, ID_PRODUCT INT(10)
, PUNTO_ACCESO VARCHAR(512)
, ID_CANAL INT(10)
, ID_AGENTE INT(11)
, ID_DIRECCION_SERVICIO INT(10)
, FC_INICIO DATETIME
, FC_INSTALACION DATETIME
, FC_FIN DATETIME
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME);

DROP TABLE IF EXISTS ODS_DM_PRODUCTOS;
CREATE TABLE IF EXISTS ODS_DM_PRODUCTOS
(ID_PRODUCTO INT(10) UNSIGNED AUTO_INCREMENT PRIMARY KEY
, DE_PRODUCTO VARCHAR (512)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME);
```

Creación claves SERVICIOS

```
/* FK DEL MODELO DE SERVICIOS */
```

```
ALTER TABLE ODS_HC_SERVICIOS MODIFY COLUMN ID_CLIENTE INT(10) UNSIGNED;
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD INDEX fk_serv_cli_idx (ID_CLIENTE ASC);
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD CONSTRAINT fk_serv_cli FOREIGN KEY (ID_CLIENTE)
        REFERENCES ODS.ODS_HC_CLIENTES (ID_CLIENTE);

ALTER TABLE ODS_HC_SERVICIOS MODIFY COLUMN ID_PRODUCT INT(10) UNSIGNED;
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD INDEX fk_serv_prod_idx (ID_PRODUCTO ASC);
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD CONSTRAINT fk_serv_prod FOREIGN KEY (ID_PRODUCTO)
        REFERENCES ODS.ODS_DM_PRODUCTOS (ID_PRODUCTO);

ALTER TABLE ODS_HC_SERVICIOS MODIFY COLUMN ID_CANAL INT(10) UNSIGNED;
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD INDEX fk_serv_canal_idx (ID_CANAL ASC);
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD CONSTRAINT fk_serv_canal FOREIGN KEY (ID_CANAL)
        REFERENCES ODS.ODS_DM_CANALES (ID_CANAL);

ALTER TABLE ODS_HC_SERVICIOS MODIFY COLUMN ID_DIRECCION_SERVICIO INT(10) UNSIGNED;
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD INDEX fk_serv_dirs_idx (ID_DIRECCION_SERVICIO ASC);
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD CONSTRAINT fk_serv_dirs FOREIGN KEY (ID_DIRECCION_SERVICIO)
        REFERENCES ODS.ODS_HC_DIRECCIONES (ID_DIRECCION_SERVICIO);

ALTER TABLE ODS_HC_CLIENTES MODIFY COLUMN ID_DIRECCION_CLIENTE INT(10) UNSIGNED;
ALTER TABLE ODS.ODS_HC_CLIENTES ADD INDEX fk_cli_dirc_idx (ID_DIRECCION_CLIENTE ASC);
ALTER TABLE ODS.ODS_HC_CLIENTES ADD CONSTRAINT fk_cli_dirc FOREIGN KEY (ID_DIRECCION_CLIENTE)
        REFERENCES ODS.ODS_HC_DIRECCIONES;

ALTER TABLE ODS_HC_DIRECCIONES MODIFY COLUMN ID_CIUDAD_ESTADO INT(10) UNSIGNED;
ALTER TABLE ODS.ODS_HC_DIRECCIONES ADD INDEX fk_dir_ciu_idx (ID_CIUDAD_ESTADO ASC);
ALTER TABLE ODS.ODS_HC_DIRECCIONES ADD CONSTRAINT fk_dir_ciu FOREIGN KEY (ID_CIUDAD_ESTADO ASC)
        REFERENCES ODS.ODS_DM_CIUDADES_ESTADO;

ALTER TABLE ODS_DM_CIUDADES_ESTADO MODIFY COLUMN ID_PAIS INT(10) UNSIGNED;
ALTER TABLE ODS.ODS_DM_CIUDADES_ESTADO ADD INDEX fk_ciu_pais_idx (ID_PAIS ASC);
ALTER TABLE ODS.ODS_DM_CIUDADES_ESTADO ADD CONSTRAINT fk_ciu_pais FOREIGN KEY (ID_PAIS ASC)
        REFERENCES ODS.ODS_DM_PAISES;
```

Poblamos SERVICIOS

