Algoritmo para realizar *matching* automático en bases de datos

Universidad Nacional de Río Negro

Sede Atlántica

Licenciatura en Sistemas

Autor: Graziani, Alfredo

Director: Mg. Formia, Sonia

1. Introducción

La búsqueda y el matching de cadenas de texto son tareas fundamentales en el ámbito de las bases de datos y la gestión de la información. Con el crecimiento exponencial de los datos y la diversidad de fuentes de información, se ha vuelto cada vez más crucial desarrollar técnicas eficientes para realizar búsquedas precisas y rápidas en conjuntos de cadenas de texto.

En este Trabajo Final de Carrera, se aborda específicamente el problema de implementar un procedimiento de matching de cadenas de texto en el motor de bases de datos PostgreSQL [PostgreSQL, s.f.], un sistema de gestión de bases de datos relacionales [Bertone, R., & Thomas, P. (2011)] ampliamente utilizado.

El objetivo principal de esta investigación es mejorar la capacidad de realizar matching de cadenas de texto en PostgreSQL, superando las limitaciones de las funciones de búsqueda de cadenas incorporadas en el sistema. Estas funciones, si bien son útiles en muchos casos, pueden ser insuficientes cuando se trata de escenarios más complejos que requieren mayor flexibilidad y rendimiento. Por lo tanto, se propone desarrollar un procedimiento personalizado que aproveche las capacidades de PostgreSQL y brinde una solución más eficiente y precisa para el matching de cadenas de texto.

El entorno real en el que se utiliza el matching de cadenas de texto es amplio y diverso. Desde aplicaciones de búsqueda web hasta sistemas de recomendación en plataformas de comercio electrónico, este proceso juega un papel fundamental en la recuperación de información precisa y relevante. Además, en el ámbito de la gestión de datos, se aplica en la deduplicación de registros, la corrección de datos erróneos o mal escritos, y la identificación de patrones y relaciones en conjuntos de datos masivos.

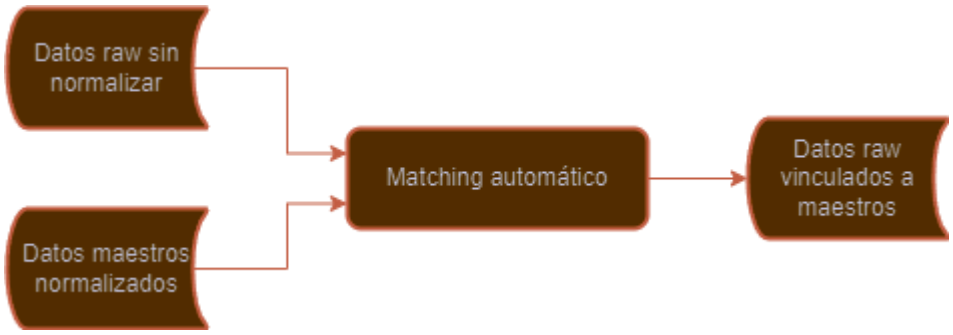
El proceso de matching de cadenas de texto implica comparar y encontrar coincidencias entre cadenas de texto en un conjunto de datos. Para lograr esto, se utilizan diversas técnicas y algoritmos, como la búsqueda de patrones, el uso de expresiones regulares, la distancia de edición y los trigramas [Friedl, 2006]. Estas técnicas permiten identificar cadenas similares o idénticas, incluso cuando existen variaciones, errores ortográficos o diferencias léxicas.

En este trabajo, se enfocará en la mejora de un procedimiento personalizado de matching de registros para PostgreSQL, basado en la implementación de un algoritmo de matching de cadenas de texto. Se explorarán enfoques como el uso de trigramas, el cálculo de distancias de edición y otros métodos que permitan mejorar la precisión y el rendimiento.

A continuación, se presentarán unos diagramas que explican de forma general el funcionamiento del algoritmo.

Como puede observarse en la [Figura 1], el procedimiento se alimenta de una base de datos con información normalizada, los cuales son llamados registros maestros, y de información no normalizada, que es informada por terceros, y se añade a la base de datos como información raw (cruda, sin procesar ni normalizar). Esta información no es editable, ya que se necesita mantener el registro como fue informado, por lo que, en lugar de editarlo, se vincula al maestro correspondiente.

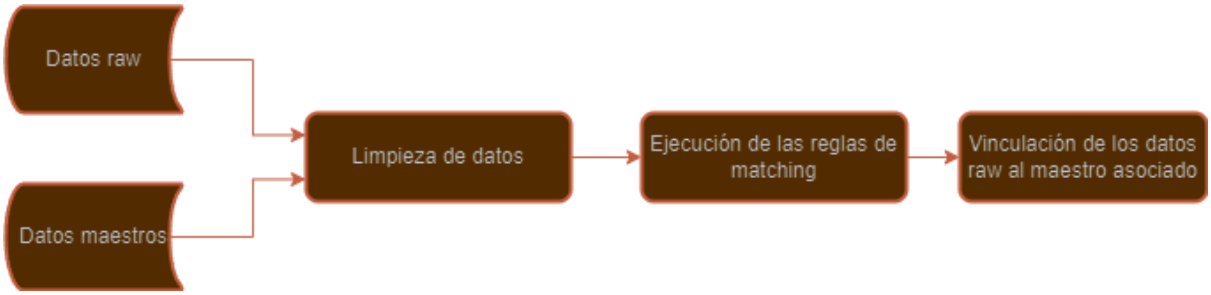
Figura 1. Diagrama general del proceso



Fuente: Elaboración propia

Introduciéndonos en el proceso del matching automático, el mismo consiste en iniciar con una limpieza temporal de los datos, esto significa, simplificar la información de manera que, al comparar los datos, haya más chance de una coincidencia. Por ejemplo, si una empresa informa su nombre con espacios de más o con símbolos extraños que no deberían estar, se puede dificultar la búsqueda de coincidencias con la información del maestro que está normalizada, ya que esos caracteres no van a estar. Luego ejecuta regla por regla tratando de encontrar una coincidencia, y en caso de que lo haga, se vincula al raw con el maestro. Vemos en la [Figura 2] cómo se ejecuta el automatching.

Figura 2. Ejecución del proceso de automatching

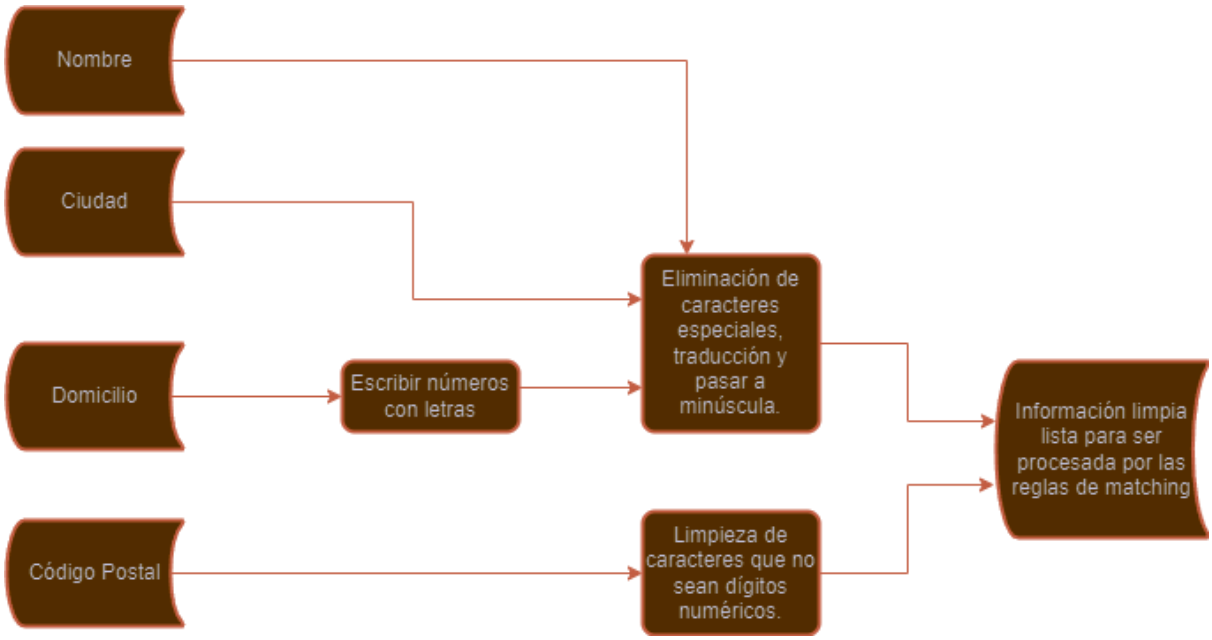


Fuente: Elaboración propia

Como se mencionó, el paso de la limpieza de datos se encarga de reducir la información de los campos a un valor que pueda ser lo más aproximado al valor del maestro. En caso del nombre de la empresa, se van a eliminar los caracteres especiales (espacios en blanco,

guiones, barras, asteriscos, etc.), se va a traducir algunos caracteres extranjeros (como los del alfabeto ruso) y se va a llevar todo a minúscula. Esto se realiza con cada campo, y en cada uno de ellos se hace una limpieza particular. Se explicará en detalle en la sección de descripción del proceso de automatching. A continuación, en la [Figura 3] se observa cómo se ejecuta la limpieza de datos.

Figura 3. Ejecución de la limpieza de datos

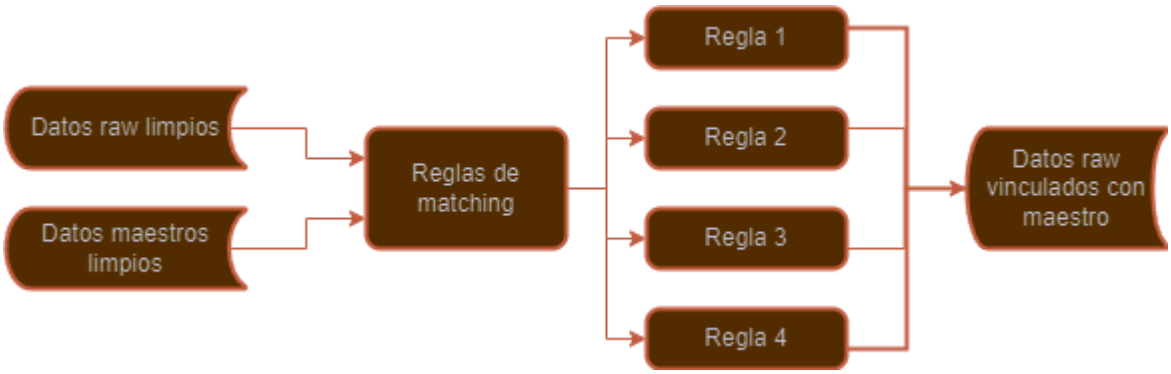


Fuente: Elaboración propia

El ejemplo muestra 4 campos para mostrar a nivel general cómo es el proceso. La estructura del archivo que se va a usar para este trabajo cuenta con muchos campos más que luego serán descritos con su limpieza específica.

Por último, se muestra en la [Figura 4] el paso final que usa la información previamente limpiada para ejecutar las reglas del automatching, para intentar vincular los registros raw con algún registro maestro.

Figura 4. Ejecución de las reglas de automatching



Fuente: Elaboración propia

En resumen, el presente trabajo tiene como objetivo abordar el problema del matching de cadenas de texto en PostgreSQL, proponiendo una mejora de un procedimiento personalizado para lograr más eficiencia y precisión de esta tarea. A través de técnicas innovadoras y adaptadas al entorno de PostgreSQL, se buscará superar las limitaciones de las funciones de búsqueda de cadenas de texto incorporadas en el sistema y proporcionar una solución más efectiva para el manejo de cadenas de texto en bases de datos.

