UNIVERSIDAD DE VALLADOLID

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALMACENAMIENTO DE DATOS SOBRE POLÍTICAS DE PRIVACIDAD BASADO EN TÉCNICAS DE ETL.

ESCUELA DE INGENIERÍA INFORMÁTICA

MENCIÓN EN COMPUTACIÓN

TRABAJO DE FIN DE GRADO

Alumno:
Adrián Sánchez Fernández

Tutora:
Mercedes Martínez González

Agradecimientos

Me gustaría dedicarle unas palabras a todas las personas que me han ayudado durante mis estudios. A mi tutora Mercedes, por su confianza, dedicación y apoyo durante estos últimos meses. A los amigos que encontré durante la carrera e hicieron el camino más ameno. A mi familia, por su cariño y ánimo siempre. A todo el que me apoyó durante este tiempo.

Gracias.

Resumen

El problema de la integración de la información resulta de la gran dispersión que existe de la información en distintos almacenamiento. En este trabajo se ha resuelto dicho problema de integración para información proveniente de distintas fuentes de datos sobre políticas de privacidad. Para resolver el problema se han utilizado técnicas de extracción carga y transformación de la información sobre un sistema de almacenamiento centralizado.

Abstract

The information integrity problem comes from the huge dispersion in the information over the different storage systems. In this project, it has been resolved for privacy policy datasets coming from different sources by using methods for extracting, loading and transforming the information into a centralized storage system.

Acrónimos y palabras clave:

- ETL: *Extract-Transform-Load*. Se conoce con estas siglas a los procesos encargados de extraer, transformar y cargar información desde una fuente a un sistema de almacenamiento.

- TFG: Trabajo Final de Grado.

- HTML: *Hypertext Markup Language*. Lenguaje estandarizado basado en etiquetas.

- CSV: *Comma Separated Values*. Formato que pueden tener los ficheros donde las columnas están delimitadas por comas y las filas por saltos de línea.

- URL: *Uniform Resource Locator.* Dirección dada a un recurso único en la web.

- JSON: *JavaScript Object Notation*. Formato ligero de intercambio de datos.

- TOSDR: *Terms of Service, Didn't Read*. Nombre de la tercera fuente de datos utilizada en este proyecto.

- API: *Application Programming Interface*. Conjunto de definiciones y protocolos que se usan para integrar y utilizar un *software*.

ÍNDICE

Capítulo 1

Introducción

1.1. Contexto:

En estos días está en auge el concepto del big data, el cual se refiere a cantidades masivas de datos, ya sea en términos de tamaño como de cantidad de información (filas en un CSV, por ejemplo). Junto con todo este auge, se potencia aún más el problema de la integración de la información, ya que las distintas fuentes que se usan, al ser muy grandes, representan un gran problema a la hora de integrarlas todas juntas. En este TFG se construirá un sistema que sea capaz de integrar fuentes de datos de varios GigaBytes de tamaño y obtener valor de toda esa información. Para este caso de uso se han escogido varias fuentes de datos sobre políticas de privacidad, aprovechando que también es un tema de actualidad.

1.2. Motivación:

En este proyecto se enfrenta el problema de la integración de la información desde distintas fuentes de datos a una común, unificando sus modelos de datos en uno solo. La motivación principal de este proyecto es integrar la información recogida de distintas fuentes, para este proyecto en concreto se utiliza información de datos sobre políticas de privacidad de web y empresas.

La razón por la cual se escogió dicho conjunto de datos es ayudar al usuario a tener acceso de manera fácil y rápida a la enorme cantidad de políticas de privacidad a las que da su consentimiento. Así pues, con este proyecto se pretende que exista un lugar en el que, utilizando una interfaz ajena al proyecto, un usuario sea capaz de consultar información referente a políticas de privacidad—tanto para conocer cómo le afectan a él mismo, como para averiguar en qué medida los cambios que realizan las empresas en sus políticas de privacidad pueden influir en el valor de estas, entre otros casos de uso.

1.3. Objetivos:

- Construir un ETL efectivo, modular, escalable y adaptable a otros conjuntos de datos.
- Unificar información sobre políticas de privacidad de distintos portales de datos.