```
/*Tablas:
        - ODS_HC_DIRECCIONES
        - ODS_HC_CLIENTES
        - ODS_DM_CIUDADES_ESTADOS
        - ODS_DM_PAISES
Ya pobladas en modelo clientes */
```

```
-- TABLA ODS_DM_CANALES

INSERT INTO ODS_DM_CANALES (DE_CANAL, FC_INSERT, FC_MODIFICACION)

SELECT DISTINCT UPPER(TRIM(CHANNEL)) DE_CANAL, NOW(), NOW()

FROM STAGE.STG_PRODUCTOS_CRM

WHERE TRIM(CHANNEL)<>'';


INSERT INTO ODS_DM_CANALES VALUES (99, 'DESCONOCIDO', NOW(), NOW());

INSERT INTO ODS_DM_CANALES VALUES (98, 'NO APLICA', NOW(), NOW());


ANALYZE TABLE ODS_DM_CANALES;


-- TABLA ODS_HC_DIRECCIONES

INSERT INTO ODS_HC_DIRECCIONES (DE_DIRECCIONES, DE_CP, ID_CIUDAD_ESTADO, FC_INSERT, FC_MODIFICACION)

SELECT DISTINCT UPPER(TRIM(ADDRESS)) DIRECCION

, CASE WHEN LENGTH(TRIM(CLI.POSTAL_CODE))<>0 THEN TRIM(CLI.POSTAL_CODE) ELSE 99999 END CP,
CIU.ID_CIUDAD_ESTADO, NOW(), NOW()

FROM STAGE.STG_CLIENTES_CRM CLI

INNER JOIN ODS.ODS_DM_PAISES PAI ON CASE WHEN LENGTH(TRIM(CLI.COUNTRY))<>0 THEN CLI.COUNTRY ELSE
'DESCONOCIDO' END=PAI.DE_PAIS

INNER JOIN ODS.ODS_DM_CIUDADES_ESTADOS CIU ON CASE WHEN LENGTH(TRIM(CLI.CITY))<>0 THEN CLI.CITY ELSE
'DESCONOCIDO' END=CIU.DE_CIUDAD

                AND CASE WHEN LENGTH(TRIM(CLI.STATE))<>0 THEN CLI.STATE ELSE 'DESCONOCIDO'
END=CIU.DE_ESTADO

WHERE TRIM(ADDRESS)<>'';


INSERT INTO ODS_HC_DIRECCIONES VALUES (999999, 'DESCONODICO', 99999, 999, NOW(), NOW());

INSERT INTO ODS_HC_DIRECCIONES VALUES (999998, 'NO APLICA', 99998, 998, NOW(), NOW());


COMMIT;


ANALYZE TABLE ODS_HC_DIRECCIONES;


-- TABLA ODS_HM_SERVICIOS


DROP TABLE IF EXISTS TMP_DIRECCIONES_SERVICIOS;

CREATE TABLE TMP_DIRECCIONES_SERVICIOS AS

SELECT DIR.ID_DIRECCION

, DIR.DE_DIRECCION

, DIR.DE_CP

, CIU.DE_CIUDAD

, CIU.DE_ESTADO

, PAI.DE_PAIS

FROM ODS.ODS_HC_DIRECCIONES DIR

INNER JOIN ODS.ODS_DM_CIUDADES_ESTADOS CIU ON DIR.ID_CIUDAD_ESTADO=CIU.ID_CIUDAD_ESTADO
```

```
INNER JOIN ODS.ODS_DM_PAISES PAI ON CIU.ID_PAIS=PAI.ID_PAIS;


DROP TABLE IF EXISTS TMP_DIRECCIONES_SERVICIOS2;

