Migración de tipo de datos tipificados

Los datos tipificados fueron lo primero que decidimos migrar, ya que no tiene Foreign Keys y son muy simples.

La primera complicación que surgió fue migrar a la misma tabla datos tipificados que se podían encontrar en muchas columnas de la tabla maestra. La solución fue hacer el `INSERT` desde más de un `SELECT` unidos por `UNION`. Con esto el código ya funcionaba, pero teníamos muchas repeticiones de la forma

```
INSERT INTO table
    (nombre)
    SELECT DISTINCT column
    FROM gd_esquema.Maestra
    WHERE column IS NOT NULL;
```

Para solucionarlo, decidimos hacer un procedure que reciba dos parámetros, tabla de destino y columna de origen, y que implementara este formato. Mientras que esto no contempla los casos con un `UNION`, estos se puede emular llamando al nuevo procedure más de una vez con columnas de origen diferentes.`INSERT`

Migración de personas

La principal complicación de esto es que en la bases de datos original no existen las personas, sino que estas pueden ser Inquilinos, Propietarios, Agentes, o Compradores, cada cual tiene sus propias columnas. Si se hubieran migrado todos los datos a la entidad Persona sin tener en cuenta el resto de las Entidades, hubiera sido dificil recuperar la información de si la persona migrada era inquilino, propietario, comprador, o agente. Además, para migrar las tablas como Inquilino, se requiere el id de la Persona, por lo que sólo se la puede migrar después de migrar la persona.

Para encarar esto, por lo tanto, usamos tablas temporales y el operador `OUTPUT`. Habiendo un procedure diferente para inquilinos, propietarios, compradores, y agentes, creamos una tabla temporal cuando se empieza el procedure. Después, se insertan los datos en Personas, y usando

```
OUTPUT inserted.id_persona INTO #Temp (id_persona)
```

insertamos todos los ids nuevamente creados en la tabla temporal, que después insertamos en la tabla que corresponda.

Luego para migrar las Ventas que tienen un Comprador o los alquileres que tienen un Inquilino usamos dos JOINS. El primero era para encontrar la persona utilizando el DNI y MAIL como atrbiutos unicos y luego unirlo al Comprador con el id de sa persona

```
JOIN ANDY_Y_SUS_SEMINARAS.Persona p ON m.COMPRADOR_DNI = p.dni AND
m.COMPRADOR_MAIL = p.mail
JOIN ANDY_Y_SUS_SEMINARAS.Comprador c ON c.persona_id = p.id_persona
```

Migración de tablas normales

Para las tablas normales, el proceso es bastante simple. Se crea la tabla, se insertan los datos, y se crean las Foreign Keys.

Luego, para migrarlas, simplemente se seleccionan los datos de la tabla original y se insertan en la nueva. Y los datos que son Foreign Keys se reemplazan por los ids de las tablas nuevas mediante JOINs.

Migracion del Alquiler

Para esta entidad, en vez de seguir el proceso tradiconal del JOIN para buscar el ID de nuestras entidades, tuvimos que usar un LEFT JOIN con el detalle importe dado que habia muchos alquileres en la tabla maestra que tenian este atrbiuto como NULL pero que igualmente consideramos eran alquileres validos

Tampoco podiamos usar el CODIGO de la tabla maestra como PK dado que el mismo estaba repetido para alquileres distintos o con periodos distintos, lo que significa eran alquileres distintos, por lo que tuvimos que usar la PK autogenerada como Identity

```
 SELECT DISTINCT
      em.ALQUILER_CODIGO,
      em.ALQUILER_FECHA_INICIO,
      em.ALQUILER_FECHA_FIN,
      inq.id_inquilino,
      e.id_estado_alquiler,
      d.id_detalle_importe,
      em.ALQUILER_CANT_PERIODOS,
      em.ALQUILER_DEPOSITO,
      em.ALQUILER_COMISION,
      em.ALQUILER_GASTOS_AVERIGUA,
      a.nro_anuncio
   FROM gd_esquema.Maestra em
   JOIN ANDY_Y_SUS_SEMINARAS.Persona p ON em.INQUILINO_DNI = p.dni AND em.INQUILINO_MAIL
= p.mail
   JOIN ANDY_Y_SUS_SEMINARAS.Inquilino inq ON inq.persona_id =  p.id_persona
   JOIN ANDY_Y_SUS_SEMINARAS.TipoPeriodo t ON em.ANUNCIO_TIPO_PERIODO = t.nombre
   JOIN ANDY_Y_SUS_SEMINARAS.EstadoAlquiler e ON em.ALQUILER_ESTADO = e.nombre
   LEFT JOIN ANDY_Y_SUS_SEMINARAS.DetalleImporte d ON em.DETALLE_ALQ_NRO_PERIODO_INI =
d.nro_periodo_inicio AND em.DETALLE_ALQ_NRO_PERIODO_FIN = d.nro_periodo_fin AND
em.DETALLE_ALQ_PRECIO = d.precio
   JOIN ANDY_Y_SUS_SEMINARAS.Anuncio a ON em.ANUNCIO_CODIGO = a.nro_anuncio
```

Migracion de Pago Alquiler

Como nos sucedio que un mismo codigoAlquiler tenia varios alquileres para cada periodo al hacer JOIN del alquiler con el Pago Alquiler siempre econtraba mas de un alquiler por lo que tuvimos que añadir una validacion mas para que filtre unicamente al alquiler cuyo rango de periodos se encuentre dentro del numero de periodo de ese pago Alquiler

```
JOIN ANDY_Y_SUS_SEMINARAS.Alquiler a ON m.ALQUILER_CODIGO = a.codigo_alquiler
          JOIN ANDY_Y_SUS_SEMINARAS.DetalleImporte di ON a.detalle_importe_id = di.id_detalle_importe
   JOIN ANDY_Y_SUS_SEMINARAS.MedioPago mp ON m.PAGO_ALQUILER_MEDIO_PAGO = mp.nombre
          WHERE PAGO_ALQUILER_NRO_PERIODO BETWEEN di.nro_periodo_inicio AND
di.nro_periodo_fin
```

Migracion de venta

Para esta entidad el problema fue que una venta tiene asociada una entidad pago_venta. En la tabla original el pago venta no tiene un id unico, por lo que tuvimos que realizar varios joins para

que el pago venta resulte unico, ademas, nuestro pago_venta tiene 2 FKs por lo que teniamos que buscar el medio de pago y la moneda en las tablas tipificadas.

Alquiler - Modicacion del DER

Nos dimos cuenta que no era necesario tener una tabla externa Duracion que contenga el tipo de periodo y la cantidad de periodos dado que el tipo de periodo ya se encuentra en el anuncio del alquiler por lo que seria un dato desnormalizado

Ahora simplemenmte la duracion es un entero con la cantidad de periodos