5 DE ENERO DE 2018

PRÁCTICA
TALLER DATA 101

JORGE DEBRÁN PÉREZ

PRIMERA PARTE

ESTUDIO DE TABLAS

STG_PRODUCTOS_CRM

En primer lugar, para analizar la tabla utilizamos el fichero ESTUDIO STG_PRODUCTOS_CRM.sql:

```
USE STAGE;

SELECT
    COUNT(*) TOTAL_REGISTROS,

    SUM(CASE LENGTH(TRIM(PRODUCT_ID)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_PRODUCT_ID,
    COUNT(DISTINCT CASE LENGTH(TRIM(PRODUCT_ID)) WHEN 0 THEN 0 ELSE PRODUCT_ID
END) TOTAL_DISTINTOS_PRODUCT_ID,

    SUM(CASE LENGTH(TRIM(CUSTOMER_ID)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_CUSTOMER_ID,
    COUNT(DISTINCT CASE LENGTH(TRIM(CUSTOMER_ID)) WHEN 0 THEN 0 ELSE
CUSTOMER_ID END) TOTAL_DISTINTOS_CUSTOMER_ID,

    SUM(CASE LENGTH(TRIM(PRODUCT_NAME)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_PRODUCT_NAME,
    COUNT(DISTINCT CASE LENGTH(TRIM(PRODUCT_NAME)) WHEN 0 THEN 0 ELSE
PRODUCT_NAME END) TOTAL_DISTINTOS_PRODUCT_NAME,

    SUM(CASE LENGTH(TRIM(ACCESS_POINT)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_ACCESS_POINT,
    COUNT(DISTINCT CASE LENGTH(TRIM(ACCESS_POINT)) WHEN 0 THEN 0 ELSE
ACCESS_POINT END) TOTAL_DISTINTOS_ACCESS_POINT,

    SUM(CASE LENGTH(TRIM(CHANNEL)) WHEN 0 THEN 0 ELSE 1 END) TOTAL_CHANNEL,
    COUNT(DISTINCT CASE LENGTH(TRIM(CHANNEL)) WHEN 0 THEN 0 ELSE CHANNEL END)
TOTAL_DISTINTOS_CHANNEL,

    SUM(CASE LENGTH(TRIM(AGENT_CODE)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_AGENT_CODE,
    COUNT(DISTINCT CASE LENGTH(TRIM(AGENT_CODE)) WHEN 0 THEN 0 ELSE AGENT_CODE
END) TOTAL_DISTINTOS_AGENT_CODE,

    SUM(CASE LENGTH(TRIM(START_DATE)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_START_DATE,
    COUNT(DISTINCT CASE LENGTH(TRIM(START_DATE)) WHEN 0 THEN 0 ELSE START_DATE
END) TOTAL_DISTINTOS_START_DATE,

    SUM(CASE LENGTH(TRIM(INSTALL_DATE)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_INSTALL_DATE,
    COUNT(DISTINCT CASE LENGTH(TRIM(INSTALL_DATE)) WHEN 0 THEN 0 ELSE
INSTALL_DATE END) TOTAL_DISTINTOS_INSTALL_DATE,

    SUM(CASE LENGTH(TRIM(END_DATE)) WHEN 0 THEN 0 ELSE 1 END) TOTAL_END_DATE,
    COUNT(DISTINCT CASE LENGTH(TRIM(END_DATE)) WHEN 0 THEN 0 ELSE END_DATE
END) TOTAL_DISTINTOS_END_DATE,

    SUM(CASE LENGTH(TRIM(PRODUCT_CITY)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_PRODUCT_CITY,
    COUNT(DISTINCT CASE LENGTH(TRIM(PRODUCT_CITY)) WHEN 0 THEN 0 ELSE
PRODUCT_CITY END) TOTAL_DISTINTOS_PRODUCT_CITY,

    SUM(CASE LENGTH(TRIM(PRODUCT_ADDRESS)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_PRODUCT_ADDRESS,
    COUNT(DISTINCT CASE LENGTH(TRIM(PRODUCT_ADDRESS)) WHEN 0 THEN 0 ELSE
PRODUCT_ADDRESS END) TOTAL_DISTINTOS_PRODUCT_ADDRESS,
```

```
    SUM(CASE LENGTH(TRIM(PRODUCT_POSTAL_CODE)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_PRODUCT_POSTAL_CODE,
    COUNT(DISTINCT CASE LENGTH(TRIM(PRODUCT_POSTAL_CODE)) WHEN 0 THEN 0 ELSE
PRODUCT_POSTAL_CODE END) TOTAL_DISTINTOS_PRODUCT_POSTAL_CODE,

    SUM(CASE LENGTH(TRIM(PRODUCT_STATE)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_PRODUCT_STATE,
    COUNT(DISTINCT CASE LENGTH(TRIM(PRODUCT_STATE)) WHEN 0 THEN 0 ELSE
PRODUCT_STATE END) TOTAL_DISTINTOS_PRODUCT_STATE,

    SUM(CASE LENGTH(TRIM(PRODUCT_COUNTRY)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_PRODUCT_COUNTRY,
    COUNT(DISTINCT CASE LENGTH(TRIM(PRODUCT_COUNTRY)) WHEN 0 THEN 0 ELSE
PRODUCT_COUNTRY END) TOTAL_DISTINTOS_PRODUCT_COUNTRY

FROM STAGE.STG_PRODUCTOS_CRM;
```

Fichero 1. ESTUDIO STG_PRODUCTOS_CRM.sql

El resultado que se obtiene de la consulta es:

TOTAL_REGISTROS	78495
TOTAL_PRODUCT_ID	78495
TOTAL_DISTINTOS_PRODUCT_ID	78495
TOTAL_CUSTOMER_ID	78495
TOTAL_DISTINTOS_CUSTOMER_ID	8001
TOTAL_PRODUCT_NAME	78495
TOTAL_DISTINTOS_PRODUCT_NAME	6
TOTAL_ACCESS_POINT	78274
TOTAL_DISTINTOS_ACCESS_POINT	78275
TOTAL_CHANNEL	78274
TOTAL_DISTINTOS_CHANNEL	5
TOTAL_AGENT_CODE	42630
TOTAL_DISTINTOS_AGENT_CODE	701
TOTAL_START_DATE	78495
TOTAL_DISTINTOS_START_DATE	8035
TOTAL_INSTALL_DATE	75363
TOTAL_DISTINTOS_INSTALL_DATE	75360
TOTAL_END_DATE	46684
TOTAL_DISTINTOS_END_DATE	46683
TOTAL_PRODUCT_CITY	78274
TOTAL_DISTINTOS_PRODUCT_CITY	82
TOTAL_PRODUCT_ADDRESS	78274
TOTAL_DISTINTOS_PRODUCT_ADDRESS	77037
TOTAL_PRODUCT_POSTAL_CODE	78274
TOTAL_DISTINTOS_PRODUCT_POSTAL_CODE	274
TOTAL_PRODUCT_STATE	78090
TOTAL_DISTINTOS_PRODUCT_STATE	4
TOTAL_PRODUCT_COUNTRY	78274
TOTAL_DISTINTOS_PRODUCT_COUNTRY	2

Tabla 1. Resultados de ESTUDIO STG_PRODUCTOS_CRM.sql

Observando estos resultados sacamos las siguientes conclusiones:

- El modelo que generaremos de esta tabla lo podremos ligar con el modelo CLIENTES por medio del campo CUSTOMER_ID. Si analizamos más a fondo este cruce vemos que hay 28 registros que

no aparecen en el cruce y que estos 28 registros se corresponden con 3 CUSTOMER_ID. Dado que ambos datos provienen del mismo sistema (CRM) podemos pensar que se ha producido un error o bien al generar el cliente o bien al borrarle. Como nuestro ecosistema provee datos de más fuentes podremos investigar si estos clientes están en otras fuentes de datos, aunque viendo el porcentaje de error que supone estos datos (0,03%) no nos influye en nuestras métricas.

- Hay 221 registros que tienen los campos ACCESS_POINT, CHANNEL, AGENT_CODE, PRODUCT_CITY, PRODUCT_ADDRESS, PRODUCT_POSTAL_CODE y PRODUCT_COUNTRY sin rellenar. De estos 221 registros llama la atención que en 4 registros el campo PRODUCT_STATE también está sin rellenar. Además, podemos observar que el campo PRODUCT_STATE está sin rellenar en 405 ocasiones. El porcentaje de error que implican estos registros es de un 0,5% por lo que no nos influye en nuestras métricas.

- El campo AGENT_CODE en algunos casos esta sin rellenar, pero en este caso está asociado al campo CHANNEL cuando tiene el valor “App” o “Online” por lo que podemos pensar que no son un error.

- El campo INSTALL_DATE está sin rellenar en 3132 registros lo que supone casi un 4%. Además, si nos fijamos el resto de campos cuando esta sin rellenar no se observa ningún patrón. Es un porcentaje suficiente como para que nos afecta a nuestras métricas por lo que deberemos ver que con la gente del operacional (CRM) de si se trata de un error o son valores que están correctamente.

- El campo END_DATE está sin rellenar en 31811 registros lo que supone casi un 40%. Dado el porcentaje podemos asumir que estos son los servicios que están actualmente activos, no estaría de más asegurar esta suposición con la gente del operacional (CRM). Además, si nos fijamos en los valores que tiene asociados vemos que 2333 registros son valores en el futuro, podríamos pensar que se debe a servicios contratos que tienen fecha de baja del servicio. Al igual que con la suposición anterior, deberíamos de aclarar estos valores con la gente del operacional (CRM).

- Para acabar, vemos que de esta tabla podemos generar 1 tabla de hechos y 2 de dimensiones:

    - SERVICIOS: hechos donde guardaremos los productos contratados por los clientes.

    - PRODUCTOS: dimensión donde guardaremos los diferentes productos que se pueden contratar.

    - CANALES: dimensión donde guardaremos los canales por donde se puede contratar los diferentes productos.

  Además de asociarse con el modelo CLIENTES a través de la tabla de hechos de CLIENTES, podremos agregar direcciones nuevas a la tabla de hechos de DIRECCIONES.