1.4. Organización de la memoria:

Esta memoria se organiza en seis capítulos de la siguiente forma:

- **Capítulo 1. Introducción:** Descripción del problema a abordar y del proyecto que se desarrolla en esta memoria.

- **Capítulo 2. Planificación:** Descripción de la planificación inicial del proyecto, así como de la comparativa con la distribución real de las horas de trabajo. Se incluye también un análisis de los riesgos y una estimación del presupuesto.

- **Capítulo 3. Análisis de las fuentes de datos:** Se analizan las fuentes de datos que se van a usar, describiendo el modelo de datos de cada fuente así como una breve descripción de las mismas junto a la frecuencia de actualización de cada una

- **Capítulo 4. Análisis de la solución:** Se realiza el proceso de análisis del proyecto.

- **Capítulo 5. Diseño de la solución:** Se expone el diseño final de la solución escogida.

- **Capítulo 6. Conclusiones y trabajo futuro:** En este capítulo se detallan las conclusiones obtenidas de la realización del trabajo y el posible trabajo futuro sobre el mismo.

- **Capítulo 7. Apéndices técnicos:** Incluye los apéndices técnicos relativos al proyecto como scripts de descarga y carga de datos, procedimientos almacenados en la base de datos y los scripts de creación de las tablas.

Capítulo 2

Planificación

En este capítulo se exponen la planificación inicial del proyecto realizada antes de comenzar el mismo, los riesgos encontrados derivados de efectos externos a dicho proyecto, un análisis sobre si se ha cumplido o no la planificación inicial y una estimación del presupuesto del proyecto.

2.1. Planificación:

Para su planificación, el proyecto se ha dividido en dos partes. Primero se encuentra la aplicación en sí, que recibirá una carga de trabajo de 150 horas, y por otra parte se encuentra la realización de la memoria, la cual recibirá una carga de otras 150 horas. De esta manera se alcanza el total de 300 horas, la cantidad esperada de un Trabajo de Final de Grado de 12 créditos (equivaliendo cada crédito a 25 horas de trabajo.)

En la siguiente tabla se muestra la planificación inicial y la duración real del proyecto.

Tarea	Duración estimada	Fecha inicio	Fecha fin estimada	Fecha fin real	Duración real
Estudio y análisis de distintas fuentes de datos	30 horas	1 de diciembre	14 de diciembre	1 de abril	30 horas
Implementación y automatización de un sistema de extracción de las fuentes de datos	30 horas	11 de diciembre	7 de enero	10 de mayo	30 horas
Diseño de un modelo de datos donde unificar la información recopilada	40 horas	7 de enero	10 de febrero	20 de mayo	40 horas
Diseño e implementación de un sistema de almacenamiento de datos	20 horas	10 de febrero	15 marzo	5 de junio	30 horas

Diseño e implementación de un sistema de inserción y transformación de la información en el sistema de almacenamiento de datos.	30 horas	15 marzo	10 de abril	06 de julio	40 horas
Elaboración de la memoria	150 horas	1 de diciembre	14 de marzo	06 de julio	130 horas
TFG completo	300 horas	1 de diciembre	14 de marzo	06 de julio	300 horas

Tabla 1: Planificación temporal del proyecto.

El siguiente diagrama de Gantt muestra la planificación inicial de forma gráfica a lo largo del tiempo:

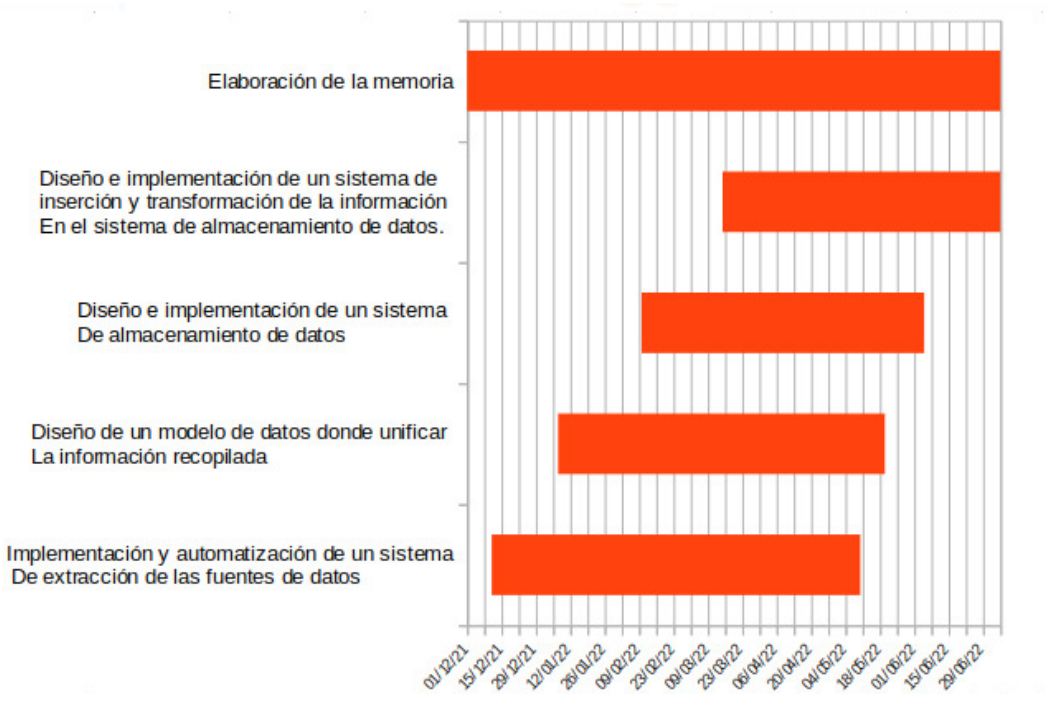

Imagen 1: Diagrama de Gantt del proyecto.

Las diferencias sustanciales entre la planificación inicial y la duración real del proyecto recaen en la falta de tiempo por parte del alumno para llevar a cabo la carga de

trabajo planificada, teniendo que repartir así las horas en más semanas de las esperadas. Dicha falta de tiempo ha venido provocado por la carga de trabajo del propio alumno en el puesto actual de trabajo que tiene. Aun así, el tiempo dedicado a las tareas ha sido el mismo que el planificado inicialmente solo que extendido más a lo largo del tiempo, haciendo que la entrega del proyecto programada se retrasase 3 meses.

2.2. Análisis de riesgos:

Riesgo	No disponibilidad de alguna de las fuentes previstas
Impacto	Alto
Probabilidad	Media
Plan de protección	Tener varias fuentes de datos, así como realizar la comprobación de que todo está disponible con tiempo suficiente para buscar más en caso de ser necesario.
Plan de contingencia	Buscar otras fuentes de datos a mayores de las que inicialmente se plantea usar, a modo de reserva.

1. Riesgo número 1: No disponibilidad de alguna de las fuentes previstas.

Riesgo	Enfermedad
Impacto	Alto
Probabilidad	Baja
Plan de protección	Holgura suficiente en la planificación de tareas
Plan de contingencia	Aprovechar la holgura en la planificación para cumplir con las fechas

2. Riesgo número 2: Enfermedad

Riesgo	Baja calidad de los datos de las fuentes
Impacto	Alto
Probabilidad	Media
Plan de protección	No pensar en el modelo hasta no haber analizado todas las fuentes y ver que problemas de integración pueden surgir
Plan de contingencia	Cargar los datos de distintas fuentes por separado y unificar los modelos dentro de la propia base de datos.

3. Riesgo número 3: Baja calidad de los datos de las fuentes

Riesgo	Pérdida del trabajo
Impacto	Alto
Probabilidad	Baja
Plan de protección	Realizar copia de seguridad a menudo.
Plan de contingencia	Utilizar dichas copias de seguridad en caso de perder alguna parte del trabajo

4. Riesgo número 4: Pérdida del trabajo

Riesgo	Colisión con otras obligaciones
Impacto	Alto
Probabilidad	Alta
Plan de protección	Tener tareas planeadas y reservar tiempo adecuadamente.
Plan de contingencia	Dar más prioridad a la realización de las tareas de este trabajo.

5. Riesgo número 5: Colisión con otras obligaciones

Riesgo	Saturación por el puesto de trabajo del alumno
Impacto	Medio
Probabilidad	Media
Plan de protección	Planificar con un margen de varios meses con respecto a la fecha final del proyecto.
Plan de contingencia	Aprovechar los días en los que la jornada no ha sido pesada para avanzar en el proyecto.