CREATE TABLE TMP_DIRECCIONES_SERVICIOS2 AS

SELECT CLIENTES.CUSTOMER_ID ID_CLIENTE

, DIR.ID_DIRECCION

FROM STAGE.STG_CLIENTES_CRM CLIENTES

INNER JOIN ODS.TMP_DIRECCIONES_SERVICIOS DIR

        ON CASE WHEN TRIM(ADDRESS)<>'' THEN UPPER(TRIM(CLIENTES.ADDRESS)) ELSE 'DESCONOCIDO'
END=DIR.DE_DIRECCION

        AND CASE WHEN TRIM(CLIENTES.POSTAL_CODE)<>'' THEN TRIM(CLIENTES.POSTAL_CODE) ELSE 99999
END=DIR.DE_CP

    AND CASE WHEN TRIM(CLIENTES.CITY)<>'' THEN CLIENTES.CITY ELSE 'DESCONOCIDO' END=DIR.DE_CIUDAD

    AND CASE WHEN TRIM(CLIENTES.STATE)<>'' THEN CLIENTES.STATE ELSE 'DESCONOCIDO' END=DIR.DE_ESTADO

    AND CASE WHEN TRIM(CLIENTES.COUNTRY)<>'' THEN CLIENTES.COUNTRY ELSE 'DESCONOCIDO'
END=DIR.DE_PAIS;


INSERT INTO ODS_HC_SERVICIOS

SELECT A.PRODUCT_ID ID_SERVICIO

, A.CUSTOMER_ID ID_CLIENTE

, C.ID_PRODUCTO ID_PRODUCTO

, A.ACCESS_POINT PUNTO_ACCESO

, D.ID_CANAL ID_CANAL

, E.ID_AGENTE_CC ID_AGENTE

, CASE WHEN TRIM(DIR.ID_DIRECCION)<>'' THEN DIR.ID_DIRECCION ELSE 999999 END ID_DIRECCION

, CASE WHEN TRIM(A.START_DATE)<>'' THEN STR_TO_DATE(A.START_DATE, '%d/%m/%Y %H-%i-%s') ELSE
STR_TO_DATE('31/12/9999','%d/%m/%Y %H-%i-%s') END FC_INICIO

, A.INSTALL_DATE FC_INSTALACION

, A.END_DATE FC_FIN

, NOW()

, STR_TO_DATE('31/12/9999','%d/%m/%Y')

FROM STAGE.STG_PRODUCTOS_CRM A

INNER JOIN ODS_DM_PRODUCTOS C ON A.PRODUCT_NAME=C.DE_PRODUCTO

INNER JOIN ODS_DM_CANALES D ON A.CHANNEL=D.DE_CANAL

INNER JOIN ODS_DM_AGENTES_CC E

        ON CASE WHEN TRIM(A.AGENT_CODE)<>'' THEN TRIM(A.AGENT_CODE) ELSE 9999 END=E.ID_AGENTE_CC

LEFT OUTER JOIN ODS.TMP_DIRECCIONES_SERVICIOS2 DIR ON DIR.ID_CLIENTE=A.CUSTOMER_ID;


COMMIT;


ANALYZE TABLE ODS_HC_SERVICIOS;

DROP TABLE IF EXISTS TMP_DIRECCIONES_SERVICIOS;

DROP TABLE IF EXISTS TMP_DIRECCIONES_SERVICIOS2;
```

Anejo 2 – Creación Modelo Facturas sentencias SQL

Sentencias SQL empleadas para la creación del modelo facturas:

Creación tablas FACTURAS

```
/* Tablas:
        - ODS_HC_CLIENTES

Ya creadas en modelo clientes */

USE ODS;

DROP TABLE IF EXISTS ODS_HC_FACTURAS
CREATE TABLE ODS_HC_FACTURAS
(ID_FACTURA INT(10) UNSIGNED AUTO_INCREMENT PRIMARY KEY
, ID_CLIENTE INT(10)
, FC_INICIO DATETIME
, FC_ESTADO DATETIME
, FC_PAGO DATETIME
, ID_CICLO_FATURACION INT(10)
, ID_METOD_PAGO INT(10)
, CANTIDAD       INT(10)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME);

DROP TABLE IF EXISTS ODS_DM_METODOS_PAGO
CREATE TABLE ODS_DM_METODOS_PAGO
(ID_METODO_PAGO INT(10) UNSIGNED 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(10) UNSIGNED AUTO_INCREMENT PRIMARY KEY
, DE_CICLO_FACTURACION VARCHAR(512)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME);
```

Creación claves FACTURAS