STG_FACTURAS_FCT

En primer lugar, para analizar la tabla utilizamos el fichero ESTUDIO STG_FACTURAS_FCT.sql:

```
USE STAGE;

SELECT
    COUNT(*) TOTAL_REGISTROS,

    SUM(CASE LENGTH(TRIM(BILL_REF_NO)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_BILL_REF_NO,
    COUNT(DISTINCT CASE LENGTH(TRIM(BILL_REF_NO)) WHEN 0 THEN 0 ELSE
BILL_REF_NO END) TOTAL_DISTINTOS_BILL_REF_NO,

    SUM(CASE LENGTH(TRIM(CUSTOMER_ID)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_CUSTOMER_ID,
    COUNT(DISTINCT CASE LENGTH(TRIM(CUSTOMER_ID)) WHEN 0 THEN 0 ELSE
CUSTOMER_ID END) TOTAL_DISTINTOS_CUSTOMER_ID,
```

```
    SUM(CASE LENGTH(TRIM(START_DATE)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_START_DATE,
    COUNT(DISTINCT CASE LENGTH(TRIM(START_DATE)) WHEN 0 THEN 0 ELSE START_DATE
END) TOTAL_DISTINTOS_START_DATE,

    SUM(CASE LENGTH(TRIM(END_DATE)) WHEN 0 THEN 0 ELSE 1 END) TOTAL_END_DATE,
    COUNT(DISTINCT CASE LENGTH(TRIM(END_DATE)) WHEN 0 THEN 0 ELSE END_DATE
END) TOTAL_DISTINTOS_END_DATE,

    SUM(CASE LENGTH(TRIM(STATEMENT_DATE)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_STATEMENT_DATE,
    COUNT(DISTINCT CASE LENGTH(TRIM(STATEMENT_DATE)) WHEN 0 THEN 0 ELSE
STATEMENT_DATE END) TOTAL_DISTINTOS_STATEMENT_DATE,

    SUM(CASE LENGTH(TRIM(PAYMENT_DATE)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_PAYMENT_DATE,
    COUNT(DISTINCT CASE LENGTH(TRIM(PAYMENT_DATE)) WHEN 0 THEN 0 ELSE
PAYMENT_DATE END) TOTAL_DISTINTOS_PAYMENT_DATE,

    SUM(CASE LENGTH(TRIM(BILL_CYCLE)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_BILL_CYCLE,
    COUNT(DISTINCT CASE LENGTH(TRIM(BILL_CYCLE)) WHEN 0 THEN 0 ELSE BILL_CYCLE
END) TOTAL_DISTINTOS_BILL_CYCLE,

    SUM(CASE LENGTH(TRIM(AMOUNT)) WHEN 0 THEN 0 ELSE 1 END) TOTAL_AMOUNT,
    COUNT(DISTINCT CASE LENGTH(TRIM(AMOUNT)) WHEN 0 THEN 0 ELSE AMOUNT END)
TOTAL_DISTINTOS_AMOUNT,

    SUM(CASE LENGTH(TRIM(BILL_METHOD)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_BILL_METHOD,
    COUNT(DISTINCT CASE LENGTH(TRIM(BILL_METHOD)) WHEN 0 THEN 0 ELSE
BILL_METHOD END) TOTAL_DISTINTOS_BILL_METHOD

FROM STAGE.STG_FACTURAS_FCT;
```

Fichero 2. ESTUDIO STG_FACTURAS_FCT.sql

El resultado que se obtiene de la consulta es:

TOTAL_REGISTROS	420000
TOTAL_BILL_REF_NO	420000
TOTAL_DISTINTOS_BILL_REF_NO	420000
TOTAL_CUSTOMER_ID	420000
TOTAL_DISTINTOS_CUSTOMER_ID	20000
TOTAL_START_DATE	420000
TOTAL_DISTINTOS_START_DATE	40
TOTAL_END_DATE	420000
TOTAL_DISTINTOS_END_DATE	20
TOTAL_STATEMENT_DATE	420000
TOTAL_DISTINTOS_STATEMENT_DATE	40
TOTAL_PAYMENT_DATE	420000
TOTAL_DISTINTOS_PAYMENT_DATE	400
TOTAL_BILL_CYCLE	420000
TOTAL_DISTINTOS_BILL_CYCLE	2
TOTAL_AMOUNT	420000
TOTAL_DISTINTOS_AMOUNT	5604
TOTAL_BILL_METHOD	420000
TOTAL_DISTINTOS_BILL_METHOD	3

Tabla 2. Resultados de ESTUDIO STG_FACTURAS_FCT.sql

Observando estos resultados sacamos las siguientes conclusiones:

- El modelo que generaremos de esta tabla lo podremos ligar con el modelo CLIENTES y con el modelo SERVICIOS por medio del campo CUSTOMER_ID. Si analizamos más a fondo estos cruces vemos que hay 48840 registros que corresponden con 2442 CUSTOMER_ID que no están en el modelo de CLIENTES y 240000 registros que corresponden con 12000 CUSTOMER_ID que no están en el modelo de SERVICIOS. Esto supone un 12% de facturas registradas de las que no tenemos información de cliente y un 57% de facturas registradas de las que no tenemos información del producto que se ha facturado. Son porcentajes muy preocupantes ya que estamos facturando a clientes de los que no tenemos información y a clientes que si tenemos información pero no de que tengan productos contratados, por lo que habría que ver con la gente del operacional tanto del CRM como del FACTURADOR para ver que está sucediendo.

- Por lo que vemos todos los campos están rellenos en todos los registros, lo único que se observa es que los campos START_DATE, STATEMENT_DATE y PAYMENT_DATE en un principio se guardaban en formato DATETIME.

- Para acabar, vemos que de esta tabla podemos generar 1 tabla de hechos y 2 de dimensiones:

    - FACTURAS: hechos donde guardaremos las facturas generadas.

    - CICLOS_FACTURACION: dimensión donde guardaremos los diferentes ciclos de facturación.

    - METODOS_PAGO: dimensión donde guardaremos los diferentes métodos de pago.

STG_CONTACTOS_IVR

En primer lugar, para analizar la tabla utilizamos el fichero ESTUDIO STG_CONTACTOS_IVR.sql:

```
USE STAGE;

SELECT
    COUNT(*) TOTAL_REGISTROS,

    SUM(CASE LENGTH(TRIM(ID)) WHEN 0 THEN 0 ELSE 1 END) TOTAL_ID,
    COUNT(DISTINCT CASE LENGTH(TRIM(ID)) WHEN 0 THEN 0 ELSE ID END)
TOTAL_DISTINTOS_ID,

    SUM(CASE LENGTH(TRIM(PHONE_NUMBER)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_PHONE_NUMBER,
    COUNT(DISTINCT CASE LENGTH(TRIM(PHONE_NUMBER)) WHEN 0 THEN 0 ELSE
PHONE_NUMBER END) TOTAL_DISTINTOS_PHONE_NUMBER,

    SUM(CASE LENGTH(TRIM(START_DATETIME)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_START_DATETIME,
    COUNT(DISTINCT CASE LENGTH(TRIM(START_DATETIME)) WHEN 0 THEN 0 ELSE
START_DATETIME END) TOTAL_DISTINTOS_START_DATETIME,

    SUM(CASE LENGTH(TRIM(END_DATETIME)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_END_DATETIME,
    COUNT(DISTINCT CASE LENGTH(TRIM(END_DATETIME)) WHEN 0 THEN 0 ELSE
END_DATETIME END) TOTAL_DISTINTOS_END_DATETIME,

    SUM(CASE LENGTH(TRIM(SERVICE)) WHEN 0 THEN 0 ELSE 1 END) TOTAL_SERVICE,
    COUNT(DISTINCT CASE LENGTH(TRIM(SERVICE)) WHEN 0 THEN 0 ELSE SERVICE END)
TOTAL_DISTINTOS_SERVICE,

    SUM(CASE LENGTH(TRIM(FLG_TRANSFER)) WHEN 0 THEN 0 ELSE 1 END)
TOTAL_FLG_TRANSFER,
    COUNT(DISTINCT CASE LENGTH(TRIM(FLG_TRANSFER)) WHEN 0 THEN 0 ELSE
FLG_TRANSFER END) TOTAL_DISTINTOS_FLG_TRANSFER,

    SUM(CASE LENGTH(TRIM(AGENT)) WHEN 0 THEN 0 ELSE 1 END) TOTAL_AGENT,
```

```
    COUNT(DISTINCT CASE LENGTH(TRIM(AGENT)) WHEN 0 THEN 0 ELSE AGENT END)
TOTAL_DISTINTOS_AGENT

FROM STAGE.STG_CONTACTOS_IVR;
```

Fichero 3. ESTUDIO STG_CONTACTOS_IVR.sql

El resultado que se obtiene de la consulta es:

TOTAL_REGISTROS	202717
TOTAL_ID	202717
TOTAL_DISTINTOS_ID	150000
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

Tabla 3. Resultados de ESTUDIO STG_CONTACTOS_IVR.sql

Observando estos resultados sacamos las siguientes conclusiones:

- El modelo que generaremos de esta tabla lo podremos ligar con el modelo CLIENTES por medio del campo PHONE_NUMBER. Cuando realizamos el cruce vemos que no hay ningún cruce entre ambas tablas por lo que deberemos aplicar el valor DESCONOCIDO hasta que tengamos alguna forma de poder asociar ambos modelos.

- Lo primero que observamos en la tabla es que el campo ID no es un identificador único, además parece estar asociado al campo FLG_TRANSFER ya que cuando su valor es True coincide el valor del campo END_DATETIME con el valor del campo START_DATATIME del siguiente registro. Si observamos el valor del campo PHONE_NUMBER cuando el campo ID coincide, el valor del campo PHONE_NUMBER no se mantiene por lo que podríamos pensar que este es el valor del PHONE_NUMBER destino y no origen; y por ello no hemos podido establecer una conexión con el modelo CLIENTES.

- Hay 39221 registros que tienen el campo PHONE_NUMBER, END_DATETIME, SERVICE o AGENT sin rellenar. Esto supone un 19% de todos los registros pero sin más información de los datos que guarda el operacional (IVR) no podemos indicar si son un error o los datos son correctos.

- El campo AGENT es un campo de texto con valores que parecen ser usuarios de agentes. El campo AGENT_CODE en la tabla STG_PRODUCTOS_CRM es un campo numérico por lo que tampoco podemos realizar un cruce entre ambos datos.

- Para acabar, vemos que de esta tabla podemos generar 1 tabla de hechos y 2 de dimensiones:

    - LLAMADAS: hechos donde guardamos todos los registros que produce la IVR.

    - DEPARTAMENTOS_CC: dimensión donde guardamos los diferentes departamentos del Call Center.

    - AGENTES_CC: dimensión donde guardamos los diferentes agentes del Call Center.