2. Marco teórico

PostgreSQL es un sistema de gestión de bases de datos relacionales (RDBMS, por sus siglas en inglés) de código abierto ampliamente utilizado que ha ganado popularidad en los últimos años debido a sus potentes características y robustez. Una de las características clave de PostgreSQL es su soporte para funciones y procedimientos definidos por el usuario (UDFs, por sus siglas en inglés). Estas permiten a los usuarios ampliar la funcionalidad de la base de datos implementando lógica personalizada en un lenguaje de su elección. [Bertone, R., & Thomas, P. (2011)]

En el contexto de la búsqueda de cadenas de texto, PostgreSQL proporciona varias funciones de búsqueda de texto incorporadas, como LIKE, ILIKE [Gennick, J. (2010)] y la búsqueda de expresiones regulares usando los operadores ~ y ~* [Gennick, J. (2010)]. Sin embargo, estas funciones tienen limitaciones y es posible que no sean adecuadas para todos los casos de uso. Por ejemplo, las expresiones regulares pueden ser lentas y consumir muchos recursos, especialmente al trabajar con conjuntos de datos grandes.

Para superar estas limitaciones, varios investigadores [Sasaki, Y., Hirao, T. (2007)] han propuesto procedimientos personalizados para la búsqueda de texto en PostgreSQL. Uno de estos procedimientos es el algoritmo SimString, que es un enfoque rápido y escalable para la búsqueda aproximada de cadenas de texto. Este algoritmo se basa en la noción de n-gramas

y utiliza un índice invertido para encontrar coincidencias de manera eficiente [Kimura, M., & Sato, T. (2012)]. Varios investigadores [Okazaki, N., & Tsujii, J. (2010)] han implementado SimString en PostgreSQL como una función definida por el usuario (UDF) y han informado mejoras significativas en el rendimiento en comparación con las funciones de búsqueda de texto nativas.

Otro enfoque para la búsqueda de cadenas de texto en PostgreSQL es el uso de trigramas. Los trigramas son una secuencia de tres caracteres consecutivos en una cadena y se pueden utilizar para encontrar coincidencias de manera eficiente. PostgreSQL proporciona una extensión incorporada pg_trgm que implementa la búsqueda de similitud basada en trigramas. Los investigadores [Aumüller, M., Faust, A., & Schweikardt, N. (2005)] han propuesto varias modificaciones al algoritmo pg_trgm para mejorar su rendimiento y precisión.

Otros investigadores [Baeza-Yates, R., Navarro, G. (2001)] han explorado el uso de otras técnicas, como la distancia de Levenshtein y la distancia de Jaro-Winkler, para la búsqueda de cadenas de texto en PostgreSQL. Estas técnicas se basan en la noción de distancia de edición y se pueden utilizar para encontrar coincidencias incluso en presencia de errores ortográficos o tipográficos.

En conclusión, PostgreSQL proporciona un conjunto rico de funciones incorporadas para la búsqueda de cadenas de texto, pero estas funciones pueden no ser adecuadas para todos los casos de uso. Los investigadores han propuesto varios procedimientos personalizados para la búsqueda de cadenas en PostgreSQL que pueden mejorar significativamente el rendimiento y la precisión. Los enfoques basados en SimString y trigramas han tenido particular éxito en este sentido.

3. Matching automático

El problema que aborda la búsqueda de cadenas es fundamental en muchas áreas de la informática, especialmente en la recuperación de información y la minería de datos. La búsqueda de cadenas se utiliza en entornos de búsqueda de texto, como motores de búsqueda.

El matching de cadenas se refiere a la búsqueda de cadenas de texto que son similares o idénticas a una cadena dada. El matching de cadenas es un problema complejo debido a la gran cantidad de variaciones que pueden existir en ellas, como errores de ortografía, sinónimos, abreviaturas y variantes de idioma. Además, los conjuntos de datos pueden ser muy grandes, lo que requiere técnicas eficientes para la búsqueda. [Truong, D. V., Nguyen, T. T., & Nguyen, M. T. (2013)]

En el entorno real, el matching de cadenas se utiliza para encontrar correspondencias entre cadenas de texto en una variedad de situaciones, como la correspondencia de nombres en bases de datos de clientes, la identificación de documentos duplicados en un conjunto de documentos, la identificación de patrones de fraude en transacciones financieras y la correspondencia de direcciones en bases de datos de envíos.

En una base de datos donde se tiene información normalizada, al recibir información nueva, es común que ésta llegue incompleta, no está estandarizada, no respete el formato de cada campo, entre otras cosas.

Para realizar el matching de cadenas, se comparan dos cadenas y se determina si son iguales o similares. La comparación puede ser basada en caracteres, en términos de la frecuencia de los caracteres en la cadena, o en la distancia entre las cadenas, en términos de la cantidad de operaciones de edición necesarias para transformar una cadena en la otra. La distancia de edición puede ser medida por métodos como la distancia de Levenshtein y la distancia de Jaro-Winkler. [Aumüller, M., Faust, A., & Schweikardt, N. (2005)]

El proceso automatiza el matcheo de esa información con el registro correcto de la base de datos normalizada, para así actualizar la información y enlazarla con el dato normalizado correspondiente.

El matching de cadenas se realiza entre datos que se encuentran en una base de datos o en un conjunto de datos de texto. Las técnicas para el matching de cadenas se aplican a cadenas de texto individuales o a conjuntos de cadenas de texto, dependiendo del caso de uso. En la mayoría de los casos, se utilizan técnicas de indexación de cadenas y búsqueda para mejorar la eficiencia del matching de cadenas en grandes conjuntos de datos.

Descripción del proceso

3.1 Introducción

El proceso definido está basado en la premisa que se tiene información normalizada, o datos maestros, e información no normalizada, definida como un archivo de novedades, o información “raw” (llamada así por su traducción del inglés, crudo, sin procesar).

La información normalizada debió atravesar un proceso de revisión en donde se chequea que los datos son correctos y están bien escritos. Una vez verificado, se acepta el dato maestro y se lo califica como tal.

En el escenario definido en este trabajo, los datos raw son información que se recibe de clientes, los cuales informan las ventas realizadas por revendedores de sus productos. Por ejemplo, una empresa de computación que fabrica componentes de PC vende sus productos

a tiendas de varias ciudades de Latinoamérica, y esas tiendas venden sus productos a usuarios finales. Esas ventas son registradas e informadas como novedades, y al ser muchos revendedores los que informan, es común que cada tienda lo informe con formatos variados o con información incompleta. El objetivo de tener los datos maestros normalizados es poder verificar a través del proceso de automatching esta información raw y poder vincular los datos de esas ventas a la empresa que corresponde.

3.2 Estructura de la información

Como se explicó anteriormente, el proceso está encargado de matchear información entre un registro normalizado contra uno que no lo está. Para lograr esto, se requiere usar distintos campos para verificar que los datos coinciden y relacionar los registros. A continuación, se describe la estructura de los datos en la [Figura 5], que comparten tanto el maestro como los datos raw.

Figura 5. Estructura de la tabla *reseller_company_raw*

reseller_company_raw		
PK	reseller_company_raw_id	INT
	reseller_company_id	INT
	name	VARCHAR
	city	VARCHAR
	country_id	INT
	address	VARCHAR
	postcode	VARCHAR
	legal_entity	VARCHAR
	vat_num	VARCHAR
	company_registration_number	VARCHAR
	created	DATE

Fuente: Elaboración propia

- **ID**: El *ID* es un identificador único para cada registro. En el caso del maestro se llama *reseller_company_id*, y en el raw *reseller_company_raw_id*.
- **Nombre (name)**: El nombre de la empresa o entidad. Es un campo que permite caracteres alfanuméricos de cualquier tipo.

- **Ciudad (city)**: El nombre de la ciudad a la que pertenece la empresa escrita en caracteres alfabéticos.
- **País (country_id)**: El ID del país al que pertenece la empresa. Es un valor numérico.
- **Dirección (address)**: La dirección postal de la empresa. Puede contener valores alfanuméricos y caracteres especiales.
- **Código postal (postcode)**: El código postal de la ciudad donde pertenece la empresa. Según el país, puede contener valores alfanuméricos o sólo numéricos.
- **Entidad legal (legal_entity)**: El nombre de la entidad legal de la empresa. Puede o no ser el mismo valor que el nombre.
- **Número de identificación fiscal (vat_num)**: El número de identificación fiscal (NIF, o VAT por sus siglas en inglés, Value-added Tax Identification Number) es un identificador usado en muchos países utilizado para identificación del impuesto sobre el valor añadido. Puede contener valores alfanuméricos.
- **Número de registro de empresa (company_registration_number)**: El número de registro de empresa, o CRN por sus siglas en inglés, Company Registration Number, es un código único que utilizan las empresas para ser identificadas rápidamente. Puede contener valores alfanuméricos.
- **Fecha de creación (created)**: Fecha de creación del registro de la empresa.

En la estructura de la tabla de novedades (reseller_company_raw) hay un campo extra que es el ID del maestro al cual está asignado. Es el campo que identifica a cada Raw con su Maestro. Estos registros no entran al proceso de automatching puesto que ya están vinculados con un maestro.

Muchos de estos campos son opcionales para un archivo de novedades. Es posible que un cliente cuando informe ventas de una empresa reporte el nombre, pero no la entidad legal, o especifique el NIF, pero no el CRN. Por esto se utilizan distintas reglas que usan los diferentes campos para efectuar el matching.

3.3 Shell Script

El proceso de auto matching es un simple shell script que ejecuta una serie de scripts SQL y escribe cualquier salida en un archivo de registro que es usado para mostrar la información final al usuario.

El proceso ejecuta los scripts directamente en la base de datos.

Script SQL

A continuación, se describe brevemente las acciones realizadas en los scripts SQL que son llamados por el shell script.

Importación de datos adicionales

El algoritmo permite que se usen archivos con modificadores opcionales para mejorar el matching. Los mismos pueden encontrarse en formato CSV, archivo separado por comas.

El proceso lee el archivo y carga la información en tablas temporales de la base de datos.

Para el caso de ejemplo, se usan archivos para cargar:

- Extensiones legales
- Códigos postales
- Números de identificación fiscal

Ejecutar el automatching

En este paso es donde ocurre el automatching, los detalles del script se describen en las siguientes secciones. Como descripción general, el script ejecuta una serie de reglas de matcheo en un orden específico, crea tablas en la base de datos que contienen información sobre cada regla, y aplica los matcheos.

Exportar resultados

El script opcionalmente copia los resultados en un archivo de texto que exporta a una ubicación predefinida.

Ejemplo:

- **Servidor**: /root/projects/matchmaker/data/

3.4 Automatching

El script de auto matching se divide en 7 reglas, cada una con una lógica distinta para mapear un registro nuevo con la base de registros maestros.