```
/* FK DEL MODELO FACTURAS */

ALTER TABLE ODS_HC_CLIENTES MODIFY COLUMN ID_CIENTE INT (10) UNSIGNED;
ALTER TABLE ODS_HC_CLIENTES ADD INDEX fk_cli_fact_idx (ID_CLIENTE ASC);
```

```
ALTER TABLE ODS.ODS_HC_CLIENTES ADD CONSTRAINT fk_cli_fact FOREIGN KEY (ID_CLIENTE)
        REFERENCES ODS.ODS_HC_FACTURAS (ID_CLIENTE)

ALTER TABLE ODS_HC_FACTURAS MODIFY COLUMN ID_CICLO_FACTURACION INT(10) UNSIGNED;
ALTER TABLE ODS.ODS_HC_FACTURAS ADD INDEX fk_fact_ciclofac_idx (ID_CICLO_FACTURACION ASC);
ALTER TABLE ODS.ODS_HC_FACTURAS ADD CONSTRAINT fk_serv_cli FOREIGN KEY (ID_CICLO_FACTURACION)
        REFERENCES ODS.ODS_DM_CICLOS_FACTURACION (ID_CICLO_FACTURACION);

ALTER TABLE ODS_HC_FACTURAS MODIFY COLUMN ID_METODO_PAGO INT(10) UNSIGNED;
ALTER TABLE ODS.ODS_HC_FACTURAS ADD INDEX fk_fact_metpago_idx (ID_METODO_PAGO ASC);
ALTER TABLE ODS.ODS_HC_FACTURAS ADD CONSTRAINT fk_fact_metpago FOREIGN KEY (ID_METODO_PAGO)
        REFERENCES ODS.ODS_DM_METODOS_PAGO (ID_METODO_PAGO)
```

Poblamos FACTURAS

```
/* Tabla:
        - ODS_HC_CLIENTES

Ya poblada en modelo clientes */

-- TABLA ODS_DM_METODOS_PAGO

INSERT INTO ODS_DM_METODOS_PAGO (DE_METODO_PAGO, FC_INSERT, FC_MODIFICACION)
SELECT DISTINCT UPPER(TRIM(BILL_METHOD)) BILL_METHOD, NOW(), NOW()
FROM STAGE.STG_FACTURAS_FCT
WHERE TRIM(BILL_METHOD)<>'';

COMMIT;

INSERT INTO ODS_DM_METODOS_PAGO VALUES (98, 'NO APLICA', NOW(), NOW());
INSERT INTO ODS_DM_METODOS_PAGO VALUES (99, 'DESCONOCIDO', NOW(), NOW());

ANALYZE TABLE ODS_DM_METODOS;

-- TABLA ODS_DM_CICLOS_FACTURACION

INSERT INTO ODS_DM_CICLOS_FACTURACION (DE_CICLO_FACTURACION, FC_INSERT, FC_MODIFICACION)
SELECT DISTINCT UPPER(TRIM(BILL_CYCLE)) BILL_CYCLE, NOW(), NOW()
FROM STAGE.STG_FACTURAS_FCT
WHERE TRIM(BILL_METHOD)<>'';

COMMIT;

INSERT INTO ODS_DM_CICLOS_FACTURACION (98, 'NO APLICA', NOW(), NOW());
```

```
INSERT INTO ODS_DM_CICLOS_FACTURACION (99, 'DESCONOCIDO', NOW(), NOW());