STG_ORDERS_CRM

En primer lugar, para analizar la tabla utilizamos el fichero ESTUDIO STG_ORDERS_CRM.sql:

```
USE STAGE;

SELECT
    COUNT(*) TOTAL_REGISTROS,

    SUM(CASE LENGTH(TRIM(ID)) WHEN 0 THEN 0 ELSE 1 END) TOTAL_ID,
    COUNT(DISTINCT CASE LENGTH(TRIM(ID)) WHEN 0 THEN 0 ELSE ID END)
TOTAL_DISTINTOS_ID,

    SUM(CASE LENGTH(TRIM(ORD.ORDER)) WHEN 0 THEN 0 ELSE 1 END) TOTAL_ORDER,
    COUNT(DISTINCT CASE LENGTH(TRIM(ORD.ORDER)) WHEN 0 THEN 0 ELSE ORD.ORDER
END) TOTAL_DISTINTOS_ORDER,

    SUM(CASE LENGTH(TRIM(PHASE)) WHEN 0 THEN 0 ELSE 1 END) TOTAL_PHASE,
    COUNT(DISTINCT CASE LENGTH(TRIM(PHASE)) WHEN 0 THEN 0 ELSE PHASE END)
TOTAL_DISTINTOS_PHASE,

    SUM(CASE LENGTH(TRIM(AGENT)) WHEN 0 THEN 0 ELSE 1 END) TOTAL_AGENT,
    COUNT(DISTINCT CASE LENGTH(TRIM(AGENT)) WHEN 0 THEN 0 ELSE AGENT END)
TOTAL_DISTINTOS_AGENT,

    SUM(CASE LENGTH(TRIM(START_DT)) WHEN 0 THEN 0 ELSE 1 END) TOTAL_START_DT,
    COUNT(DISTINCT CASE LENGTH(TRIM(START_DT)) WHEN 0 THEN 0 ELSE START_DT
END) TOTAL_DISTINTOS_START_DT,

    SUM(CASE LENGTH(TRIM(END_DT)) WHEN 0 THEN 0 ELSE 1 END) TOTAL_END_DT,
    COUNT(DISTINCT CASE LENGTH(TRIM(END_DT)) WHEN 0 THEN 0 ELSE END_DT END)
TOTAL_DISTINTOS_END_DT

FROM STAGE.STG_ORDERS_CRM ORD;
```

Fichero 4. ESTUDIO STG_ORDERS_CRM.sql

El resultado que se obtiene de la consulta es:

TOTAL_REGISTROS	360067
TOTAL_ID	360067
TOTAL_DISTINTOS_ID	324081
TOTAL_ORDER	360067
TOTAL_DISTINTOS_ORDER	78000
TOTAL_PHASE	360067
TOTAL_DISTINTOS_PHASE	7
TOTAL_AGENT	360032
TOTAL_DISTINTOS_AGENT	101
TOTAL_START_DT	360067
TOTAL_DISTINTOS_START_DT	342069
TOTAL_END_DT	282067
TOTAL_DISTINTOS_END_DT	270383

Tabla 4. Resultados de ESTUDIO STG_ORDERS_CRM.sql

Observando estos resultados sacamos las siguientes conclusiones:

- El modelo que generaremos de esta tabla lo podremos ligar con el modelo SERVICIOS por medio del campo ORDER. Si analizamos más a fondo este cruce vemos que hay 495 registros que no aparecen en el cruce. Dado que ambos datos provienen del mismo sistema (CRM) podemos pensar que se ha producido un error a la hora provisionar esos servicios. El porcentaje que supone estos registros es de un 0,63% por lo que no nos influye en nuestras métricas, pero

deberemos hablar con la gente del operacional (CRM) para ver si hemos perdido servicios por provisionar.

- Lo primero que observamos en la tabla es que el campo ID no es un identificador único, además parece estar asociado al campo ORDER pero en algunos casos el campo ID y el campo ORDER no se mantiene lo cual podría indicar un error en el operacional (CRM). También se observa que cuando ambos coinciden se produce un cambio en el campo PHASE y ambos registros también están asociados mediante el valor del campo END_DT del primer registro y el valor del campo START_DT del segundo.

- Hay 35 registros que tiene el campo AGENT sin rellenar, lo cual produce un error que no nos afecta, pero del que debemos de informar al operacional (CRM). Además, vemos que los valores son similares a los vistos en el modelo LLAMADAS. Sin embargo, cuando realizamos el cruce solo 1 registro es coincidente por lo que no podemos establecer relación.

- Hay 78000 registros que tiene el campo END_DT sin rellenar, lo cual es un 21% de los registros pero no podemos indicar sin son registros erróneos o correctos.

- Para acabar, vemos que de esta tabla podemos generar 1 tabla de hechos y 2 de dimensiones:

    - PROVISION: hechos donde guardaremos los diferentes registros que produce el provisionamiento de los servicios.

    - FASES: dimensión donde guardaremos las diferentes fases por las que pasa un servicio.

    - AGENTES_PRV: dimensión donde guardamos los diferentes agentes de provisionamiento.

CREAR EL MODELO, LAS TABLAS, LAS FK Y POBLAR LOS MODELOS

SERVICIOS

El modelo generado es el siguiente:

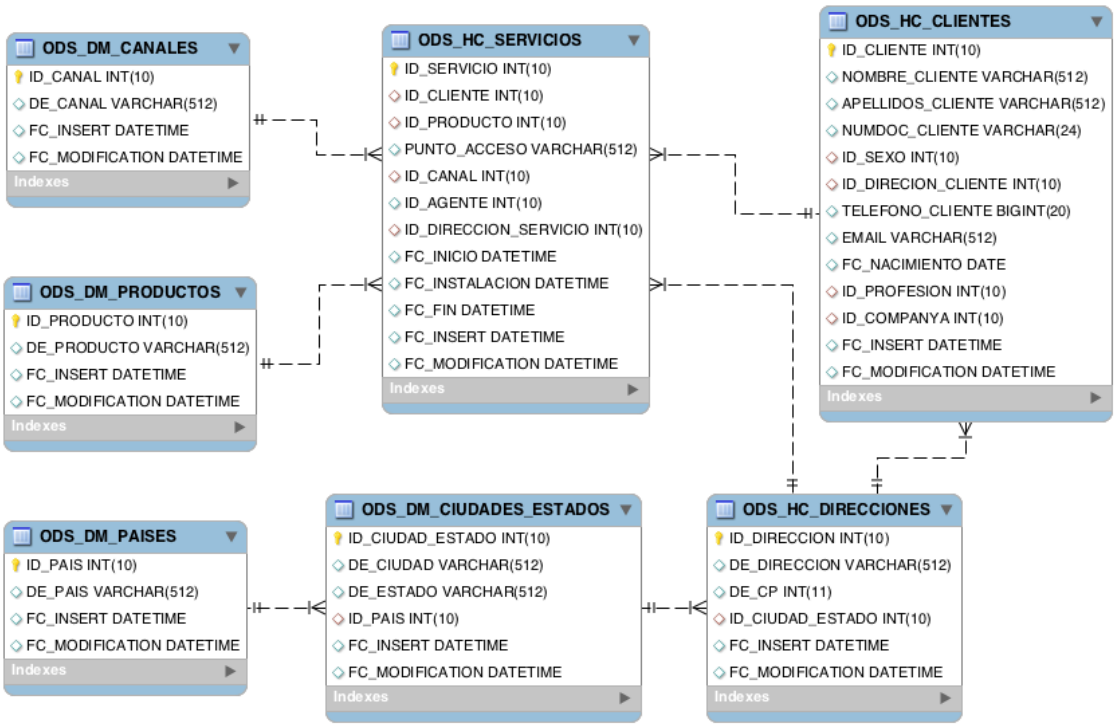

Ilustración 1. Modelo SERVICIOS

En primer lugar, crearemos las tablas para ello utilizamos el fichero MODELO SERVICIOS.sql:

```
USE ODS;

DROP TABLE IF EXISTS ODS_HC_SERVICIOS;

CREATE TABLE ODS_HC_SERVICIOS (
    ID_SERVICIO INT UNSIGNED NOT NULL PRIMARY KEY,
    ID_CLIENTE INT UNSIGNED,
    ID_PRODUCTO INT UNSIGNED,
    PUNTO_ACCESO VARCHAR(512),
    ID_CANAL INT UNSIGNED,
    ID_AGENTE INT UNSIGNED,
    ID_DIRECCION_SERVICIO INT UNSIGNED,
    FC_INICIO DATETIME,
    FC_INSTALACION DATETIME,
    FC_FIN DATETIME,
    FC_INSERT DATETIME,
    FC_MODIFICATION DATETIME
);

DROP TABLE IF EXISTS ODS_DM_PRODUCTOS;

CREATE TABLE ODS_DM_PRODUCTOS (
    ID_PRODUCTO INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    DE_PRODUCTO VARCHAR(512),
    FC_INSERT DATETIME,
    FC_MODIFICATION DATETIME
);

DROP TABLE IF EXISTS ODS_DM_CANALES;
```

```
CREATE TABLE ODS_DM_CANALES (
    ID_CANAL INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    DE_CANAL VARCHAR(512),
    FC_INSERT DATETIME,
    FC_MODIFICATION DATETIME
);
```

Fichero 5. MODELO SERVICIOS.sql

A continuación, generamos las FK del modelo utilizando el fichero FK SERVICIOS.sql:

```
USE ODS;

ALTER TABLE ODS_HC_SERVICIOS ADD INDEX fk_ser_cli_idx (ID_CLIENTE ASC);
ALTER TABLE ODS_HC_SERVICIOS ADD CONSTRAINT fk_ser_cli FOREIGN KEY
(ID_CLIENTE) REFERENCES ODS_HC_CLIENTES (ID_CLIENTE);

ALTER TABLE ODS_HC_SERVICIOS ADD INDEX fk_ser_pro_idx (ID_PRODUCTO ASC);
ALTER TABLE ODS_HC_SERVICIOS ADD CONSTRAINT fk_ser_pro FOREIGN KEY
(ID_PRODUCTO) REFERENCES ODS_DM_PRODUCTOS (ID_PRODUCTO);

ALTER TABLE ODS_HC_SERVICIOS ADD INDEX fk_ser_can_idx (ID_CANAL ASC);
ALTER TABLE ODS_HC_SERVICIOS ADD CONSTRAINT fk_ser_can FOREIGN KEY (ID_CANAL)
REFERENCES ODS_DM_CANALES (ID_CANAL);

ALTER TABLE ODS_HC_SERVICIOS ADD INDEX fk_ser_dir_idx (ID_DIRECCION_SERVICIO
ASC);
ALTER TABLE ODS_HC_SERVICIOS ADD CONSTRAINT fk_ser_dir FOREIGN KEY
(ID_DIRECCION_SERVICIO) REFERENCES ODS_HC_DIRECCIONES (ID_DIRECCION);
```