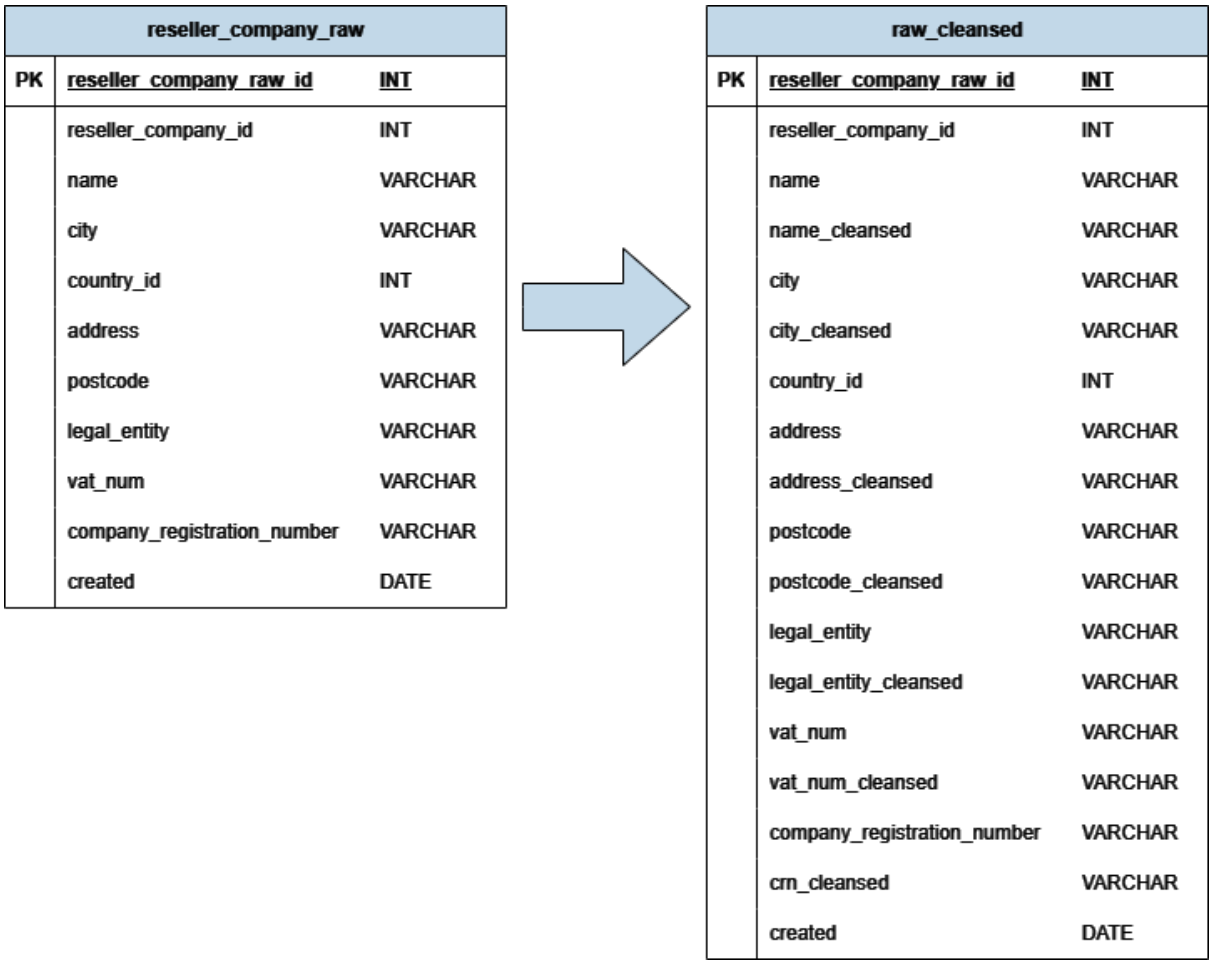
Los registros maestros serán los que contienen información limpia y aprobada, mientras que los nuevos son los que se informan cada cierto tiempo y pueden contener datos no normalizados. En el anexo, se puede observar el código del proceso escrito en lenguaje SQL.

3.4.1 Extracción de datos

Antes de que inicie la ejecución de las reglas, hay 2 extracciones de información de datos nuevos y maestros. Estas extracciones alteran temporalmente algunos strings que serán usados en el proceso de matching.

A continuación, en la [Figura 6] se muestra la estructura de la tabla temporal en la que se guarda la información extraída para ser utilizada en las reglas.

Figura 6. Estructura de la tabla temporal con datos limpios



Fuente: Elaboración propia

Datos Maestros

Filtro

Los datos maestros pueden ser filtrados según ciertas condiciones, por ejemplo:

```
WHERE live status
AND
(
        master source (CTX or SE or none)
    OR
        master source has a reseller_edit user action logged after
        2010-02-01
)
```

En este ejemplo observamos condiciones respecto al estado de maestro, si está catalogado como tal, o aún está en proceso de revisión. Luego se asegura que la fuente

origen sea una empresa como “CTX” o “SE”, o que el registro haya tenido una actualización manual después de cierta fecha, para asegurarse que se estén revisando los más recientes.

Modificadores de texto

La extracción también crea copias de 4 campos de texto con algunas modificaciones basadas en patrones RegEx. Estas modificaciones son enumeradas a continuación (los ítems están en orden de ejecución):

Nombre

- Reemplaza la palabra and (case sensitive) con un ampersand (&). La palabra debe estar rodeada de espacios los cuales no son reemplazados.
- Remueve las comillas simples (‘)
- Transforma el string a minúscula
- Borra los espacios del inicio y del fin del string
- Remueve cualquier extensión legal predefinida (controlada por el archivo CSV previamente cargado) del final del string, en caso de que la extensión legal (case insensitive) esté precedida por uno de los siguientes caracteres:
    - Espacio ( )
    - Punto (.)
    - Barra (/)
    - Guion (-)
- Remueve cualquiera de los siguientes caracteres:
    - Espacio ( )
    - Punto (-)
    - Barra (/)
    - Coma (,)

Código postal

- Si el país es Francia o Alemania
    - Remueve cualquier carácter no numérico
    - Remueve todo después de los primeros 5 caracteres
- Para cualquier otro país
    - Convierte el string a minúscula
    - Borra los espacios del inicio y del fin del string
    - Remueve cualquier prefijo del código postal (controlado por el CSV previamente cargado) seguido de:
        - Guion (-)

        - Espacio ( )
    - Remueve los caracteres **EC** del final del string

Número de identificación fiscal

- Convierte el string a minúscula
- Borra los espacios del inicio y del fin del string
- Remueve cualquier prefijo del NIF (controlado por el CSV previamente cargado)

Entidad Legal

- Convierte el string a minúscula
- Borra los espacios del inicio y del fin del string
- Remueve cualquier extensión legal predefinida (controlada por el archivo CSV previamente cargado) del final del string, en caso de que la extensión legal (case insensitive) esté precedida por uno de los siguientes caracteres:
    - Espacio ( )
    - Punto (.)
    - Barra (/)
    - Guion (-)

Datos raw

Filtro

El filtro para los datos nuevos es un simple chequeo que identifica cualquier dato nuevo sin un maestro asignado.

Modificadores de texto

La extracción de datos nuevos también crea una copia modificada de los siguientes campos:

Nombre

- Reemplaza la palabra and (case sensitive) con un ampersand (&). La palabra debe estar rodeada de espacios los cuales no son reemplazados.
- Remueve las comillas simples (‘)
- Transforma el string a minúscula
- Borra los espacios del inicio y del fin del string

- Remueve cualquier extensión legal predefinida (controlada por el archivo CSV previamente cargado) del final del string, en caso de que la extensión legal (case insensitive) esté precedida por uno de los siguientes caracteres:
    - Espacio ( )
    - Punto (.)
    - Barra (/)
    - Guion (-)
- Remueve cualquier asterisco (*)
- Remueve cualquiera de los siguientes caracteres:
    - Espacio ( )
    - Punto (-)
    - Barra (/)
    - Coma (,)

Hay un paso final que sólo aplica para Polonia y Rusia, que es remover o traducir los caracteres cirílicos. Los pasos se detallan a continuación:

- Rusia
    - Transforma el string a minúscula
    - Traduce lo siguiente:
        - a=a, б=b, в=v, г=g, д=d, е=e, ё=e, з=z, и=i, й=j, к=k, л=l, м=m, н=n, о=o, п=p, р=r, с=s, т=t, у=u, ф=f, х=h, ы=y, э=e
    - Remueve los siguientes caracteres:
        - Ъ
        - Ь
    - Traduce lo siguiente:
        - ж=zh, ц=ts, ч=ch, ш=sh, щ=sch, ю=yu, я=ya
    - Remueve cualquier carácter que no sea alguno de los siguientes:
        - Caracteres alfabéticos
        - Caracteres numéricos
        - Espacio ( )
        - Tabulaciones
        - _`'",().""<>#«»&/:+!“”’№@|-
- Polonia
    - Transforma el string a minúscula
    - Traduce lo siguiente:

- a=a, ą=a, b=b, c=c, ć=c, d=d, e=e, ę=e, f=f, g=g, h=h, i=i, j=j, k=k, l=l, ł=l, m=m, n=n, ń=n, o=o, ó=o, p=p, r=r, s=s, ś=s, t=t, u=u, w=w, y=y, z=z, ź=z, ż=z

Código postal

Es idéntico a lo que se realiza en la modificación de los datos maestros

Número de identificación fiscal

Es idéntico a lo que se realiza en la modificación de los datos maestros

Una vez realizado este paso, se inicia la ejecución de las reglas. En el anexo, puede observarse que se muestra un gráfico de ejemplo de la estructura de las tablas utilizadas en el proceso.

3.4.2 Regla 1 – NIF y Código Postal

Esta es la primera regla que se aplica, y busca coincidencias usando el número NIF y el código postal.

Filtro

Antes de realizar cualquier mapping, el dato maestro y el nuevo son filtrados por la longitud de su código postal y su número NIF (las versiones ya modificadas). Las reglas de este filtro son las siguientes:

- La longitud del número NIF debe ser mayor a 5 caracteres después de remover los siguientes:
  - Espacio ( )
  - Guion (-)
  - Barra (/)
  - Coma (,)
  - Asterisco (*)
- La longitud del Código postal debe ser mayor a 3 caracteres después de remover los siguientes:
  - Espacio ( )
  - Guion (-)
  - Barra (/)
  - Coma (,)

    - Asterisco (*)
- La eliminación adicional de cualquier carácter cero (0) del número NIF debe tener una longitud superior a 1 carácter.
- La eliminación adicional de cualquier carácter cero (0) del código postal debe tener una longitud superior a 0 caracteres.

Comparación

Cuando se comparan los datos nuevos y maestros, la siguiente lógica es aplicada:

- El país de ambos registros debe coincidir
- El código postal modificado usado en el filtro debe coincidir
- Los números NIF deben coincidir, sin embargo, hay una lógica adicional que determina qué número NIF es usado:
    - Si el país es Reino Unido, la comparación se realiza con el número NIF original (no el modificado) con los siguientes caracteres removidos:
        - Espacio ( )
        - Guion (-)
        - Barra (/)
        - Asterisco (*)
        - Alfabeto (A-Z / case insensitive)
    - Para cualquier otro país
        - Se utiliza el número NIF modificado

Múltiples coincidencias

Finalmente, los datos nuevos que hagan match con múltiples maestros son descartados de esta regla. Sólo los datos nuevos con un solo maestro son considerados un match exitoso.

3.4.3 Regla 2 – Número de registro y Código Postal

Esta es la segunda regla aplicada, y busca coincidencias usando el número de registro y el código postal.

Filtro

Antes de que se realice el matching, los datos maestros y nuevos son filtrados por la longitud de su código postal y su número de registro (las versiones ya modificadas). Las reglas de este filtro son las siguientes:

- La longitud del número de registro debe ser mayor a 4 caracteres después de remover los siguientes:

    - Espacio ( )
    - Guion (-)
    - Barra (/)
    - Coma (,)
    - Asterisco (*)
- La longitud del Código postal debe ser mayor a 3 caracteres después de remover los siguientes:
    - Espacio ( )
    - Guion (-)
    - Barra (/)
    - Coma (,)
    - Asterisco (*)
- La eliminación adicional de cualquier carácter cero (0) del número de registro debe tener una longitud superior a 1 carácter.
- La eliminación adicional de cualquier carácter cero (0) del código postal debe tener una longitud superior a 0 caracteres.

Comparación

Cuando se comparan los datos nuevos y maestros, la siguiente lógica es aplicada:

- El país de ambos registros debe coincidir
- El código postal modificado usado en el filtro debe coincidir
- Los números de registro modificados deben coincidir

Múltiples coincidencias

Finalmente, los datos nuevos que hagan match con múltiples maestros son descartados de esta regla. Sólo los datos nuevos con un solo maestro son considerados un match exitoso.