-- TABLA ODS_HC_FACTURAS


INSERT INTO ODS_HC_FACTURAS
SELECT BILL_REF_NO ID_FACTURA
, FACT_CUSTOMER_ID ID_CLIENTE
, STR_TO_DATE(START_DATE, '%Y-%m-%d %H:%i:%s') FC_INICIO
, STR_TO_DATE(END_DATE, '%Y-%m-%d %H:%i:%s') FC_FIN
, STR_TO_DATE(STATEMENT_DATE, '%Y-%m-%d %H:%i:%s') FC_ESTADO
, STR_TO_DATE(PAYMENT_DATE, '%Y-%m-%d %H:%i:%s') FC_PAGO
, CICLOS.ID_CICLO_FACTURACION
, PAGO.ID_METODO_PAGO
, ROUND(FACT.AMOUNT) CANTIDAD
FROM STG_FACTURAS_FCT FACT
INNER JOIN ODS_DM_CICLOS_FACTURACION CICLOS ON FACT.BILL_CYCLE=CICLOS.DE_CICLO_FACTURACION
INNER JOIN ODS_DM_METODOS_PAGO PAGO ON FACT.BILL_METHOD=PAGO.DE_METODO_PAGO
;
```

Anejo 3 – Creación Modelo Llamadas sentencias SQL

Sentencias SQL empleadas para la creación del modelo Llamadas:

Creación tablas Llamadas

```
/* Tablas:
        - ODS_HC_CLIENTES

Ya creadas en modelo clientes */

USE ODS;

DROP TABLE IF EXISTS ODS_HC_LLAMADAS;
CREATE TABLE ODS_HC_LLAMADAS
(ID_LLAMADA INT(10) UNSIGNED AUTO_INCREMENT PRIMARY KEY
, ID_IVR INT(10)
, TELEFONO_LLAMADA BIGINT(20)
, ID_CLIENTE INT(11)
, FC_INICIO_LLAMADA DATETIME
, FC_FIN_LLAMADA DATETIME
, ID_DEPARTAMENTO_CC INT(10)
, FLG_TRANSFERIDO TINYINT(1)
, ID_AGENTE_CC INT(10)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME
);

DROP TABLE IF EXISTS ODS_DM_DEPARTAMENTOS_CC;
CREATE TABLE ODS_DM_DEPARTAMENTOS_CC
(ID_DEPARTAMENTO_CC INT(10) UNSIGNED 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(10) UNSIGNED AUTO_INCREMENT PRIMARY KEY
, DE_AGENTE_CC INT(10)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME);
```

Creación claves Llamadas

```
/* FK DEL MODELO LLAMADAS */
```

```
ALTER TABLE ODS_HC_CLIENTES MODIFY COLUMN ID_CIENTE INT (10) UNSIGNED;
ALTER TABLE ODS_HC_CLIENTES ADD INDEX fk_cli_llam_idx (ID_CLIENTE ASC);
ALTER TABLE ODS.ODS_HC_CLIENTES ADD CONSTRAINT fk_cli_llam FOREIGN KEY (ID_CLIENTE)
        REFERENCES ODS.ODS_HC_LLAMADAS (ID_CLIENTE);

ALTER TABLE ODS_HC_LLAMADAS MODIFY COLUMN ID_DEPARTAMENTO_CC INT(10) UNSIGNED;
ALTER TABLE ODS_HC_LLAMADAS ADD INDEX fk_llam_dept_idx (ID_DEPARTAMENTO_CC ASC);
ALTER TABLE ODS_HC_LLAMADAS ADD CONSTRAINT fk_llam_dept FOREIGN KEY (ID_DEPARTAMENTO_CC)
        REFERENCES ODS.ODS_DM_DEPARTAMENTOS_CC (ID_DEPARTAMENTO_CC);

ALTER TABLE ODS_HC_LLAMADAS MODIFY COLUMN ID_AGENTE_CC INT(10) UNSIGNED;
ALTER TABLE ODS_HC_LLAMADAS ADD INDEX fk_llam_agen_idx (ID_AGENTE_CC ASC);
ALTER TABLE ODS_HC_LLAMADAS ADD CONSTRAINT fk_llam_agen FOREIGN KEY (ID_AGENTE_CC)
        REFERENCES ODS.ODS_DM_AGENTES_CC (ID_AGENTE_CC)
```

Poblamos Llamadas