Fichero 6. FK SERVICIOS.sql

Por último, poblamos el modelo utilizando el fichero POBLAR SERVICIOS.sql:

```
USE ODS;

INSERT INTO ODS_DM_CANALES (DE_CANAL, FC_INSERT, FC_MODIFICATION)
SELECT DISTINCT UPPER(TRIM(CHANNEL)) AS CANAL, NOW(), NOW()
FROM STAGE.STG_PRODUCTOS_CRM
WHERE LENGTH(TRIM(CHANNEL))<>0
ORDER BY CANAL;

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

COMMIT;

ANALYZE TABLE ODS_DM_CANALES;


INSERT INTO ODS_DM_PRODUCTOS (DE_PRODUCTO, FC_INSERT, FC_MODIFICATION)
SELECT DISTINCT UPPER(TRIM(PRODUCT_NAME)) AS PRODUCTO, NOW(), NOW()
FROM STAGE.STG_PRODUCTOS_CRM
WHERE LENGTH(TRIM(PRODUCT_NAME))<>0
ORDER BY PRODUCTO;

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

COMMIT;

ANALYZE TABLE ODS_DM_PRODUCTOS;


INSERT INTO ODS_DM_PAISES (DE_PAIS, FC_INSERT, FC_MODIFICATION)
SELECT DISTINCT UPPER(TRIM(PRODUCT_COUNTRY)) AS PAIS, NOW(), NOW()
FROM STAGE.STG_PRODUCTOS_CRM
LEFT OUTER JOIN ODS_DM_PAISES PAI ON
    CASE WHEN LENGTH(TRIM(PRODUCT_COUNTRY))<>0 THEN REPLACE(PRODUCT_COUNTRY,
```

```
'United States', 'US') ELSE 'DESCONOCIDO' END = PAI.DE_PAIS
WHERE LENGTH(TRIM(PRODUCT_COUNTRY))<>0
AND PAI.ID_PAIS IS NULL
ORDER BY PAIS;

COMMIT;

ANALYZE TABLE ODS_DM_PAISES;


INSERT INTO ODS_DM_CIUDADES_ESTADOS (DE_CIUDAD, DE_ESTADO, ID_PAIS, FC_INSERT,
FC_MODIFICATION)
SELECT DISTINCT CASE WHEN TRIM(PRODUCT_CITY)<>'' THEN
UPPER(TRIM(PRODUCT_CITY)) ELSE 'DESCONOCIDO' END CIUDAD,
CASE WHEN TRIM(PRODUCT_STATE)<>'' THEN UPPER(TRIM(PRODUCT_STATE)) ELSE
'DESCONOCIDO' END ESTADO,
PAI.ID_PAIS, NOW(), NOW()
FROM STAGE.STG_PRODUCTOS_CRM PRO
INNER JOIN ODS_DM_PAISES PAI ON CASE WHEN LENGTH(TRIM(PRO.PRODUCT_COUNTRY))<>0
THEN REPLACE(PRO.PRODUCT_COUNTRY, 'United States', 'US') ELSE 'DESCONOCIDO'
END=PAI.DE_PAIS
LEFT OUTER JOIN ODS_DM_CIUDADES_ESTADOS CIU ON
    CASE WHEN TRIM(PRODUCT_CITY)<>'' THEN UPPER(TRIM(PRODUCT_CITY)) ELSE
'DESCONOCIDO' END = CIU.DE_CIUDAD
    AND CASE WHEN TRIM(PRODUCT_STATE)<>'' THEN UPPER(TRIM(PRODUCT_STATE)) ELSE
'DESCONOCIDO' END = CIU.DE_ESTADO
    AND PAI.ID_PAIS = CIU.ID_PAIS
WHERE (TRIM(PRODUCT_CITY)<>'' OR TRIM(PRODUCT_STATE)<>'')
AND CIU.ID_CIUDAD_ESTADO IS NULL;

COMMIT;

ANALYZE TABLE ODS_DM_CIUDADES_ESTADOS;


INSERT INTO ODS_HC_DIRECCIONES (DE_DIRECCION, DE_CP, ID_CIUDAD_ESTADO,
FC_INSERT, FC_MODIFICATION)
SELECT DISTINCT UPPER(TRIM(PRODUCT_ADDRESS)) DIRECCION,
CASE WHEN LENGTH(TRIM(PRO.PRODUCT_POSTAL_CODE))<>0 THEN
TRIM(PRO.PRODUCT_POSTAL_CODE) ELSE 99999 END CP,
CIU.ID_CIUDAD_ESTADO, NOW(), NOW()
FROM STAGE.STG_PRODUCTOS_CRM PRO
INNER JOIN ODS_DM_PAISES PAI ON CASE WHEN LENGTH(TRIM(PRO.PRODUCT_COUNTRY))<>0
THEN REPLACE(PRO.PRODUCT_COUNTRY, 'United States', 'US') ELSE 'DESCONOCIDO'
END=PAI.DE_PAIS
INNER JOIN ODS_DM_CIUDADES_ESTADOS CIU ON CASE WHEN
LENGTH(TRIM(PRO.PRODUCT_CITY))<>0 THEN PRO.PRODUCT_CITY ELSE 'DESCONOCIDO'
END=CIU.DE_CIUDAD
                                        AND CASE WHEN
LENGTH(TRIM(PRO.PRODUCT_STATE))<>0 THEN PRO.PRODUCT_STATE ELSE 'DESCONOCIDO'
END=CIU.DE_ESTADO
LEFT OUTER JOIN ODS_HC_DIRECCIONES DIR ON
    CASE WHEN TRIM(PRODUCT_ADDRESS)<>'' THEN UPPER(TRIM(PRODUCT_ADDRESS)) ELSE
'DESCONOCIDO' END = DIR.DE_DIRECCION
    AND CASE WHEN TRIM(PRODUCT_POSTAL_CODE)<>'' THEN
UPPER(TRIM(PRODUCT_POSTAL_CODE)) ELSE 'DESCONOCIDO' END = DIR.DE_CP
    AND CIU.ID_CIUDAD_ESTADO = DIR.ID_CIUDAD_ESTADO
WHERE TRIM(PRODUCT_ADDRESS)<>''
AND DIR.ID_CIUDAD_ESTADO IS NULL;

COMMIT;

ANALYZE TABLE ODS_HC_DIRECCIONES;


INSERT INTO ODS_HC_CLIENTES
SELECT DISTINCT UPPER(TRIM(CUSTOMER_ID)) ID_CLIENTE,
'DESCONOCIDO', 'DESCONOCIDO', '99-999-9999', 99, 999999, 9999999999,
```

```
'DESCONOCIDO', STR_TO_DATE('31/12/9999', '%d/%m/%Y'), 999, 999, NOW(),
STR_TO_DATE('31/12/9999', '%d/%m/%Y')
FROM STAGE.STG_PRODUCTOS_CRM PRO
LEFT OUTER JOIN ODS_HC_CLIENTES CLI ON
    CASE WHEN TRIM(CUSTOMER_ID)<>'' THEN UPPER(TRIM(CUSTOMER_ID)) ELSE
'DESCONOCIDO' END = CLI.ID_CLIENTE
WHERE TRIM(PRODUCT_ADDRESS)<>''
AND CLI.ID_CLIENTE IS NULL;

COMMIT;

ANALYZE TABLE ODS_HC_CLIENTES;


INSERT INTO ODS_HC_SERVICIOS
SELECT
    PRODUCT_ID AS ID_SERVICIO,
    CLI.ID_CLIENTE,
    PRO.ID_PRODUCTO,
    ACCESS_POINT AS PUNTO_ACCESO,
    CAN.ID_CANAL,
    CASE WHEN LENGTH(TRIM(AGENT_CODE))<>0 THEN TRIM(AGENT_CODE) ELSE 9998 END
ID_AGENTE,
    CASE WHEN LENGTH(TRIM(DIR.ID_DIRECCION))<>0 THEN DIR.ID_DIRECCION ELSE
999999 END ID_DIRECCION,
    CASE WHEN LENGTH(TRIM(START_DATE))<>0 THEN STR_TO_DATE(START_DATE,
'%d/%m/%Y') ELSE STR_TO_DATE('31/12/9999', '%d/%m/%Y') END FC_INICIO,
    CASE WHEN LENGTH(TRIM(INSTALL_DATE))<>0 THEN
STR_TO_DATE(SUBSTRING(INSTALL_DATE, 1, 19),'%Y-%m-%d %H:%i:%s') ELSE
STR_TO_DATE('31/12/9998', '%d/%m/%Y') END FC_INSTALACION,
    CASE WHEN LENGTH(TRIM(END_DATE))<>0 THEN STR_TO_DATE(SUBSTRING(END_DATE,
1, 19),'%Y-%m-%d %H:%i:%s') ELSE STR_TO_DATE('31/12/9998', '%d/%m/%Y') END
FC_FIN,
    NOW(),
    STR_TO_DATE('31/12/9999', '%d/%m/%Y')

FROM
    STAGE.STG_PRODUCTOS_CRM PRODUCTOS

INNER JOIN
       ODS_DM_PRODUCTOS PRO ON CASE WHEN LENGTH(TRIM(PRODUCT_NAME))<>0 THEN
UPPER(TRIM(PRODUCTOS.PRODUCT_NAME)) ELSE 'DESCONOCIDO' END=PRO.DE_PRODUCTO
INNER JOIN
       ODS_DM_CANALES CAN ON CASE WHEN LENGTH(TRIM(CHANNEL))<>0 THEN
UPPER(TRIM(PRODUCTOS.CHANNEL)) ELSE 'DESCONOCIDO' END=CAN.DE_CANAL
LEFT OUTER JOIN
       ODS_HC_CLIENTES CLI ON CASE WHEN LENGTH(TRIM(CUSTOMER_ID))<>0 THEN
TRIM(CUSTOMER_ID) ELSE 999999999 END=CLI.ID_CLIENTE
LEFT OUTER JOIN (
       ODS_HC_DIRECCIONES DIR INNER JOIN ODS_DM_CIUDADES_ESTADOS CIU ON
DIR.ID_CIUDAD_ESTADO=CIU.ID_CIUDAD_ESTADO
       INNER JOIN ODS_DM_PAISES PAI ON CIU.ID_PAIS=PAI.ID_PAIS
) ON
       CASE WHEN TRIM(PRODUCT_ADDRESS)<>'' THEN
UPPER(TRIM(PRODUCTOS.PRODUCT_ADDRESS)) ELSE 'DESCONOCIDO' END=DIR.DE_DIRECCION
       AND CASE WHEN TRIM(PRODUCTOS.PRODUCT_POSTAL_CODE)<>'' THEN
UPPER(TRIM(PRODUCTOS.PRODUCT_POSTAL_CODE)) ELSE 99999 END=DIR.DE_CP
       AND CASE WHEN TRIM(PRODUCTOS.PRODUCT_CITY)<>'' THEN
UPPER(TRIM(PRODUCTOS.PRODUCT_CITY)) ELSE 'DESCONOCIDO' END=CIU.DE_CIUDAD
       AND CASE WHEN TRIM(PRODUCTOS.PRODUCT_STATE)<>'' THEN
UPPER(TRIM(PRODUCTOS.PRODUCT_STATE)) ELSE 'DESCONOCIDO' END=CIU.DE_ESTADO
       AND CASE WHEN TRIM(PRODUCTOS.PRODUCT_COUNTRY)<>'' THEN
UPPER(TRIM(PRODUCTOS.PRODUCT_COUNTRY)) ELSE 'DESCONOCIDO' END=PAI.DE_PAIS;

COMMIT;

ANALYZE TABLE ODS_HC_SERVICIOS;
```