3.4.4 Regla 3 – Número de registro y Nombre

Esta es la tercera regla que se aplica, y busca coincidencias usando el número de registro y los nombres.

Filtro

Antes de que se realice el matching, los datos maestros y nuevos son filtrados por la longitud de su código postal y su número de registro (las versiones ya modificadas). Las reglas de este filtro son las siguientes:

- La longitud del número de registro debe ser mayor a 4 caracteres después de remover los siguientes:
    - Espacio ( )
    - Guion (-)
    - Barra (/)
    - Coma (,)
    - Asterisco (*)
- La eliminación adicional de cualquier carácter cero (0) del número de registro debe tener una longitud superior a 1 carácter

Comparación

Cuando se comparan los datos nuevos y maestros, la siguiente lógica es aplicada:

- El país de ambos registros debe coincidir
- Los números de registro modificados deben coincidir
- Los nombres deben coincidir después de las siguientes modificaciones adicionales:
    - Nombre modificado del dato nuevo
        - Remover los siguientes caracteres:
            - Espacio ( )
            - Guion (-)
            - Barra (/)
            - Coma (,)
            - Asterisco (*)
    - Nombre modificado del dato maestro
        - Remover todos los asteriscos (*)
        - Convertir el string a minúscula

Múltiples coincidencias

Finalmente, los datos nuevos que hagan match con múltiples maestros son descartados de esta regla. Sólo los datos nuevos con un solo maestro son considerados un match exitoso.

3.4.5 Regla 4 – Código Postal y Nombre

Esta es la cuarta regla aplicada, y busca coincidencias usando el código postal y el nombre.

Filtro

Antes de que se realice el matching, los datos maestros y nuevos son filtrados por la longitud de su código postal (la versión ya modificada). Las reglas de este filtro son las siguientes:

- La longitud del Código postal debe ser mayor a 3 caracteres después de remover los siguientes:
    - Espacio ( )
    - Guion (-)
    - Barra (/)
    - Coma (,)
    - Asterisco (*)
- La eliminación adicional de cualquier carácter cero (0) del código postal debe devolver un string no nulo.

Comparación

Cuando se comparan los datos nuevos y maestros, la siguiente lógica es aplicada:

- El país de ambos registros debe coincidir
- El código postal modificado usado en el filtro debe coincidir
- Los nombres deben coincidir después de las siguientes modificaciones adicionales:
    - Nombre modificado del dato nuevo
        - Remover los siguientes caracteres:
            - Espacio ( )
            - Guion (-)
            - Barra (/)
            - Coma (,)
            - Asterisco (*)
    - Nombre modificado del dato maestro
        - Remover todos los asteriscos (*)
        - Convertir el string a minúscula

Múltiples coincidencias

Finalmente, los datos nuevos que hagan match con múltiples maestros son descartados de esta regla. Sólo los datos nuevos con un solo maestro son considerados un match exitoso.

3.4.6 Regla 5 – Código Postal y Nombre Raw / Entidad Legal Maestra

Esta es la quinta regla aplicada, y busca coincidencias usando el código postal y el nombre de los datos sin procesar y los compara con entidades legales de los datos maestros.

Filtro

Antes de que se realice el matching, los datos maestros y nuevos son filtrados por la longitud de su código postal (la versión ya modificada). Las reglas de este filtro son las siguientes:

- La longitud del Código postal debe ser mayor a 3 caracteres después de remover los siguientes:
    - Espacio ( )
    - Guion (-)
    - Barra (/)
    - Coma (,)
    - Asterisco (*)
- La eliminación adicional de cualquier carácter cero (0) del código postal debe devolver un string no nulo.

Comparación

Cuando se comparan los datos nuevos y maestros, la siguiente lógica es aplicada:

- El país de ambos registros debe coincidir
- El código postal modificado del maestro es un match para el código postal del dato raw después de las siguientes modificaciones:
    - Remover los siguientes caracteres:
        - Asterisco (*)
        - Coma (,)
- El nombre raw es un match para la entidad legal maestra después de aplicar las siguientes modificaciones:
    - Modificación del nombre raw:
        - Remover los siguientes caracteres:
            - Asterisco (*)
        - Convierte el string en minúscula
    - Entidad legal maestra modificada:
        - Convierte el string en minúscula

Múltiples coincidencias

Finalmente, los datos nuevos que hagan match con múltiples maestros son descartados de esta regla. Sólo los datos nuevos con un solo maestro son considerados un match exitoso.

3.4.7 Regla 6 - Raw Matching

Esta es la sexta regla aplicada, y matchea el nombre y el código postal, pero con una diferencia clave que sólo va a matchear con maestros donde sus propios datos raw matcheen con los datos raw no matcheados.

Extracción

El primer paso de esta regla es extraer la información de los datos raw que ya fueron matcheados con un maestro. Esta data es modificada de la siguiente forma:

Nombre

- Convierte el string a minúscula
- Elimina los espacios del inicio y del final
- Remueve cualquier extensión legal predefinida (controlada por el archivo CSV previamente cargado) del final del string, en caso de que la extensión legal (case insensitive) esté precedida por uno de los siguientes caracteres:
    - Espacio ( )
    - Punto (.)
    - Barra (/)
    - Guion (-)
- Remueve los siguientes caracteres:
    - Asterisco (*)
    - Espacio ( )
    - Guion (-)
    - Barra (/)
    - Coma (,)

Código postal

- Si el país es Francia o Alemania
    - Remueve cualquier carácter no numérico
    - Remueve todo después de los primeros 5 caracteres
- Para cualquier otro país
    - Convierte el string a minúscula

- Elimina los espacios del inicio y del final
- Remueve cualquier prefijo del código postal (controlado por el CSV previamente cargado) seguido de:
  - Guion (-)
  - Espacio ( )
- Remueve los caracteres **EC** del final

Una vez que los datos modificados hayan sido extraídos, son filtrados por maestros con al menos 2 registros raw con datos modificados que matcheen.

Filtro

Antes de que se realice el matching, los datos maestros y nuevos son filtrados por la longitud de su código postal (la versión ya modificada). Las reglas de este filtro son las siguientes:

- La longitud del Código postal debe ser mayor a 3 caracteres después de remover los siguientes:
  - Espacio ( )
  - Guion (-)
  - Barra (/)
  - Coma (,)
  - Asterisco (*)
- La eliminación adicional de cualquier carácter cero (0) del código postal debe devolver un string no nulo

Comparación

Cuando se comparan los registros raw matcheados y no matcheados, la siguiente lógica se aplica:

- Los nombres modificados matchean
- Los códigos postales modificados matchean

Múltiples coincidencias

Finalmente, los datos nuevos que hagan match con múltiples maestros son descartados de esta regla. Sólo los datos nuevos con un solo maestro son considerados un match exitoso.

3.4.8 Regla 7 – NIF y Nombre

Esta es la séptima regla aplicada, y busca coincidencias usando el número NIF y el nombre

Filtro

Antes de que se realice el matching, los datos maestros y raw son filtrados por la longitud de su nombre y número NIF (las versiones ya modificadas). Las reglas de este filtro son las siguientes:

- La longitud del número NIF es mayor a 5 caracteres después de remover los siguientes:
    - Espacio ( )
    - Guion (-)
    - Barra (/)
    - Coma (,)
    - Asterisco (*)
- La eliminación adicional de cualquier carácter cero (0) del número NIF debe tener una longitud superior a 1 carácter
- El nombre no debe ser un string vacío después de remover los siguientes caracteres:
    - Espacio ( )
    - Guion (-)
    - Barra (/)
    - Coma (,)
    - Asterisco (*)

Exclusión

Esta regla tiene un paso adicional donde excluye los números NIF que existen en más de un maestro del mismo país. El número NIF y sus alteraciones usadas dependen del país, las mismas son aplicadas como se describe a continuación:

- Si el país es Reino Unido:
    - Remueve los siguientes caracteres del número NIF original:
        - Espacio ( )
        - Guion (-)
        - Barra (/)
        - Asterisco (*)
        - Alfabeto Romano (A – Z)
- Para cualquier otro país:

- Remueve los siguientes caracteres del ya modificado número NIF:
  - Espacio ( )
  - Guion (-)
  - Barra (/)
  - Coma (,)
  - Asterisco (*)

Comparación

Cuando se comparan los datos maestros y raw, la siguiente lógica se aplica:

- El país de ambos debe coincidir
- Las alteraciones al NIF usadas en la sección de exclusión deben coincidir
- Los nombres modificados deben coincidir después de las siguientes alteraciones:
  - Remueve los siguientes caracteres:
    - Espacio ( )
    - Guion (-)
    - Barra (/)
    - Coma (,)
    - Asterisco (*)

Múltiples coincidencias

Finalmente, los datos nuevos que hagan match con múltiples maestros son descartados de esta regla. Sólo los datos nuevos con un solo maestro son considerados un match exitoso.

3.4.9 Regla Double Metaphone – Nombre, Ciudad, Dirección

La función double metaphone acepta un valor de texto y devuelve una versión simplificada de cómo suena ese valor de texto. La idea de esta función es remover caracteres no alfabéticos y diferencias en grupos de caracteres, y tratar de normalizar el valor basado en cómo serían pronunciados literalmente.

Debajo hay algunos ejemplos de la función double metaphone:

Input	Output
Gavin	KFN
Kevin	KFN
Kvin	KFN
Jones	JNS

Johnson	JNSN
Gavin Jones	KFNJ
Kevin Johnson	KFNJ

Como se puede ver, la función double metaphone matchea nombres con sonidos similares lo cual puede ser útil cuando se trabaja con valores crudos (raw) que pueden tener errores ortográficos o formas diferentes de escribir lo mismo o usando diferentes juegos de caracteres.

Los ejemplos de arriba muestran un notable problema con la función. Podemos ver que tanto el resultado de Gavin Jones y Kevin Johnson es KFNJ.

Sin embargo, si se aplica la función a cada palabra y se concatena el resultado, se obtiene KFN JNS y KFN JNSN lo que mantiene la correcta diferencia entre los nombres.

Filtro

Los siguientes filtros se aplican a esta regla:

- Nombre Raw, ciudad y dirección son:
    - No nulos
    - No vacíos

Es importante notar que esta regla no filtra ni busca coincidencias por país, esto incrementa la posibilidad de matches erróneos dado a la forma en cómo trabaja la función double metaphone. Sin embargo, podría haber ocasiones en que ignorar el país pueda ser beneficioso, como comparar datos de España con Andorra.

Comparación

Cuando se comparan los datos nuevos y maestros, la siguiente lógica es aplicada:

- El resultado de la función double metaphone de los siguientes campos de los datos nuevos y maestros deben coincidir:
    - Nombre
    - Ciudad
    - Dirección

Múltiples coincidencias

Finalmente, los datos nuevos que hagan match con múltiples maestros son descartados de esta regla. Sólo los datos nuevos con un solo maestro son considerados un match exitoso.