6. Riesgo número 6: Saturación por el puesto de trabajo del alumno

2.3. Estimación del presupuesto:

En este apartado se estimará el coste del proyecto, tanto en el equipo necesario como en los gastos de personal.

En gastos de equipos, el coste ha sido de 0 €. El equipo utilizado era propiedad del alumno y la máquina virtual utilizada para la construcción del sistema de almacenamiento fue proporcionado por la Universidad de Valladolid.

En cuanto a gasto personal, también ha sido de 0 €, ya que el alumno no cobrará por dicho trabajo. Sin embargo, puede estimarse el coste que habría conllevado si se hubiera cobrado por el trabajo. Teniendo en cuenta que el sueldo medio de un administrador de bases de datos en España es de 37.913 € brutos anuales[1], para una jornada de 40 horas a la semana, 2080 horas al año, serían 18’22 € brutos por hora, siendo el proyecto de 300 horas, el coste resultante sería de 5.468’22 € brutos.

[1] Dato obtenido de la plataforma *Glassdoor* a fecha 26 de junio de 2022: https://www.glassdoor.es/Sueldos/database-administrator-dba-sueldo-SRCH_KO0,26.htm

Capítulo 3

Análisis de las fuentes de datos

En este capítulo se presenta un análisis de las tres fuentes de datos que se van a utilizar.

El análisis consta de una descripción de alto nivel de la fuente de datos, una descripción del modelo de datos que nos proporciona, con una explicación de alto nivel para cada campo así como el tipo de dato de cada uno y la frecuencia de actualización de cada fuente en caso de que sigan actualizándose.

3.1. *PrivaSeer*:

Descripción de la fuente:

PrivaSeer[2] es un motor de búsqueda de política de privacidad creado en la Universidad de Ciencias de la Información y Tecnología del estado de Pennsylvania [1]. Su objetivo es hacer las políticas de privacidad más transparentes, detectables y buscables. Actualmente tienen almacenadas más de 1.4 millones de políticas recogidas de *Common Crawl*[3] [2] y *Free Company Dataset*[4] *[3]*. El 25 de mayo de 2021 se lanzó la primera versión.

Las políticas de privacidad están almacenadas en HTML, además también se disponen de unos ficheros de metadato en formato CSV, que serán los que trataremos en este trabajo mayoritariamente.

Descripción de los datos:

En la siguiente tabla se describen los datos que se encuentran en los conjuntos de datos de *PrivaSeer*.

[2] Link a la web de Privaseer: https://privaseer.ist.psu.edu/
[3] Link a la web de Common Crawl: https://commoncrawl.org/
[4] Link a la web de Free Company Dataset: https://docs.peopledatalabs.com/docs/free-company-dataset

Nombre del dato	Descripción del dato	Tipo del dato
hash	Código identificador único para cada política de privacidad.	varchar(42)
URL	URL desde donde se ha recogido dicha política de privacidad.	varchar(500)
file_path	En qué fichero interno se encuentra el fichero HTML correspondiente a dicha política de privacidad.	number
Probability	Probabilidad de que el fichero recogido sea una política de datos. Solo aquellos con probabilidad mayor que 0.5 están recogidos en el corpus[5].	float
Timestamp	Fecha en la que se recopiló la información sobre la política de privacidad.	date

Tabla 8. Modelo de datos de la fuente Privaseer.

Frecuencia de actualización:

No hay fecha de una siguiente actualización. Según se indica en su página web, los desarrolladores siguen trabajando para mejorar el motor de búsqueda sobre su conjunto de datos.

3.2. Princeton-Leuven Longitudinal Corpus of Privacy Policies:

Descripción de la fuente:

Se trata de un conjunto de datos elaborado por la Universidad de Princeton[6] a partir del cual se pueda estudiar la evolución y cambio de las políticas de privacidad a lo largo del tiempo para la misma empresa o web [4]. Este conjunto de

[5] Esta probabilidad se calcula desde *PrivaSeer* ya que ellos extraen los datos de una fuente de datos con muchas URL, e intentan localizar políticas de privacidad entre ellas con búsqueda de patrones como por ejemplo, URL que contenga privacy-policy en el nombre, por tanto algunas pueden no serlo, de ahí que calculen dicha probabilidad y la usen como filtro.