Fichero 7.POBLAR SERVICIOS.sql

FACTURAS

El modelo generado es el siguiente:

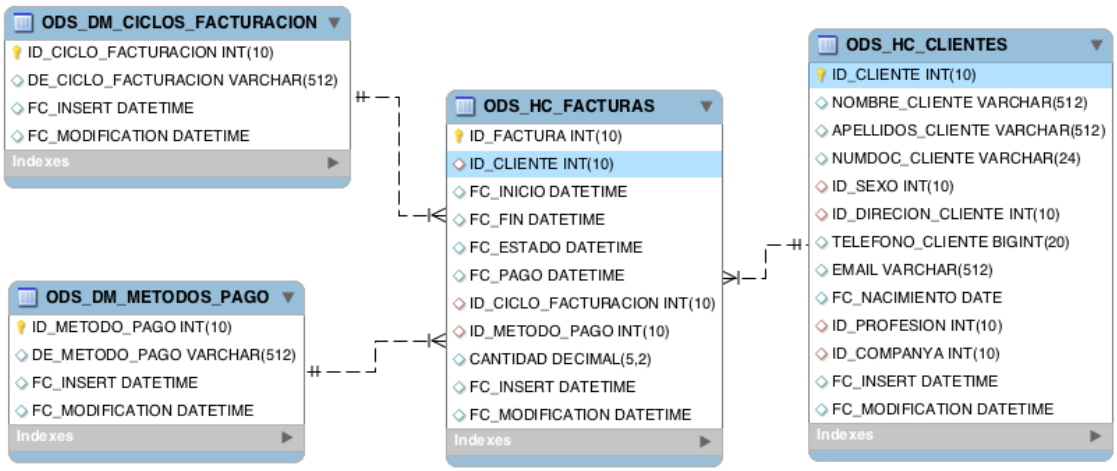

Ilustración 2. Modelo FACTURAS

En primer lugar, crearemos las tablas para ello utilizamos el fichero MODELO FACTURAS.sql:

```
USE ODS;

DROP TABLE IF EXISTS ODS_HC_FACTURAS;

CREATE TABLE ODS_HC_FACTURAS (
    ID_FACTURA INT UNSIGNED NOT NULL PRIMARY KEY,
    ID_CLIENTE INT UNSIGNED,
    FC_INICIO DATETIME,
    FC_FIN DATETIME,
    FC_ESTADO DATETIME,
    FC_PAGO DATETIME,
    ID_CICLO_FACTURACION INT UNSIGNED,
    ID_METODO_PAGO INT UNSIGNED,
    CANTIDAD DECIMAL(5,2),
    FC_INSERT DATETIME,
    FC_MODIFICATION DATETIME
);

DROP TABLE IF EXISTS ODS_DM_METODOS_PAGO;

CREATE TABLE ODS_DM_METODOS_PAGO (
    ID_METODO_PAGO INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    DE_METODO_PAGO VARCHAR(512),
    FC_INSERT DATETIME,
    FC_MODIFICATION DATETIME
);

DROP TABLE IF EXISTS ODS_DM_CICLOS_FACTURACION;

CREATE TABLE ODS_DM_CICLOS_FACTURACION (
    ID_CICLO_FACTURACION INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    DE_CICLO_FACTURACION VARCHAR(512),
    FC_INSERT DATETIME,
    FC_MODIFICATION DATETIME
);
```

Fichero 8. MODELO FACTURAS.sql

A continuación, generamos las FK del modelo utilizando el fichero FK FACTURAS.sql:

```
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 ADD INDEX fk_fac_mp_idx (ID_METODO_PAGO ASC);
ALTER TABLE ODS_HC_FACTURAS ADD CONSTRAINT fk_fac_mp FOREIGN KEY
(ID_METODO_PAGO) REFERENCES ODS_DM_METODOS_PAGO (ID_METODO_PAGO);

ALTER TABLE ODS_HC_FACTURAS ADD INDEX fk_fac_cf_idx (ID_CICLO_FACTURACION
ASC);
ALTER TABLE ODS_HC_FACTURAS ADD CONSTRAINT fk_fac_cf FOREIGN KEY
(ID_CICLO_FACTURACION) REFERENCES ODS_DM_CICLOS_FACTURACION
(ID_CICLO_FACTURACION);
```

Fichero 9. FK FACTURAS.sql

Por último, poblamos el modelo utilizando el fichero POBLAR FACTURAS.sql:

```
USE ODS;

INSERT INTO ODS_DM_CICLOS_FACTURACION (DE_CICLO_FACTURACION, FC_INSERT,
FC_MODIFICATION)
SELECT DISTINCT UPPER(TRIM(BILL_CYCLE)) AS CICLO_FACTURACION, NOW(), NOW()
FROM STAGE.STG_FACTURAS_FCT
WHERE LENGTH(TRIM(BILL_CYCLE))<>0
ORDER BY CICLO_FACTURACION;

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

COMMIT;

ANALYZE TABLE ODS_DM_CICLOS_FACTURACION;


INSERT INTO ODS_DM_METODOS_PAGO (DE_METODO_PAGO, FC_INSERT, FC_MODIFICATION)
SELECT DISTINCT UPPER(TRIM(BILL_METHOD)) AS METODO_PAGO, NOW(), NOW()
FROM STAGE.STG_FACTURAS_FCT
WHERE LENGTH(TRIM(BILL_METHOD))<>0
ORDER BY METODO_PAGO;

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

COMMIT;

ANALYZE TABLE ODS_DM_METODOS_PAGO;

INSERT INTO ODS_HC_CLIENTES VALUES (999999999,'DESCONOCIDO',
'DESCONOCIDO','99-999-9999', 99, 999999, 9999999999, 'DESCONOCIDO',
STR_TO_DATE('31/12/9999', '%d/%m/%Y'), 999, 999, NOW(),
STR_TO_DATE('31/12/9999', '%d/%m/%Y'));

COMMIT;

ANALYZE TABLE ODS_HC_CLIENTES;



INSERT INTO ODS_HC_FACTURAS
SELECT
    BILL_REF_NO AS ID_FACTURA,
    CASE WHEN LENGTH(TRIM(CLI.ID_CLIENTE))<>0 THEN TRIM(CLI.ID_CLIENTE) ELSE
999999999 END ID_CLIENTE,
    CASE WHEN LENGTH(TRIM(START_DATE))<>0 THEN
```

```
STR_TO_DATE(SUBSTRING(START_DATE, 1, 19),'%Y-%m-%d %H:%i:%s') ELSE
STR_TO_DATE('31/12/9999', '%d/%m/%Y') END FC_INICIO,
    CASE WHEN LENGTH(TRIM(END_DATE))<>0 THEN STR_TO_DATE(SUBSTRING(END_DATE,
1, 19),'%Y-%m-%d %H:%i:%s') ELSE STR_TO_DATE('31/12/9999', '%d/%m/%Y') END
FC_FIN,
    CASE WHEN LENGTH(TRIM(STATEMENT_DATE))<>0 THEN
STR_TO_DATE(SUBSTRING(STATEMENT_DATE, 1, 19),'%Y-%m-%d %H:%i:%s') ELSE
STR_TO_DATE('31/12/9999', '%d/%m/%Y') END FC_ESTADO,
    CASE WHEN LENGTH(TRIM(PAYMENT_DATE))<>0 THEN
STR_TO_DATE(SUBSTRING(PAYMENT_DATE, 1, 19),'%Y-%m-%d %H:%i:%s') ELSE
STR_TO_DATE('31/12/9999', '%d/%m/%Y') END FC_PAGO,
    CI_FA.ID_CICLO_FACTURACION,
    ME_PA.ID_METODO_PAGO,
    CASE WHEN LENGTH(TRIM(AMOUNT))<>0 THEN CAST(AMOUNT AS DECIMAL(5,2)) ELSE
'99999.99' END CANTIDAD,
    NOW(),
    STR_TO_DATE('31/12/9999', '%d/%m/%Y')

FROM
    STAGE.STG_FACTURAS_FCT FACTURAS

INNER JOIN
       ODS_DM_CICLOS_FACTURACION CI_FA ON CASE WHEN LENGTH(TRIM(BILL_CYCLE))<>0
THEN UPPER(TRIM(FACTURAS.BILL_CYCLE)) ELSE 'DESCONOCIDO'
END=CI_FA.DE_CICLO_FACTURACION
INNER JOIN
       ODS_DM_METODOS_PAGO ME_PA ON CASE WHEN LENGTH(TRIM(BILL_METHOD))<>0 THEN
UPPER(TRIM(FACTURAS.BILL_METHOD)) ELSE 'DESCONOCIDO' END=ME_PA.DE_METODO_PAGO
LEFT OUTER JOIN
       ODS_HC_CLIENTES CLI ON CASE WHEN LENGTH(TRIM(CUSTOMER_ID))<>0 THEN
TRIM(CUSTOMER_ID) ELSE 999999999 END=CLI.ID_CLIENTE;

COMMIT;

ANALYZE TABLE ODS_HC_FACTURAS;
```

Fichero 10. POBLAR FACTURAS.sql

LLAMADAS

El modelo generado es el siguiente:

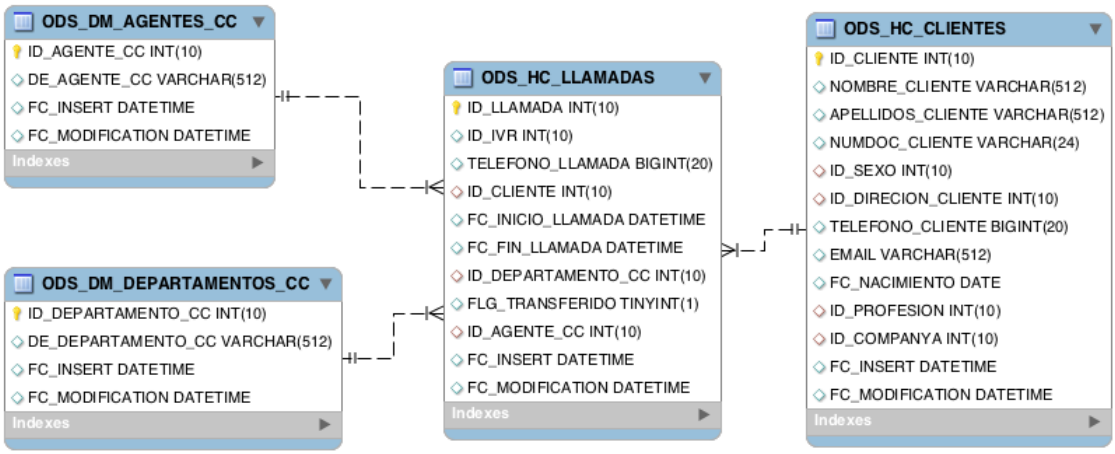

Ilustración 3. Modelo LLAMADAS

En primer lugar, crearemos las tablas, para ello utilizamos el fichero MODELO LLAMADAS.sql:

```
USE ODS;

DROP TABLE IF EXISTS ODS_HC_LLAMADAS;
```

```
CREATE TABLE ODS_HC_LLAMADAS (
    ID_LLAMADA INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    ID_IVR INT UNSIGNED,
    TELEFONO_LLAMADA BIGINT,
    ID_CLIENTE INT UNSIGNED,
    FC_INICIO_LLAMADA DATETIME,
    FC_FIN_LLAMADA DATETIME,
    ID_DEPARTAMENTO_CC INT UNSIGNED,
    FLG_TRANSFERIDO TINYINT(1),
    ID_AGENTE_CC INT UNSIGNED,
    FC_INSERT DATETIME,
    FC_MODIFICATION DATETIME
);

DROP TABLE IF EXISTS ODS_DM_DEPARTAMENTOS_CC;

CREATE TABLE ODS_DM_DEPARTAMENTOS_CC (
    ID_DEPARTAMENTO_CC INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    DE_DEPARTAMENTO_CC VARCHAR(512),
    FC_INSERT DATETIME,
    FC_MODIFICATION DATETIME
);

DROP TABLE IF EXISTS ODS_DM_AGENTES_CC;

CREATE TABLE ODS_DM_AGENTES_CC (
    ID_AGENTE_CC INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    DE_AGENTE_CC VARCHAR(512),
    FC_INSERT DATETIME,
    FC_MODIFICATION DATETIME
);
```

Fichero 11. MODELO LLAMADAS.sql

A continuación, generamos las FK del modelo utilizando el fichero FK LLAMADAS.sql:

```
USE ODS;

ALTER TABLE ODS_HC_LLAMADAS ADD INDEX fk_lla_cli_idx (ID_CLIENTE ASC);
ALTER TABLE ODS_HC_LLAMADAS ADD CONSTRAINT fk_lla_cli FOREIGN KEY (ID_CLIENTE)
REFERENCES ODS_HC_CLIENTES (ID_CLIENTE);

ALTER TABLE ODS_HC_LLAMADAS ADD INDEX fk_lla_dep_idx (ID_DEPARTAMENTO_CC ASC);
ALTER TABLE ODS_HC_LLAMADAS ADD CONSTRAINT fk_dep_pro FOREIGN KEY
(ID_DEPARTAMENTO_CC) REFERENCES ODS_DM_DEPARTAMENTOS_CC (ID_DEPARTAMENTO_CC);

ALTER TABLE ODS_HC_LLAMADAS ADD INDEX fk_lla_age_idx (ID_AGENTE_CC ASC);
ALTER TABLE ODS_HC_LLAMADAS ADD CONSTRAINT fk_age_can FOREIGN KEY
(ID_AGENTE_CC) REFERENCES ODS_DM_AGENTES_CC (ID_AGENTE_CC);
```

Fichero 12. FK LLAMADAS.sql

Por último, poblamos el modelo utilizando el fichero POBLAR LLAMADAS.sql:

```
USE ODS;

INSERT INTO ODS_DM_DEPARTAMENTOS_CC (DE_DEPARTAMENTO_CC, FC_INSERT,
FC_MODIFICATION)
SELECT DISTINCT UPPER(TRIM(SERVICE)) AS DEPARTAMENTO_CC, NOW(), NOW()
FROM STAGE.STG_CONTACTOS_IVR
WHERE LENGTH(TRIM(SERVICE))<>0
ORDER BY DEPARTAMENTO_CC;

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

COMMIT;

ANALYZE TABLE ODS_DM_DEPARTAMENTOS_CC;
```

```
INSERT INTO ODS_DM_AGENTES_CC (DE_AGENTE_CC, FC_INSERT, FC_MODIFICATION)
SELECT DISTINCT UPPER(TRIM(AGENT)) AS AGENTE_CC, NOW(), NOW()
FROM STAGE.STG_CONTACTOS_IVR
WHERE LENGTH(TRIM(AGENT))<>0
ORDER BY AGENTE_CC;

INSERT INTO ODS_DM_AGENTES_CC VALUES (999, 'DESCONOCIDO', NOW(), NOW());
INSERT INTO ODS_DM_AGENTES_CC VALUES (998, 'NO APLICA', NOW(), NOW());

COMMIT;

ANALYZE TABLE ODS_DM_AGENTES_CC;


INSERT INTO ODS_HC_LLAMADAS (ID_IVR, TELEFONO_LLAMADA, ID_CLIENTE,
FC_INICIO_LLAMADA, FC_FIN_LLAMADA, ID_DEPARTAMENTO_CC, FLG_TRANSFERIDO,
ID_AGENTE_CC, FC_INSERT, FC_MODIFICATION)
SELECT
    ID AS ID_IVR,
    CASE WHEN LENGTH(TRIM(PHONE_NUMBER))<>0 THEN TRIM(PHONE_NUMBER) ELSE
9999999999 END TELEFONO_LLAMADA,
    CASE WHEN LENGTH(TRIM(CLI.ID_CLIENTE))<>0 THEN TRIM(CLI.ID_CLIENTE) ELSE
999999999 END ID_CLIENTE,
    CASE WHEN LENGTH(TRIM(START_DATETIME))<>0 THEN
STR_TO_DATE(SUBSTRING(START_DATETIME, 1, 19),'%Y-%m-%d %H:%i:%s') ELSE
STR_TO_DATE('31/12/9999', '%d/%m/%Y') END FC_INICIO_LLAMADA,
    CASE WHEN LENGTH(TRIM(END_DATETIME))<>0 THEN
STR_TO_DATE(SUBSTRING(END_DATETIME, 1, 19),'%Y-%m-%d %H:%i:%s') ELSE
STR_TO_DATE('31/12/9999', '%d/%m/%Y') END FC_FIN_LLAMADA,
    DEP_CC.ID_DEPARTAMENTO_CC AS ID_DEPARTAMENTO_CC,
    CASE UPPER(TRIM(FLG_TRANSFER)) WHEN 'TRUE' THEN 1 ELSE 0 END
FLG_TRANSFERIDO,
    AGE_CC.ID_AGENTE_CC AS ID_AGENTE_CC,
    NOW(),
    STR_TO_DATE('31/12/9999', '%d/%m/%Y')

FROM
    STAGE.STG_CONTACTOS_IVR CONTACTOS

INNER JOIN
       ODS_DM_DEPARTAMENTOS_CC DEP_CC ON CASE WHEN LENGTH(TRIM(SERVICE))<>0
THEN UPPER(TRIM(CONTACTOS.SERVICE)) ELSE 'DESCONOCIDO'
END=DEP_CC.DE_DEPARTAMENTO_CC
INNER JOIN
       ODS_DM_AGENTES_CC AGE_CC ON CASE WHEN LENGTH(TRIM(AGENT))<>0 THEN
UPPER(TRIM(CONTACTOS.AGENT)) ELSE 'DESCONOCIDO' END=AGE_CC.DE_AGENTE_CC
LEFT OUTER JOIN
       ODS_HC_CLIENTES CLI ON CASE WHEN LENGTH(TRIM(PHONE_NUMBER))<>0 THEN
TRIM(PHONE_NUMBER) ELSE 999999999 END=CLI.TELEFONO_CLIENTE;

COMMIT;

ANALYZE TABLE ODS_HC_LLAMADAS;
```

Fichero 13. POBLAR LLAMADAS.sql

PROVISION

El modelo generado es el siguiente:

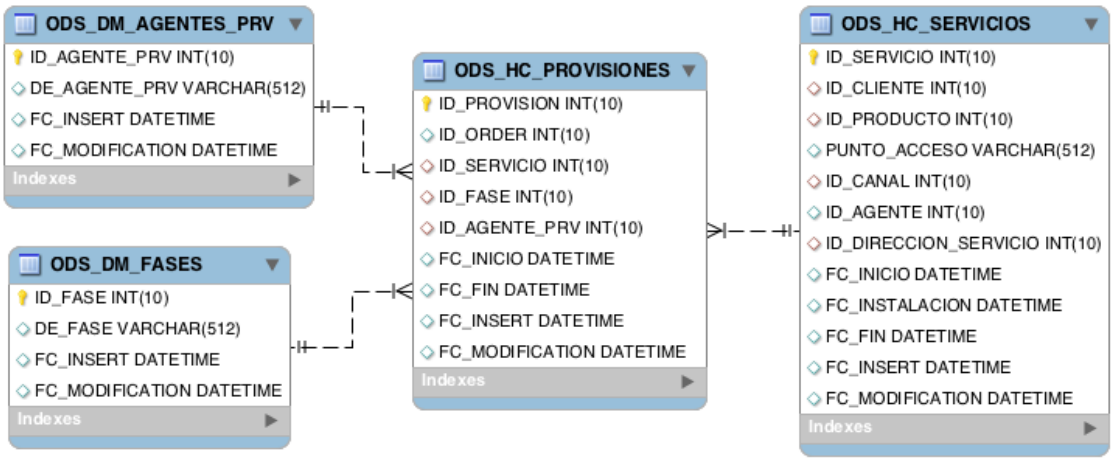

Ilustración 4. Modelo PROVISION

En primer lugar, crearemos las tablas para ello utilizamos el fichero MODELO PROVISION.sql:

```
USE ODS;

DROP TABLE IF EXISTS ODS_HC_PROVISIONES;

CREATE TABLE ODS_HC_PROVISIONES (
    ID_PROVISION INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    ID_ORDER INT UNSIGNED,
    ID_SERVICIO INT UNSIGNED,
    ID_FASE INT UNSIGNED,
    ID_AGENTE_PRV INT UNSIGNED,
    FC_INICIO DATETIME,
    FC_FIN DATETIME,
    FC_INSERT DATETIME,
    FC_MODIFICATION DATETIME
);

DROP TABLE IF EXISTS ODS_DM_FASES;

CREATE TABLE ODS_DM_FASES (
    ID_FASE INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    DE_FASE VARCHAR(512),
    FC_INSERT DATETIME,
    FC_MODIFICATION DATETIME
);

DROP TABLE IF EXISTS ODS_DM_AGENTES_PRV;

CREATE TABLE ODS_DM_AGENTES_PRV (
    ID_AGENTE_PRV INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    DE_AGENTE_PRV VARCHAR(512),
    FC_INSERT DATETIME,
    FC_MODIFICATION DATETIME
);
```