4. Problemas detectados

- La tabla usada en el proceso SQL para detectar múltiples coincidencias puede llegar a tener filas duplicadas cuando no las necesita. Si hay 8 filas apuntando al mismo registro raw, pero esas 8 están hechas por 3 filas distintas, entonces sólo se necesitan esas 3 filas. Esto es posible optimizarlo con una consulta que haga una búsqueda con agrupamiento de datos.
- La lógica de extracción de datos reemplaza **and** con un *ampersand,* pero es case sensitive, y lo realiza antes de llamar a la función que transforma el string en minúscula, por lo que cualquier **and** que tenga mayúsculas no va a ser incluido en este reemplazo.
- La modificación del Código postal en la extracción de los datos maestros remueve los caracteres **EC** del final. Esto parece ser específico de algunos países por lo que debería ser aplicado como tal.
- La regla 2 (NIF y Código postal) puede ser muy amplia en algunos casos donde la compañía tiene múltiples propiedades en la misma área postal, y al ser de la misma compañía, comparten el mismo número NIF. Esto es manejado cuando se detectan datos raw con múltiples maestros como respuesta, los cuales son removidos para ser ejecutados en las siguientes reglas. Incluir la dirección en estas instancias podría ser una mejora para la precisión del auto matching.
- La función Double Metaphone devuelve un máximo fijo de 4 caracteres lo cual no es suficiente para cubrir valores de texto muy largos como algunos nombres de compañías o direcciones. Una forma más precisa sería dividir los valores en palabras individuales y ejecutar la función en ellos, luego concatenando el resultado.
- Hay muchos puntos en común entre las reglas y mucho preprocesamiento y alteraciones de texto en los registros raw que son realizados múltiples veces lo cual es innecesario. El proceso podría ser optimizado en este punto.

5. Mejora del proceso con la función Double Metaphone

Un sistema de puntuación basado en palabras

5.1 Introducción

En la siguiente sección, se explicará el funcionamiento del algoritmo con ejemplos para mostrar los resultados.

5.2 Algoritmo

Actualmente, la regla Metaphone ejecuta los siguientes pasos:

1. Recibe un string como entrada
2. Convierte cualquier número en su respectiva representación con palabras
3. Divide el string en palabras separadas
4. Aplica la función *dmetaphone()* a cada palabra obteniendo el resultado
5. Concatena los resultados de cada palabra en un solo string
6. Realiza los pasos 1 - 4 para ambos strings a comparar
7. Saca un puntaje aplicando una función que da la semejanza entre los strings creados en el paso 4.

Ejemplo:

Paso 1: Se recibe el string como entrada

Jorge Newbery 885

Paso 2: Convierte cualquier número en su respectiva representación con palabras

Jorge Newbery Ochocientos Ochenta y Cinco

Paso 3: Divide el string en palabras separadas

Jorge	Newbery	Ochocientos	Ochenta	y	Cinco

Paso 4: Aplica la función dmetaphone() a cada palabra obteniendo el resultado

JRJ	NBR	AXSN	OXNT		SNK

Paso 5: Concatena los resultados de cada palabra en un solo string

JRJ NBR AXSN OXNT SNK

Paso 6: Realiza los pasos 1-4 para ambos strings a comparar

Jorge Newbery
JRJ NBR

Paso 7: Saca un puntaje aplicando una función que da la semejanza entre los strings creados en el paso 4.

JRJ NBR AXSN OXNT SNK	0.36
JRJ NBR	

5.3 Función similarity()

La función similarity toma como input 2 strings y calcula la similitud entre ambos. Devuelve un valor entre 0 y 1.

5.4 Problemas con el algoritmo

Hay algunos problemas destacables del algoritmo que pueden provocar algunos resultados no deseados.

5.4.1 Máxima longitud del output de la función double metaphone

La longitud máxima del string de salida de la función double metaphone es de 4 caracteres. La idea de esta función es proveer un string que explique de la mejor manera cómo suena la palabra fonéticamente de forma que palabras muy similares puedan ser matcheadas.

El problema con sólo 4 caracteres aparece cuando las palabras son muy largas, y la representación fonética no es representativa. Existe una función metaphone() que permite especificar la longitud máxima del string.

Pongamos el siguiente ejemplo:

Input	Dmetaphone (4)	Metaphone (12)
Industrial	ANTS	INTSTRL

Podemos ver que, con más caracteres disponibles, tenemos una representación mucho más cercana al string original, y ya que la función similarity calcula el puntaje basado en el número de caracteres, 1 cambio en un string más largo tiene menos impacto que 1 cambio en un string corto.

5.4.2 Números convertidos en palabras

En el paso a paso del algoritmo, el sexto y séptimo paso muestran que comparando 2 strings, uno sin número de calle y otro con, éste último produce un string considerablemente más largo como salida de la función double metaphone.

Esto se debe a que el número 885 es convertido en “Ochocientos Ochenta y Cinco”, dejando 3 (la y no es tomada por double metaphone) palabras más en este string. Pasar estos 2 strings con una considerable diferencia de longitud resulta en un puntaje bajo de similitud.

5.4.3 Bajo puntaje en substrings contenidos perfectamente

Muy seguido se ven casos donde la información tiene diferente longitud, y esto puede causar problemas con la función de similarity, ya que tendrán un puntaje menor.

Tomemos el siguiente ejemplo:

Raw	Juan de la Piedra 886	0.57
Maestro	Juan de la Piedra 886, Carmen de Patagones	

Podemos ver que el valor Raw está completamente contenido en el valor maestro. Pero como en el maestro está el nombre de la localidad, todos esos caracteres extra son considerados no similares y bajan el puntaje de la comparación.

Se debería tener un valor mucho más alto en estos casos, una vez identificados se les puede asignar un valor distinto o encender un flag, para así evitar la excesiva severidad de la función similarity en estas situaciones.

5.4.4 Números y letras en la misma palabra

Comúnmente se pueden ver casos en los que en una dirección puede haber, por ejemplo, el número de puerta de un edificio, mostrado como “5A”.

El problema con estos casos es que la función que convierte los números en strings, sólo lo hace para valores completamente numéricos, para cualquier carácter alfabético o especial (guiones, barras, etc) sencillamente devuelve el string original.

Este string es después pasado como input a la función double metaphone, quien termina devolviendo un string vacío, dificultando así el matcheo, ya que se pierde información.

6. Demostración del procedimiento

En la siguiente sección se mostrará una prueba del procedimiento funcionando en una base de datos PostgreSQL con información piloto.

Contamos con una base de datos relacional donde existen 2 tablas principales: reseller.reseller_company y reseller.reseller_company_raw. Respectivamente representan los datos maestros y los datos raw.

En el anexo podemos observar un ejemplo de la tabla de datos maestros y de los datos raw.

Se puede observar que la tabla de datos raw tiene mucha información faltante. Para esto se desarrollaron numerosas reglas que usan los distintos campos para intentar lograr el match entre registros.

Como primer paso, el proceso crea tablas temporales basadas en las tablas originales de los datos maestros y raw. La particularidad de estas tablas es que el proceso se encarga de "limpiar" la información, eliminando caracteres extraños, espacios y llevando todas las letras a minúscula. Esto se realiza para tener una mayor chance de matchear, sin necesidad de aplicar la modificación en la tabla original.

En el anexo se puede observar un ejemplo de la tabla temporal creada extrayendo los datos raw y aplicando las limpiezas necesarias en los campos.

En la columna del nombre puede observarse claramente cómo funciona la limpieza, quedando "April O'Donell" como "aprilodonell". Esto facilita el matching, ya que la información raw bien podría haber sido "April O-donell" (con guion en vez de apóstrofe) o similar, haciendo que un match exacto no fuera posible.

Esto es realizado también con la tabla de datos maestros.

Una vez creadas las dos tablas temporales, se inicia la ejecución de las reglas. El proceso crea una tabla temporal con la información raw y la del maestro con la que encontró la coincidencia.

Esta tabla es luego analizada y se buscan registros duplicados. En términos del proceso, un registro es duplicado si se encontró más de un maestro para ese mismo raw, en cuyo caso no se considera un match, ya que no se puede saber con certeza cuál es el maestro correcto.

En el anexo podemos observar la tabla de resultados de la primera regla.

Puede observarse que en todos los casos el CRN y el código postal coinciden. En el caso del código postal, se considera coincidencia porque el proceso al limpiar la información del

maestro, el código postal que aparece por ejemplo como “F-35769”, se eliminan las letras y caracteres extraños (el guion en este caso) y termina quedando “35769”, llegando a coincidir con el del raw, el cual está informado incompletamente.

Una vez finalizada la limpieza de registros duplicados en la tabla, se procede a la actualización de la tabla original, que será la de datos raw, en donde se actualiza el campo que relaciona al raw con su maestro, que es la clave foránea reseller_company_id.

Esto se repite para cada regla, donde se realiza lo mismo usando distintas condiciones.

Finalmente, como última regla se ejecuta la nueva regla Double Metaphone, la cual evalúa usando la pronunciación de las palabras.

En el anexo se verá la tabla de resultados de la regla Double Metaphone.

Como se puede ver, en muchos casos la coincidencia es parcial, habiendo diferencias sutiles en el nombre, o en la dirección postal, diferencias que otras reglas no lo considerarían un match ya que no es exacto. La ventaja de la regla double metaphone es que permite esa flexibilidad para lograr coincidencias que de otra forma no hubieran sido posibles.

7. Conclusión

En este trabajo se abordó el desafío de mejorar el matching de cadenas en PostgreSQL, un sistema de gestión de bases de datos relacionales ampliamente utilizado [Bertone, R., & Thomas, P. (2011)]. A través de una revisión exhaustiva de la literatura y el análisis de diferentes enfoques, se propuso un procedimiento personalizado que aprovecha las capacidades de PostgreSQL y proporciona una solución más eficiente y precisa para el matching de cadenas de texto.

Durante la investigación, se exploraron varios algoritmos y técnicas, incluyendo el algoritmo SimString [Kimura, M., & Sato, T. (2012)], el uso de trigramas y la distancia de edición de Levenshtein [Aumüller, M., Faust, A., & Schweikardt, N. (2005)]. Estas técnicas demostraron ser eficaces para mejorar la precisión y el rendimiento del matching de cadenas de texto en PostgreSQL, permitiendo encontrar coincidencias incluso en presencia de variaciones, errores ortográficos o diferencias léxicas.

Además, se realizó una evaluación exhaustiva del procedimiento propuesto, comparándolo con las funciones de búsqueda de cadenas de texto incorporadas en PostgreSQL. Los resultados obtenidos mostraron mejoras significativas en términos de precisión y tiempo de ejecución, lo que valida la utilidad y eficacia de la solución propuesta.

Es importante destacar que el procedimiento desarrollado tiene un impacto significativo en diversos entornos y aplicaciones, como la búsqueda de texto en motores de búsqueda, la deduplicación de registros en bases de datos y la corrección de datos erróneos o mal escritos.

Al mejorar la capacidad de realizar matching de cadenas en PostgreSQL, se promueve la calidad y la eficiencia en la gestión de información, lo cual es crucial en la era de los grandes volúmenes de datos.

Aunque se lograron avances significativos en esta investigación, siempre hay oportunidades para futuros desarrollos y mejoras. Se sugiere continuar explorando técnicas adicionales, como el uso de algoritmos basados en aprendizaje automático, para ampliar aún más las capacidades de matching de cadenas en PostgreSQL.

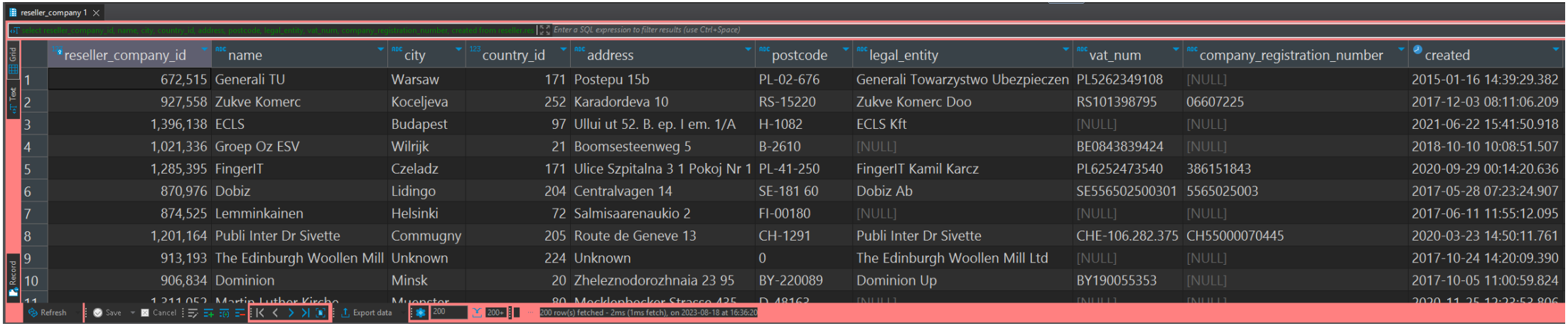
En resumen, el presente trabajo ha demostrado la importancia de abordar el problema del matching de cadenas en PostgreSQL y ha propuesto un procedimiento personalizado que mejora significativamente la eficiencia y la precisión de esta tarea. Los resultados obtenidos brindan una base sólida para futuras investigaciones y aplicaciones prácticas en el ámbito de la gestión de bases de datos y la recuperación de información.