[6] Link a la web con del proyecto de la fuente de datos Princeton: https://privacypolicies.cs.princeton.edu/

datos contiene más de un millón de políticas de privacidad, todas en inglés, recibidas de más de 130.000 sitios web escogidos de Alexa Top 100K[7] [5].

El conjunto de datos se compone de políticas de privacidad en formato HTML y de metadatos asociados a ellas en formato JSON, que serán los que usaremos en este proyecto mayoritariamente. Las políticas de privacidad son desde los años 90, aunque más del 90% son posteriores al 2007.

Descripción de los datos:

En la siguiente tabla se describen los metadatos que encontramos en el *datasets* de Princeton University. Solo se describen los datos que son de utilidad para este TFG:

Nombre del dato	Descripción del dato	Tipo del dato
alexa_rank	Alexa Ranking[8]	int
analysis_subcorpus		boolean
categories	Categoría de la web (tecnología, negocios, agricultura, etc.)	varchar
char_count	Número de caracteres de la política de privacidad	int
classifier_probability	Probabilidad da por su *software* de ser verdaderamente una política de privacidad.	float
cross_domain_homepage_redir		boolean
flesch_ease		varchar
flesch_kincaid		float
homepage_snapshot_redirected_domain		varchar
homepage_snapshot_redirected_url	URL de donde se ha obtenido la política de privacidad	varchar
homepage_snapshot_url		varchar
interval		varchar

[7] LInk a la web de Alexa Top 100k: https://www.alexa.com/ (Service ended)

[8] ** Alexa ranking: es un ranking global que clasifica las páginas web de internet según su popularidad.

link_text		varchar
parked_domain		boolean
phase		varchar
policy_domain	Nombre del dominio	varchar
policy_filetype	Tipo del fichero para la política de privacidad (HTML)	varchar
policy_snapshot_url	URL de donde se ha cogido el snapshot de la política de privacidad	varchar
policy_title	Título de la política de privacidad	varchar
policy_url	URL de la política de privacidad	varchar
sha1	Identificador único (hash)	int
simhash		boolean
simhash_updated		varchar
site_hostname	Nombre del host	varchar
site_url	URL	varchar
smog		float
strict_updated		boolean
timestamp	Fecha de publicación	int
word_count	Número de palabras en la política de privacidad	int
year	Año de publicación de la política de privacidad	int

Tabla 9. Modelo de datos de la fuente Princeton-Leuven Longitudinal Corpus of Privacy Policies.

Frecuencia de actualización:

Bi-anual hasta 2019 (no se actualiza desde entonces)

3.3. “Terms of Service, Didn’t Read”:

Descripción de la fuente:

“Terms of Service, Didn’t Read”[9] Es un portal donde se analizan las políticas de privacidad de casi 200 empresas, viendo qué cambios han implementado y cómo podría afectar eso al sentimiento del usuario con dicha empresa, siendo este sentimiento casi un sinónimo de confianza [6]. Sus datos son accesibles a través de una API y obtenibles en formato JSON compacto.

Descripción de los datos:

En la siguiente tabla se describen los metadatos que se encuentran en el conjunto de datos de “Terms of Service, Didn’t Read”. Solo se describen los datos que son de utilidad para este TFG:

Nombre del dato	Descripción del dato	Tipo del dato
id	Identificador en el sistema	int
is_comprehensively_reviewed	Revisado por un humano o solo por una máquina	boolean
urls	URL del sitio	varchar
name	Nombre del sitio	varchar
status		varchar
Updated_at (timezone,pgsql,unix)	Fecha y lugar de cuando se actualizó	varchar-timestamp-int
Created_at (timezone,pgsql,unix)	Fecha y lugar de cuando se creó	varchar-timestamp-int
slug	nombre	varchar
wikipedia	URL de Wikipedia sobre el sitio web	varchar
Rating (hex,human,letter)		int-varchar-varchar
Links (phoenix,crisp)		json-json
Phoenix (service,documents,comment,new_edit)	Datos sobre TOSDR	varchar-varchar-varchar-varchar

[9] LInk a la web de “terms of service, didn’t read”: https://tosdr.org/

Crisp (api,service,badge)	Datos sobre TOSDR	varchar-varchar-json
Badge (svg,png)	Datos sobre TOSDR	varchar-varchar

Tabla 10: Modelo de datos de la fuente TOSDR.