Fichero 14. MODELO PROVISION.sql

A continuación, generamos las FK del modelo utilizando el fichero FK PROVISION.sql:

```
USE ODS;

ALTER TABLE ODS_HC_PROVISIONES ADD INDEX fk_pro_ser_idx (ID_SERVICIO ASC);
ALTER TABLE ODS_HC_PROVISIONES ADD CONSTRAINT fk_pro_ser FOREIGN KEY
(ID_SERVICIO) REFERENCES ODS_HC_SERVICIOS (ID_SERVICIO);

ALTER TABLE ODS_HC_PROVISIONES ADD INDEX fk_pro_fas_idx (ID_FASE ASC);
ALTER TABLE ODS_HC_PROVISIONES ADD CONSTRAINT fk_pro_fas FOREIGN KEY (ID_FASE)
```

```
REFERENCES ODS_DM_FASES (ID_FASE);

ALTER TABLE ODS_HC_PROVISIONES ADD INDEX fk_pro_age_idx (ID_AGENTE_PRV ASC);
ALTER TABLE ODS_HC_PROVISIONES ADD CONSTRAINT fk_pro_age FOREIGN KEY
(ID_AGENTE_PRV) REFERENCES ODS_DM_AGENTES_PRV (ID_AGENTE_PRV);
```

Fichero 15. FK PROVISION.sql

Por último, poblamos el modelo utilizando el fichero POBLAR PROVISION.sql:

```
USE ODS;

INSERT INTO ODS_DM_FASES (DE_FASE, FC_INSERT, FC_MODIFICATION)
SELECT DISTINCT UPPER(TRIM(PHASE)) AS FASE, NOW(), NOW()
FROM STAGE.STG_ORDERS_CRM
WHERE LENGTH(TRIM(PHASE))<>0
ORDER BY FASE;

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

COMMIT;

ANALYZE TABLE ODS_DM_FASES;


INSERT INTO ODS_DM_AGENTES_PRV (DE_AGENTE_PRV, FC_INSERT, FC_MODIFICATION)
SELECT DISTINCT UPPER(TRIM(AGENT)) AS AGENTE_PRV, NOW(), NOW()
FROM STAGE.STG_ORDERS_CRM
WHERE LENGTH(TRIM(AGENT))<>0
ORDER BY AGENTE_PRV;

INSERT INTO ODS_DM_AGENTES_PRV VALUES (999, 'DESCONOCIDO', NOW(), NOW());
INSERT INTO ODS_DM_AGENTES_PRV VALUES (998, 'NO APLICA', NOW(), NOW());

COMMIT;

ANALYZE TABLE ODS_DM_AGENTES_PRV;


INSERT INTO ODS_HC_PROVISIONES (ID_ORDER, ID_SERVICIO, ID_FASE, ID_AGENTE_PRV,
FC_INICIO, FC_FIN, FC_INSERT, FC_MODIFICATION)
SELECT
    ID AS ID_ORDER,
    SER.ID_SERVICIO AS ID_SERVICIO,
    FAS.ID_FASE AS ID_FASE,
    AGE_PRV.ID_AGENTE_PRV AS ID_AGENTE_PRV,
    CASE WHEN LENGTH(TRIM(START_DT))<>0 THEN STR_TO_DATE(SUBSTRING(START_DT,
1, 19),'%Y-%m-%d %H:%i:%s') ELSE STR_TO_DATE('31/12/9999', '%d/%m/%Y') END
FC_INICIO,
    CASE WHEN LENGTH(TRIM(END_DT))<>0 THEN STR_TO_DATE(SUBSTRING(END_DT, 1,
19),'%Y-%m-%d %H:%i:%s') ELSE STR_TO_DATE('31/12/9999', '%d/%m/%Y') END
FC_FIN,
    NOW(),
    STR_TO_DATE('31/12/9999', '%d/%m/%Y')

FROM
    STAGE.STG_ORDERS_CRM ORDERS

INNER JOIN
      ODS_DM_FASES FAS ON CASE WHEN LENGTH(TRIM(PHASE))<>0 THEN
UPPER(TRIM(ORDERS.PHASE)) ELSE 'DESCONOCIDO' END=FAS.DE_FASE
INNER JOIN
      ODS_DM_AGENTES_PRV AGE_PRV ON CASE WHEN LENGTH(TRIM(AGENT))<>0 THEN
UPPER(TRIM(ORDERS.AGENT)) ELSE 'DESCONOCIDO' END=AGE_PRV.DE_AGENTE_PRV
LEFT OUTER JOIN
      ODS_HC_SERVICIOS SER ON CASE WHEN LENGTH(TRIM(ORDERS.ORDER))<>0 THEN
TRIM(ORDERS.ORDER) ELSE 999999999 END=SER.ID_SERVICIO;
```

```
COMMIT;

ANALYZE TABLE ODS_HC_PROVISIONES;
```

Fichero 16. POBLAR PROVISION.sql

SEGUNDA PARTE

DIAGRAMA ODS (NÚMERO DE REGISTROS EN CADA TABLA)

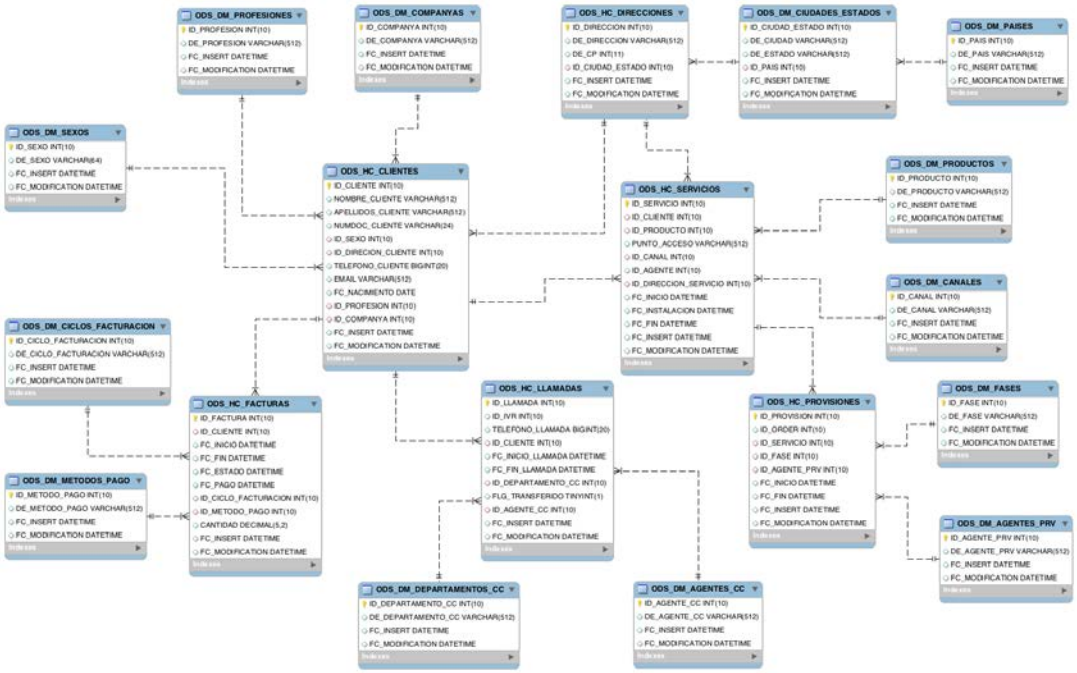

Ilustración 5. Diagrama ODS

Cuando hemos realizado la población de los diferentes modelos, hemos tenido en cuenta el origen de los datos para establecer si en los cruces se aplicaba los valores de “NO APLICA”, “DESCONOCIDO” o generar un nuevo registro con la información de una de las partes.

El número de registros que contiene cada tabla es el siguiente:

ODS_DM_AGENTES_CC	595
ODS_DM_AGENTES_PRV	102
ODS_DM_CANALES	6
ODS_DM_CICLOS_FACTURACION	4
ODS_DM_CIUDADES_ESTADOS	158
ODS_DM_COMPANYAS	385
ODS_DM_DEPARTAMENTOS_CC	8
ODS_DM_FASES	9
ODS_DM_METODOS_PAGO	5
ODS_DM_PAISES	3
ODS_DM_PRODUCTOS	8
ODS_DM_PROFESIONES	197
ODS_DM_SEXOS	4
ODS_HC_CLIENTES	17562
ODS_HC_DIRECCIONES	95766
ODS_HC_FACTURAS	420000

ODS_HC_LLAMADAS	202717
ODS_HC_PROVISIONES	360067
ODS_HC_SERVICIOS	78495

Tabla 5. Número de registros por tabla

¿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?

Cuando realizamos el estudio observamos que en nuestra base de datos existe una ciudad cuyo nombre está en dos estados. Este caso es el de la ciudad Glendale que en nuestra base de datos esta tanto en el estado de California como en el de Arizona.

Puede ser un caso aislado, por ello realizamos una búsqueda de una base de datos de ciudades y estados de Estados Unidos y observamos que es algo habitual. A continuación, las 10 ciudades que más se repiten en diferentes estados:

Washington	88
Springfield	41
Franklin	35
Lebanon	34
Clinton	30
Greenville	31
Bristol	29
Fairview	28
Salem	26
Madison	24

Tabla 6. Ciudades y número de Estados

¿SERÍAS CAPAZ DE SEPARAR EL CAMPO DE_DIRECCION DE LA TABLA DE DIRECCIONES EN DOS CAMPOS: NOMBRE_VIA Y NUM_VIA?