```
/* Tabla:
        - ODS_HC_CLIENTES

Ya poblada en modelo clientes */

-- TABLA ODS_DM_DEPARTAMENTOS_CC

INSERT INTO ODS_DM_DEPARTAMENTOS_CC (DE_DEPARTAMENTO_CC, FC_INSERT, FC_MODIFICACION)
SELECT DISTINCT UPPER(TRIM(SERVICE)) DE_DEPARTAMENTO_CC, NOW(), NOW()
FROM STAGE.STG_CONTACTOS_IVR
WHERE TRIM(SERVICE);

INSERT INTO ODS_DM_DEPARTAMENTOS_CC VALUES (99, 'DESCONOCIDO', NOW(), NOW());
INSERT INTO ODS_DM_DEPARTAMENTOS_CC VALUES (98, 'NO APLICA', NOW(), NOW());

-- TABLA ODS_DM_AGENTES_CC
INSERT INTO ODS_DM_DEPARTAMENTOS_CC (DE_AGENTE_CC, FC_INSERT, FC_MODIFICACION)
SELECT DISTINCT UPPER(TRIM(AGENT)) DE_AGENTE_CC, NOW(), NOW()
FROM STAGE.STG_CONTACTOS_IVR
WHERE TRIM(AGENT);

INSERT INTO ODS_DM_DEPARTAMENTOS_CC (9998, 'NO APLICA', NOW(), NOW());
INSERT INTO ODS_DM_DEPARTAMENTOS_CC (9999, 'DESCONOCIDO', NOW(), NOW());

-- TABLA ODS_HC_LLAMADAS

INSERT INTO ODS_HC_LLAMADAS
```

```
SELECT ID

, CASE WHEN TRIM(A.PHONE_NUMBER)<>'' THEN TRIM(A.PHONE_NUMBER) ELSE 9999999999 END TELEFONO_LLAMADA

, CASE WHEN TRIM(B.CUSTOMER_ID)<>'' THEN TRIM(B.CUSTOMER_ID) ELSE 999999999 END  ID_CLIENTE

, CASE WHEN TRIM(A.START_DATETIME)<>'' THEN STR_TO_DATE(A.START_DATETIME, '%Y-%m-%d %H:%i:%s') ELSE
STR_TO_DATE('9999-12-31','%Y-%m-%d %H:%i:%s') END FC_INICIO_LLAMADA

, CASE WHEN TRIM(A.END_DATETIME)<>'' THEN STR_TO_DATE(A.END_DATETIME, '%Y-%m-%d %H:%i:%s') ELSE
STR_TO_DATE('9999-12-31','%Y-%m-%d %H:%i:%s') END FC_FIN_LLAMADA

, D.ID_DEPARTAMENTO_CC

, CASE

    WHEN A.FLG_TRANSFER = 'True' THEN 0

    WHEN A.FLG_TRANSFER = 'False' THEN 1

    ELSE 'UNDEF'

END  FLG_TRANSFERIDO

, E.ID_AGENTE_CC ID_AGENTE_CC

FROM STAGE.STG_CONTACTOS_IVR A

INNER JOIN STG_CLIENTES_CRM B

        ON A.PHONE_NUMBER=B.PHONE

INNER JOIN ODS_DM_DEPARTAMENTOS_CC D

        ON A.SERVICE = D.DE_DEPARTAMENTO_CC

INNER JOIN ODS_DM_AGENTES_CC E

        ON CASE WHEN TRIM(A.AGENT)<>'' THEN TRIM(A.AGENT) ELSE 9999 END=E.DE_AGENTE_CC;


INSERT INTO ODS_DM_DEPARTAMENTOS_CC (9999999999, 'DESCONOCIDO', NOW(), NOW());
```

Bibliografía

https://datawarehouse.es/2009/los-10-mandamientos-de-kimball.html

https://gravitar.biz/datawarehouse/mejores-practicas-para-construir-un-data-warehouse/

https://www.lluiscodina.com/como-disenar-una-base-de-datos-para-nuestro-proyecto-de-investigacion/

htthttp://www.vertabelo.com/blog/technical-articles/5-steps-for-an-effective-database-modelps://www.linkedin.com/pulse/tutorial-step-database-design-sql-david-mccaldin

https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/243583/El-pilar-de-la-gesti-n-de-datos-Data-Modeling-Design