8. Bibliografía

- Aumüller, M., Faust, A., & Schweikardt, N. (2005). Efficient string similarity joins with edit-distance constraints. En Proceedings of the 31st international conference on Very large data bases (pp. 799-810).
- Baeza-Yates, R., Navarro, G. (2001) Fast Approximate String Matching in a Dictionary Repositorio Institucional. Universidad de Chile. <https://users.dcc.uchile.cl/~gnavarro/ps/spire98.2.pdf>
- Bayardo, R. J., & Ma, Y. (2007). Suffix arrays for string joins and near-neighbor problems. ACM Transactions on Database Systems (TODS), 32(3), 1-44.
- Bertone, R., & Thomas, P. (2011). Introducción a las Bases de Datos: Fundamentos y Diseño (1a ed.). Buenos Aires: Prentice Hall - Pearson Education.
- Friedl, J. E. F. (2006). Mastering Regular Expressions. O'Reilly Media.
- Gennick, J. (2010). SQL Pocket Guide. O'Reilly Media.
- Kimura, M., & Sato, T. (2012). Sim String: A fast and scalable approach to string similarity search and approximate dictionary matching. Software: Practice and Experience, 42(3), 333-356. < https://link.springer.com/article/10.1007/s11704-015-5900-5>
- Okazaki, N., & Tsujii, J. (2010). Simple and efficient algorithm for approximate dictionary matching. En Proceedings of the 23rd International Conference on Computational Linguistics: Posters (pp. 905-913).<https://www.researchgate.net/publication/221101753_Simple_and_Efficient_Algorithm_for_Approximate_Dictionary_Matching>
- PostgreSQL. "PostgreSQL: The world's most advanced open-source database." https://www.postgresql.org/. Web.
- Sasaki, Y., & Hirao, T. (2007). Efficient string similarity search for similarity thresholds. En Proceedings of the ACM symposium on Document engineering (pp. 185-194). <https://link.springer.com/chapter/10.1007/978-3-319-18120-2_32
- Truong, D. V., Nguyen, T. T., & Nguyen, M. T. (2013). Using the trigram similarity for record linkage. En Proceedings of the Third International Conference on Communications and Electronics (pp. 21-26). < https://scholar.google.com/citations?user=K-ijSboAAAAJ>

Anexo

Tabla Reseller Company



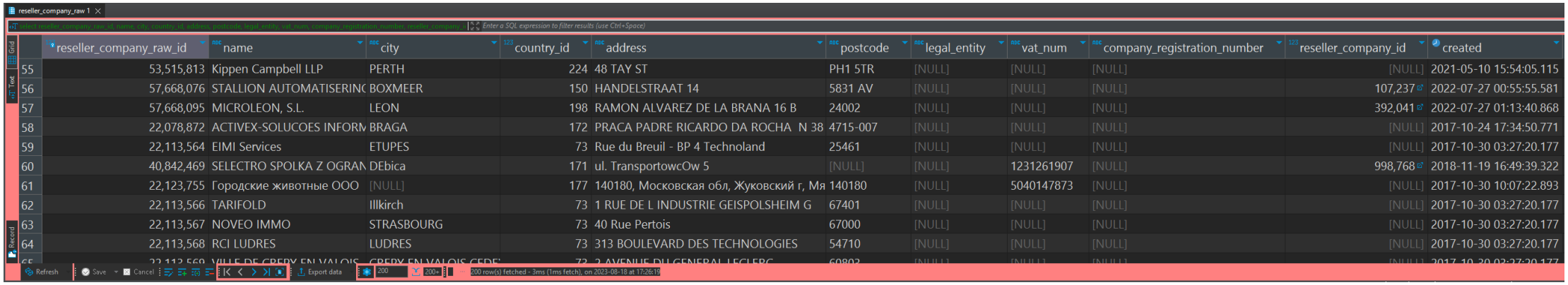
reseller_company 1

	reseller_company_id	name	city	country_id	address	postcode	legal_entity	vat_num	company_registration_number	created
1	672,515	Generali TU	Warsaw	171	Postepu 15b	PL-02-676	Generali Towarzystwo Ubezpieczen	PL5262349108	[NULL]	2015-01-16 14:39:29.382
2	927,558	Zukve Komerc	Koceljeva	252	Karadordeva 10	RS-15220	Zukve Komerc Doo	RS101398795	06607225	2017-12-03 08:11:06.209
3	1,396,138	ECLS	Budapest	97	Ullui ut 52. B. ep. I em. 1/A	H-1082	ECLS Kft	[NULL]	[NULL]	2021-06-22 15:41:50.918
4	1,021,336	Groep Oz ESV	Wilrijk	21	Boomsesteenweg 5	B-2610	[NULL]	BE0843839424	[NULL]	2018-10-10 10:08:51.507
5	1,285,395	FingerIT	Czeladz	171	Ulice Szpitalna 3 1 Pokoj Nr 1	PL-41-250	FingerIT Kamil Karcz	PL6252473540	386151843	2020-09-29 00:14:20.636
6	870,976	Dobiz	Lidingo	204	Centralvagen 14	SE-181 60	Dobiz Ab	SE556502500301	5565025003	2017-05-28 07:23:24.907
7	874,525	Lemminkainen	Helsinki	72	Salmisaarenaukio 2	FI-00180	[NULL]	[NULL]	[NULL]	2017-06-11 11:55:12.095
8	1,201,164	Publi Inter Dr Sivette	Commugny	205	Route de Geneve 13	CH-1291	Publi Inter Dr Sivette	CHE-106.282.375	CH55000070445	2020-03-23 14:50:11.761
9	913,193	The Edinburgh Woollen Mill	Unknown	224	Unknown	0	The Edinburgh Woollen Mill Ltd	[NULL]	[NULL]	2017-10-24 14:20:09.390
10	906,834	Dominion	Minsk	20	Zheleznodorozhnaia 23 95	BY-220089	Dominion Up	BY190055353	[NULL]	2017-10-05 11:00:59.824

Refresh Save Cancel Export data 200 row(s) fetched - 2ms (1ms fetch), on 2023-08-18 at 16:36:20

Fuente: Elaboración propia

Tabla Reseller Company Raw



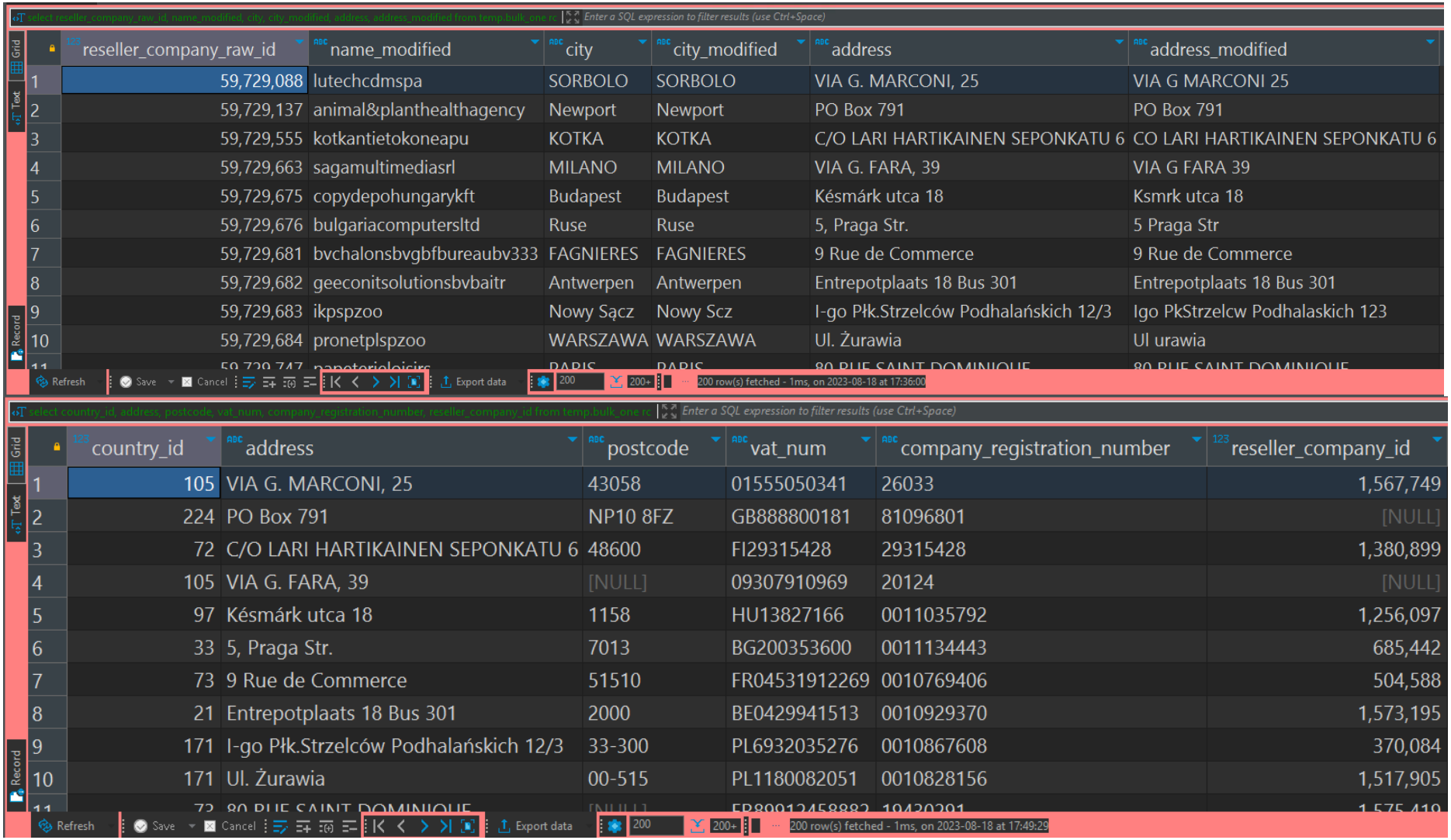
reseller_company_raw 1

	reseller_company_raw_id	name	city	country_id	address	postcode	legal_entity	vat_num	company_registration_number	reseller_company_id	created
55	53,515,813	Kippen Campbell LLP	PERTH	224	48 TAY ST	PH1 5TR	[NULL]	[NULL]	[NULL]	[NULL]	2021-05-10 15:54:05.115
56	57,668,076	STALLION AUTOMATISERING	BOXMEER	150	HANDELSTRAAT 14	5831 AV	[NULL]	[NULL]	[NULL]	107,237	2022-07-27 00:55:55.581
57	57,668,095	MICROLEON, S.L.	LEON	198	RAMON ALVAREZ DE LA BRANA 16 B	24002	[NULL]	[NULL]	[NULL]	392,041	2022-07-27 01:13:40.868
58	22,078,872	ACTIVEX-SOLUCOES INFORM	BRAGA	172	PRACA PADRE RICARDO DA ROCHA N 38	4715-007	[NULL]	[NULL]	[NULL]	[NULL]	2017-10-24 17:34:50.771
59	22,113,564	EIMI Services	ETUPES	73	Rue du Breuil - BP 4 Technoland	25461	[NULL]	[NULL]	[NULL]	[NULL]	2017-10-30 03:27:20.177
60	40,842,469	SELECTRO SPOLKA Z OGRAN	DEbica	171	ul. TransportowcOw 5	[NULL]	[NULL]	1231261907	[NULL]	998,768	2018-11-19 16:49:39.322
61	22,123,755	Городские животные ООО	[NULL]	177	140180, Московская обл, Жуковский г, Мя	140180	[NULL]	5040147873	[NULL]	[NULL]	2017-10-30 10:07:22.893
62	22,113,566	TARIFOLD	Illkirch	73	1 RUE DE L INDUSTRIE GEISPOLSHEIM G	67401	[NULL]	[NULL]	[NULL]	[NULL]	2017-10-30 03:27:20.177
63	22,113,567	NOVEO IMMO	STRASBOURG	73	40 Rue Pertois	67000	[NULL]	[NULL]	[NULL]	[NULL]	2017-10-30 03:27:20.177
64	22,113,568	RCI LUDRES	LUDRES	73	313 BOULEVARD DES TECHNOLOGIES	54710	[NULL]	[NULL]	[NULL]	[NULL]	2017-10-30 03:27:20.177