Observamos que el campo DE_DIRECCION podemos separar el NOMBRE_VIA y NUM_VIA buscando el primer espacio dentro de la cadena de texto. Para poder realizar esta separación podemos ejecutar el siguiente código SQL:

```
SELECT

CASE DE_DIRECCION
    WHEN 'NO APLICA' THEN 99998
    WHEN 'DESCONOCIDO' THEN 99999
    ELSE SUBSTRING_INDEX(DE_DIRECCION, ' ', 1) END NUM_VIA,
CASE DE_DIRECCION
    WHEN 'NO APLICA' THEN 'NO APLICA'
    WHEN 'DESCONOCIDO' THEN 'DESCONOCIDO'
    ELSE SUBSTRING(DE_DIRECCION, LENGTH(SUBSTRING_INDEX(DE_DIRECCION, ' ', 1))
+ 2) END 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

Si nos centramos en las diferentes dimensiones de la calidad de los datos podemos decir que no tenemos datos duplicados dentro de una misma tabla. Además, pese que a todos los datos estaban guardados en formato texto, hemos podidos distinguir el tipo de dato de cada campo dentro de las tablas.

Por otro lado, hemos visto que hay registros con campos sin rellenar en diferentes tablas. En algunos casos esto lo podríamos evitar utilizando valores por defecto y en otros casos será interesante tener dentro de Master Data tablas donde tener los posibles valores para los campos de las tablas de los operacionales. Un claro ejemplo de esto último es una base de datos con un callejero teniendo los valores de calles, ciudades y estados, de esta forma evitamos errores a la hora de rellenar los campos o dejarles en blanco. Además, haciendo esto les dará una exactitud a los datos ya que podremos verificar que son certeros. También será necesario comprobar que datos que hemos visto sin rellenar no son incorrectos y que se producen por la operativa del operacional y así darles precisión, como es el campo INSTALL_DATE dentro de la tabla PRODUCTOS del operacional del CRM.

Por último, hemos visto que la coherencia y la integridad referencial en algunos casos no se puede establecer. Esto se puede deber a que los propios operacionales tienen diferentes orígenes y no se ha establecido una conexión entre ellos o que no nos hemos traído todos los datos. El ejemplo más claro es la relación entre IVR y CRM donde intentamos establecer un cruce pero el campo PHONE_NUMBER de la IVR no parece ser el teléfono origen. Otro ejemplo de los errores de coherencia son clientes que tenemos en el operacional del FACTURADOR y no les tenemos en el operacional del CRM; o más grave dentro del propio operacional del CRM, en la tabla de PRODUCTOS tenemos clientes que no están en la tabla CLIENTES.

MASTER DATA

Hemos podido observar que hay datos que son utilizados en diferentes sistemas, incluso datos dentro de un mismo sistema que se guardan en diferentes tablas con valores distintos.

En primer lugar, podríamos crear un callejero donde tener calles, ciudades, estados y país. Con ello evitaríamos ciudades de las que desconocemos su estado, o no guardar el país cuando nuestro mercado parece que solo se centra en United States, o no guardar el país de dos formas diferentes: “US” o “United States”.

Otro caso sería guardar en una tabla maestra los agentes ya que dentro del operacional CRM en la tabla de PRODUCTOS se guardan con un código y la tabla ORDERS se guardan con lo que parece un valor de usuario. Además, tenemos en el operacional IVR el campo agente guardado con un valor de usuario, aunque como vimos el cruce entre IVR y la tabla ORDERS del CRM solo cruza un usuario deberemos distinguir entre diferentes tipos de agentes.

DATA MODELING & DESIGN

Hemos podido ver como los operacionales no están diseñados utilizando la tercera forma normal (3FN). Esto nos hubiera quitado muchos quebraderos de cabeza ya que estarían establecidas las Primary Key (PK) y las Foreign Key (FK). Con ello hubiéramos visto de una pasada como se relacionan las tablas o evitar que campos definidos como ID no estén repetidos y sean identificadores únicos.

Además, se podría haber establecido los diferentes tipos de datos en vez de guardar todo en formato varchar(512). Incluso establecer que campos pueden tener valor nulo o establecer un valor por defecto para algunos campos.

OPCIONAL: ¿ACONSEJARÍAS ALGÚN CAMBIO EN LOS SISTEMAS ORIGEN EXTRA TENIENDO EN CUENTA EL RESTO DE DISCIPLINAS DEL DATA GOVERNANCE?

Desde el punto de vista de la trazabilidad de los datos vemos como el operacional IVR cada iteración tenemos una fecha de inicio y fin y el agente asociado a esa iteración. Esto mismo sucede dentro del operacional CRM para la tabla ORDERS. Dentro del CRM la tabla PRODUCTOS no tenemos forma de ver si se produce alguna actualización sobre los registros.

Por otro lado, tenemos en el operacional FACTURADOR del que no sabemos quién emite esos registros ni si se producen actualizaciones sobre los mismos. Al igual sucede dentro del operacional CRM la tabla CLIENTES no sabemos ni quien crea o actualiza los campos ni cuando han sido creados.

OPCIONAL++: UTILIZANDO ALGUNA HERRAMIENTA DEL MERCADO O INVENTÁNDOTE UN MODELO EN BBDD GENERA LA TRAZABILIDAD DE LA INFORMACIÓN (DATA GOVERNANCE).

Existen multitud de herramientas en el mercado como las de Informática, IBM, Teradata, Oracle, SAP, etc... de las cuales desconocemos su funcionamiento.

Por ello, para poder hacer una traza de la forma más sencilla posible podríamos generar el siguiente modelo:

Ilustración 6. Modelo para trazabilidad

Cada vez que poblemos en ODS generamos un registro cuyo identificador y los campos de inserción y modificación coinciden con los valores generados ODS; y además deberemos indicar su origen: base de datos + tabla para identificar el origen de ese registro.

Por ejemplo, en ODS_HC_DIRECCIONES poblaremos desde el operacional CRM en concreto desde dos tablas: CLIENTES y PRODUCTOS, por lo que tendremos dos BD_ORIGEN.

CUARTA PARTE

DESPUÉS DE TODO LO VISTO, ¿DEJARÍAS NUESTRO ECOSISTEMA ASÍ O PLANTEARÍAS OTRO DISEÑO MEJORADO?

Visto la cantidad de errores, y aplicando Data Quality, se podría agregar un proceso entre la etapa de STAGE y ODS donde generemos un informe con todos esos datos para que la gente del operacional aplique las posibles correcciones dentro de sus sistemas.

El nuevo ecosistema quedaría de la siguiente manera:

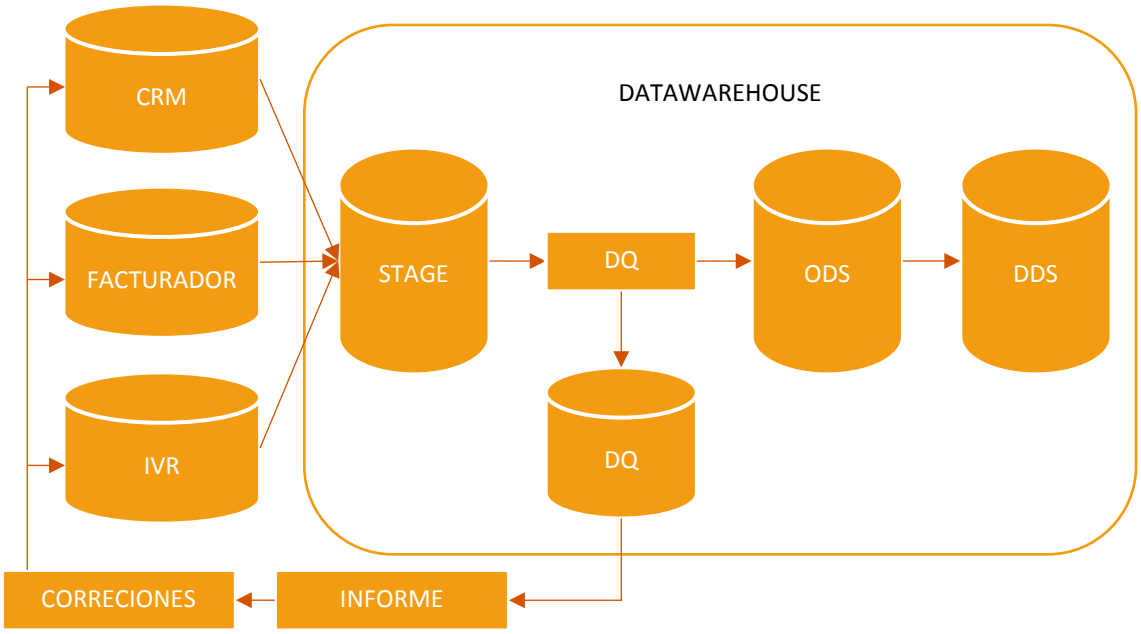

Ilustración 7. Ecosistema

QUINTA PARTE

ESCRIBE TUS PROPIAS REGLAS O MANDAMIENTOS DE UN DATAWAREHOUSE

1. El modelo de diseño del DataWarehouse utilizamos el modelo copo de nieve, pero lo más cercano al modelo de estrella posible. No hay que crear dimensiones a loco.
2. Las cargas a ODS mejor incrementales aún así vamos a necesitar un buen equipo.
3. No hay que borrar datos y tenemos que evitar las actualizar dentro de lo posible.
4. Debemos consolidar los datos, ya sean por erróneos o nulos, para ello tenemos el valor “NO APLICA” y “DESCONOCIDO”.

SEXTA PARTE

¿NIVEL DE SQL ANTES Y DESPUÉS?

El nivel de antes estaría entre “Dejad que las queries se acerquen a mí” y “Todo lo que quiso saber sobre SQL y no se atrevió a preguntar”.

El nivel actual estaría más cerca de “Todo lo que quiso saber sobre SQL y no se atrevió a preguntar”.

¿ALGÚN COMENTARIO EXTRA QUE QUIERAS HACER?

Para hacer la practica he aprovechado la cuenta de Azure y he utilizado una máquina más potente, aún así algunas sentencias que viendo que parecen claramente funcionar hasta que ejecutas y ves el resultado final no lo sabes, y esta ejecución en algunos casos es desesperante.

Por otro lado, me hubiera gustado mucho ver más a fondo las diversas herramientas ETL que existen en el mercado.