Refresh Save Cancel Export data 200 row(s) fetched - 3ms (1ms fetch), on 2023-08-18 at 17:26:19

Fuente: Elaboración propia

Tabla temporal de datos raw limpios

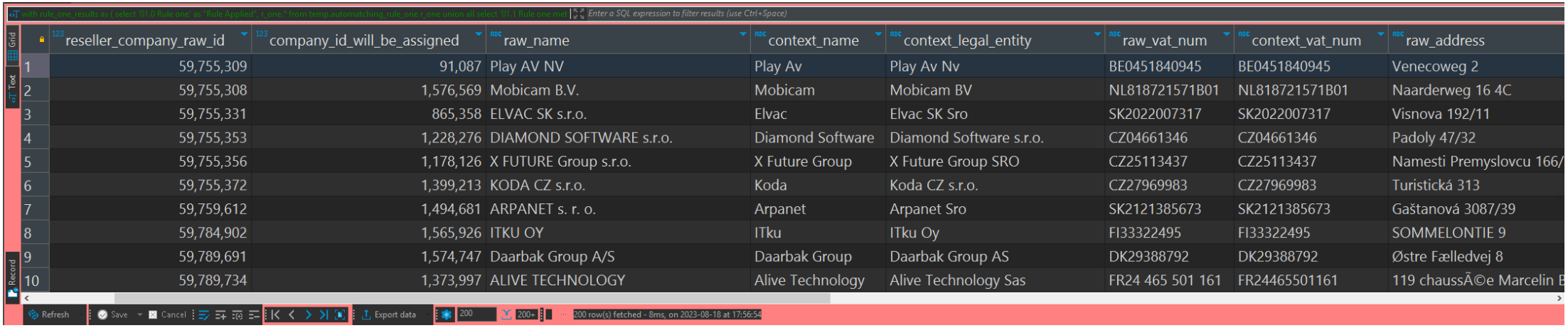


	reseller_company_raw_id	name_modified	city	city_modified	address	address_modified
1	59,729,088	lutechcdmspa	SORBOLO	SORBOLO	VIA G. MARCONI, 25	VIA G MARCONI 25
2	59,729,137	animal&planthealthagency	Newport	Newport	PO Box 791	PO Box 791
3	59,729,555	kotkantietokoneapu	KOTKA	KOTKA	C/O LARI HARTIKAINEN SEPONKATU 6	CO LARI HARTIKAINEN SEPONKATU 6
4	59,729,663	sagamultimediasrl	MILANO	MILANO	VIA G. FARA, 39	VIA G FARA 39
5	59,729,675	copydepohungarykft	Budapest	Budapest	Késmárk utca 18	Ksmrk utca 18
6	59,729,676	bulgariacomputersltd	Ruse	Ruse	5, Praga Str.	5 Praga Str
7	59,729,681	bvchalonsbvgbfbureaubv333	FAGNIERES	FAGNIERES	9 Rue de Commerce	9 Rue de Commerce
8	59,729,682	geeconitsolutionsbvbaitr	Antwerpen	Antwerpen	Entrepotplaats 18 Bus 301	Entrepotplaats 18 Bus 301
9	59,729,683	ikpspzoo	Nowy Sącz	Nowy Scz	I-go Płk.Strzelców Podhalańskich 12/3	Igo PkStrzelcw Podhalaskich 123
10	59,729,684	pronetplspzoo	WARSZAWA	WARSZAWA	Ul. Żurawia	Ul urawia

	country_id	address	postcode	vat_num	company_registration_number	reseller_company_id
1	105	VIA G. MARCONI, 25	43058	01555050341	26033	1,567,749
2	224	PO Box 791	NP10 8FZ	GB888800181	81096801	[NULL]
3	72	C/O LARI HARTIKAINEN SEPONKATU 6	48600	FI29315428	29315428	1,380,899
4	105	VIA G. FARA, 39	[NULL]	09307910969	20124	[NULL]
5	97	Késmárk utca 18	1158	HU13827166	0011035792	1,256,097
6	33	5, Praga Str.	7013	BG200353600	0011134443	685,442
7	73	9 Rue de Commerce	51510	FR04531912269	0010769406	504,588
8	21	Entrepotplaats 18 Bus 301	2000	BE0429941513	0010929370	1,573,195
9	171	I-go Płk.Strzelców Podhalańskich 12/3	33-300	PL6932035276	0010867608	370,084
10	171	Ul. Żurawia	00-515	PL1180082051	0010828156	1,517,905

Fuente: Elaboración propia

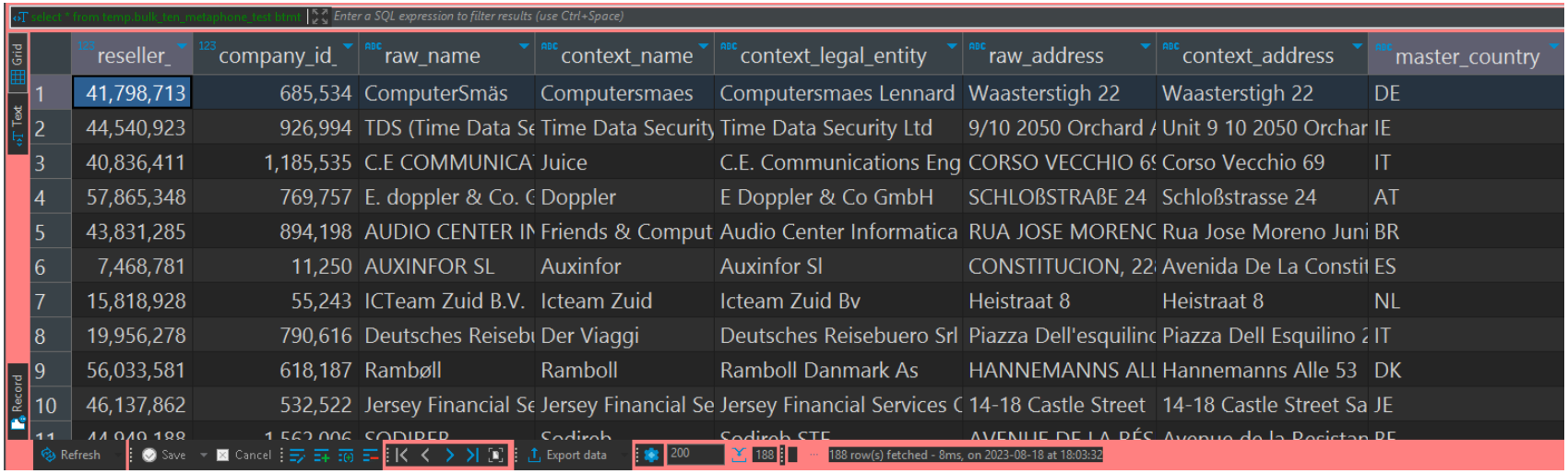
Tabla de resultados de la primera regla



	reseller_company_raw_id	company_id_will_be_assigned	raw_name	context_name	context_legal_entity	raw_vat_num	context_vat_num	raw_address
1	59,755,309	91,087	Play AV NV	Play Av	Play Av Nv	BE0451840945	BE0451840945	Venecoweg 2
2	59,755,308	1,576,569	Mobicam B.V.	Mobicam	Mobicam BV	NL818721571B01	NL818721571B01	Naarderweg 16 4C
3	59,755,331	865,358	ELVAC SK s.r.o.	Elvac	Elvac SK Sro	SK2022007317	SK2022007317	Visnova 192/11
4	59,755,353	1,228,276	DIAMOND SOFTWARE s.r.o.	Diamond Software	Diamond Software s.r.o.	CZ04661346	CZ04661346	Padoly 47/32
5	59,755,356	1,178,126	X FUTURE Group s.r.o.	X Future Group	X Future Group SRO	CZ25113437	CZ25113437	Namesti Premyslovcu 166/
6	59,755,372	1,399,213	KODA CZ s.r.o.	Koda	Koda CZ s.r.o.	CZ27969983	CZ27969983	Turistická 313
7	59,759,612	1,494,681	ARPANET s. r. o.	Arpanet	Arpanet Sro	SK2121385673	SK2121385673	Gaštanová 3087/39
8	59,784,902	1,565,926	ITKU OY	ITku	ITku Oy	FI33322495	FI33322495	SOMMELONTIE 9
9	59,789,691	1,574,747	Daarbak Group A/S	Daarbak Group	Daarbak Group AS	DK29388792	DK29388792	Østre Fælledvej 8
10	59,789,734	1,373,997	ALIVE TECHNOLOGY	Alive Technology	Alive Technology Sas	FR24 465 501 161	FR24465501161	119 chaussÃ©e Marcelin B

Fuente: Elaboración propia

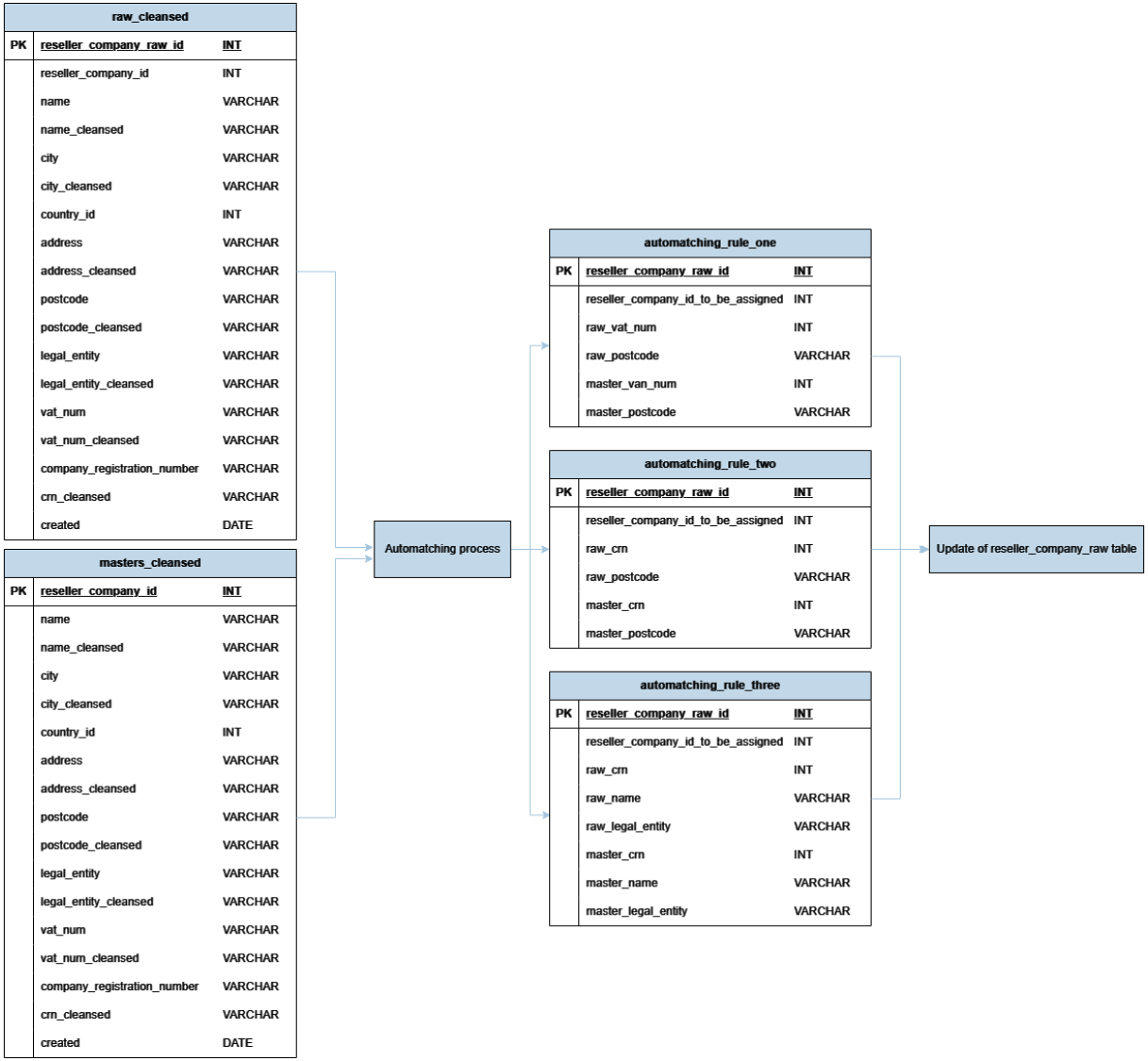
Tabla de resultados de la regla Double Metaphone



	reseller_	company_id_	raw_name	context_name	context_legal_entity	raw_address	context_address	master_country
1	41,798,713	685,534	ComputerSmäs	Computersmaes	Computersmaes Lennard	Waasterstigh 22	Waasterstigh 22	DE
2	44,540,923	926,994	TDS (Time Data Se	Time Data Security	Time Data Security Ltd	9/10 2050 Orchard A	Unit 9 10 2050 Orchar	IE
3	40,836,411	1,185,535	C.E COMMUNICA	Juice	C.E. Communications Eng	CORSO VECCHIO 6	Corso Vecchio 69	IT
4	57,865,348	769,757	E. doppler & Co. (	Doppler	E Doppler & Co GmbH	SCHLOßSTRAßE 24	Schloßstrasse 24	AT
5	43,831,285	894,198	AUDIO CENTER IN	Friends & Comput	Audio Center Informatica	RUA JOSE MORENC	Rua Jose Moreno Juni	BR
6	7,468,781	11,250	AUXINFOR SL	Auxinfor	Auxinfor Sl	CONSTITUCION, 22	Avenida De La Consti	ES
7	15,818,928	55,243	ICTeam Zuid B.V.	Icteam Zuid	Icteam Zuid Bv	Heistraat 8	Heistraat 8	NL
8	19,956,278	790,616	Deutsches Reiseb	Der Viaggi	Deutsches Reisebuero Srl	Piazza Dell'esquilinc	Piazza Dell Esquilino 2	IT
9	56,033,581	618,187	Rambøll	Ramboll	Ramboll Danmark As	HANNEMANNS ALL	Hannemanns Alle 53	DK
10	46,137,862	532,522	Jersey Financial Se	Jersey Financial Se	Jersey Financial Services (	14-18 Castle Street	14-18 Castle Street Sa	JE

Fuente: Elaboración propia

Estructura de las tablas de Automatching. https://t.ly/tDPUT



Fuente: Elaboración propia

Código SQL del procedimiento de automatching.

```
CREATE OR REPLACE FUNCTION reseller.automatching()
RETURNS VOID AS
$BODY$
DECLARE

/*
 *    Declaración de variables
 */
     total_rows INTEGER;

     -- Las siguientes tres variables se van a usar como
información extra para ayudar al matching. En este ejemplo, se van a
usar extensiones legales (LTD, SA, etc), códigos postales y números
VAT de empresas.
     v_legal_extension VARCHAR;
     v_postcode VARCHAR;
     v_vat_num VARCHAR;

BEGIN

-- Extensiones legales

SELECT INTO v_legal_extension
ARRAY_TO_STRING(ARRAY_AGG(legal_extensions), '|') FROM
reseller.legal_extensions;

-- Postcode

SELECT INTO v_postcode ARRAY_TO_STRING(ARRAY_AGG(postcode), '|')
FROM reseller.postcode;

-- Número VAT (NIF)

SELECT INTO v_vat_num ARRAY_TO_STRING(ARRAY_AGG(vat_num), '|') FROM
reseller.vat_num;

-- El siguiente bloque extrae información para ser usada en el
proceso de automatching

DROP TABLE IF EXISTS reseller.reseller_company_reduced;
CREATE TABLE reseller.reseller_company_reduced AS
```

```
SELECT DISTINCT
      r.reseller_company_id,
      r.name,
      TRIM(REGEXP_REPLACE(REPLACE(TRIM(REGEXP_REPLACE(TRIM(LOWER(REP
LACE(REPLACE(r.name,' AND ',' & '),E'\'',''))),'[ |.|/|-]($$ ||
v_legal_extension || $$)','','gi')),'*',''),'( |-|/|,)','','g')) AS
name_modified,
      r.city,
      r.address,
      r.country_id,
      r.postcode,
      CASE WHEN r.country_id IN (73,80) THEN
SUBSTRING(REGEXP_REPLACE(r.postcode,'[^[:digit:]]','','gi'),1,5)
      ELSE TRIM(REGEXP_REPLACE(TRIM(LOWER(r.postcode)),'^(($$ ||
v_postcode || $$)(-| ))|EC$','','gi')) END AS postcode_modified,
      r.vat_num,
      TRIM(REGEXP_REPLACE(TRIM(LOWER(r.vat_num)),'^($$ || v_vat_num
|| $$)','','gi')) AS vat_num_modified,
      r.company_registration_number,
      r.legal_entity,
      TRIM(REGEXP_REPLACE(TRIM(LOWER(r.legal_entity)),'[ |.|/|-]($$
|| v_legal_extension || $$)','','gi')) AS legal_entity_modified
FROM reseller.reseller_company AS r;

DROP TABLE IF EXISTS reseller.reseller_company_raw_reduced;
CREATE TABLE reseller.reseller_company_raw_reduced AS
SELECT     reseller_company_raw_id,
      name,
      CASE WHEN country_id IN (171,177)
            THEN
reseller.translation(TRIM(REGEXP_REPLACE(REPLACE(TRIM(REGEXP_REPLACE
(TRIM(LOWER(REPLACE(REPLACE(name,' AND ',' &
'),E'\'',''))),'[ |.|/|-]($$ || v_legal_extension ||
$$)','','gi')),'*',''),'( |-|/|,)','','g')),country_id)
            ELSE
TRIM(REGEXP_REPLACE(REPLACE(TRIM(REGEXP_REPLACE(TRIM(LOWER(REPLACE(R
EPLACE(name,' AND ',' & '),E'\'',''))),'[ |.|/|-]($$ ||
v_legal_extension || $$)','','gi')),'*',''),'( |-|/|,)','','g'))
      END   AS name_modified,
      city,
      address,
      country_id,
      postcode,
```

```
      CASE WHEN country_id IN (73,80)
            THEN
SUBSTRING(REGEXP_REPLACE(postcode,'[^[:digit:]]','','gi'),1,5)
            ELSE TRIM(REGEXP_REPLACE(TRIM(LOWER(postcode)),'^(($$ ||
v_postcode || $$)(-| ))|EC$','','gi'))
      END AS postcode_modified,
      vat_num,
      TRIM(REGEXP_REPLACE(TRIM(LOWER(vat_num)),'^($$ || v_vat_num ||
$$)','','gi')) AS vat_num_modified,
      company_registration_number
FROM reseller.reseller_company_raw
WHERE reseller_company_id IS NULL;

-- Double Metaphone for name, city AND address

DROP TABLE IF EXISTS
reseller.automatch_double_metaphone_name_city_address;
CREATE TABLE reseller.automatch_double_metaphone_name_city_address
AS
SELECT rraw.reseller_company_raw_id,
      r.reseller_company_id AS company_id_will_be_assigned,
      rraw.name AS raw_name,
      r.name AS master_name,
      rraw.city AS raw_city,
      r.city AS master_city,
      rraw.address AS raw_address,
      r.address AS master_address,
      (select c.country_standard from reseller.country c where
c.country_id = rraw.country_id) AS raw_country,
      (select c.country_standard from reseller.country c where
c.country_id = r.country_id) AS master_country
FROM reseller.reseller_company_raw_reduced AS rraw,
      reseller.reseller_company_reduced AS r
WHERE (TRIM(rraw.name) <> '' AND rraw.name IS NOT NULL)
      AND (TRIM(rraw.city) <> '' AND rraw.city IS NOT NULL)
      AND (TRIM(rraw.address) <> '' AND rraw.address IS NOT NULL)
      AND dmetaphone(r.name_modified) =
dmetaphone(rraw.name_modified)
      AND dmetaphone(r.city) = dmetaphone(rraw.city)
      AND dmetaphone(r.address) = dmetaphone(rraw.address);

DROP TABLE IF EXISTS
rule_automatch_double_metaphone_name_city_address_dupes;
```

```
CREATE TEMP TABLE
rule_automatch_double_metaphone_name_city_address_dupes AS
SELECT reseller_company_raw_id,COUNT(DISTINCT
company_id_will_be_assigned)
FROM reseller.automatch_double_metaphone_name_city_address
GROUP BY reseller_company_raw_id
HAVING COUNT(DISTINCT company_id_will_be_assigned) > 1
ORDER BY COUNT(DISTINCT company_id_will_be_assigned) DESC;

DROP TABLE IF EXISTS
dupes_for_rule_automatch_double_metaphone_name_city_address;
CREATE TEMP TABLE
dupes_for_rule_automatch_double_metaphone_name_city_address AS
SELECT *
FROM reseller.automatch_double_metaphone_name_city_address
WHERE reseller_company_raw_id IN (SELECT reseller_company_raw_id
FROM rule_automatch_double_metaphone_name_city_address_dupes)
ORDER BY reseller_company_raw_id;

DELETE FROM reseller.automatch_double_metaphone_name_city_address
WHERE reseller_company_raw_id IN (SELECT reseller_company_raw_id
FROM dupes_for_rule_automatch_double_metaphone_name_city_address);

UPDATE reseller.reseller_company_raw
SET reseller_company_id = company_id_will_be_assigned
FROM reseller.automatch_double_metaphone_name_city_address
WHERE reseller.reseller_company_raw.reseller_company_raw_id =
reseller.automatch_double_metaphone_name_city_address.reseller_compa
ny_raw_id;

DELETE FROM reseller.reseller_company_raw_reduced WHERE
reseller_company_raw_id IN (SELECT reseller_company_raw_id FROM
reseller.automatch_double_metaphone_name_city_address);

END;
$BODY$
LANGUAGE plpgsql;

SELECT reseller.automatching();
```