Frecuencia de actualización: mensual.

Capítulo 4

Análisis de la solución

En el presente capítulo se muestra un análisis de la solución. Posteriormente se presenta cada una de las necesidades que surgen durante el desarrollo del proyecto y las tablas que las cubren, así como del modelo de datos final y los procesos encargados de extraer la información desde las fuentes hasta el sistema de almacenamiento.

4.1. Necesidades:

- Almacenar los datos recogidos de las fuentes con la menor modificación posible fuera de la base de datos, ya que todo se realizará dentro del sistema de almacenamiento. Para cubrir dicha necesidad se crea una tabla para cada una de las fuentes donde se insertarán los datos, con las menores modificaciones posibles, intentando que sean ninguna. No se describen todos los campos, porque se introducen todos aquellos descritos en el capítulo 3: análisis de las fuentes de datos, solo se indica cuál es la clave primaria para cada una de ellas.

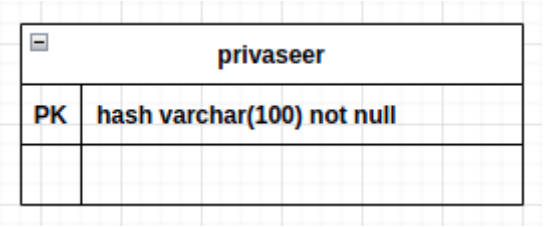

Imagen 2: Privaseer y su PK.

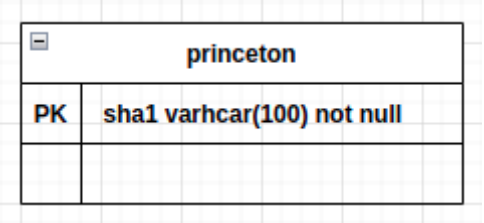

Imagen 3: Princeton y su PK.

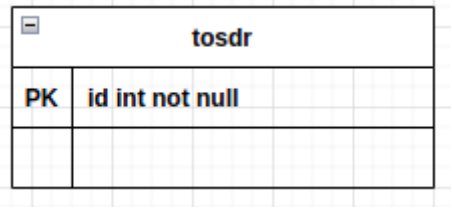

Imagen 4: tosdr y su PK.

- Encontraremos información para la misma empresa en las distintas bases de datos, necesitamos una tabla donde se relacionan los ID únicos de cada fuente entre ellos, y con el ID propio de esa entidad en nuestro sistema de almacenamiento. Dicha tabla se llamará ***mapping***, y nos permitirá relacionar cada entidad desde su fuente original hasta el sistema de almacenamiento creado, se añade un nombre para poder identificar si una entidad ya está en nuestro sistema aunque provenga de otra fuente distinta.

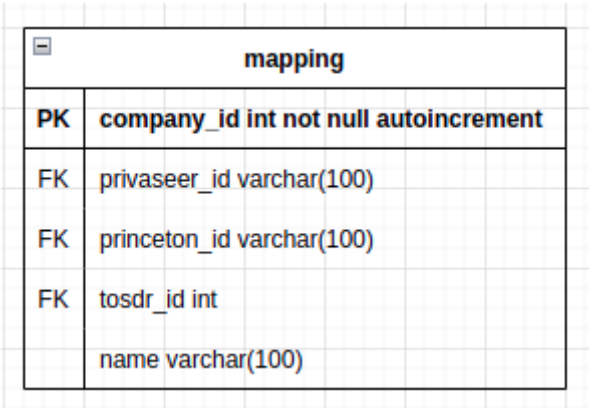

Imagen 5: tabla de* mapping *entre los ID de las 3 fuentes y el propio del sistema de almacenamiento creado.

- Almacenar la información útil relativa a las compañías. Se crea la tabla ***companies*** para cubrir esta necesidad, la información importante que se almacenará es la siguiente: (Se indica con 1 (Princeton), 2 (PrivaSeer) o 3 (TOSDR) cada campo dependiendo de qué fuente de datos provenga)**.**

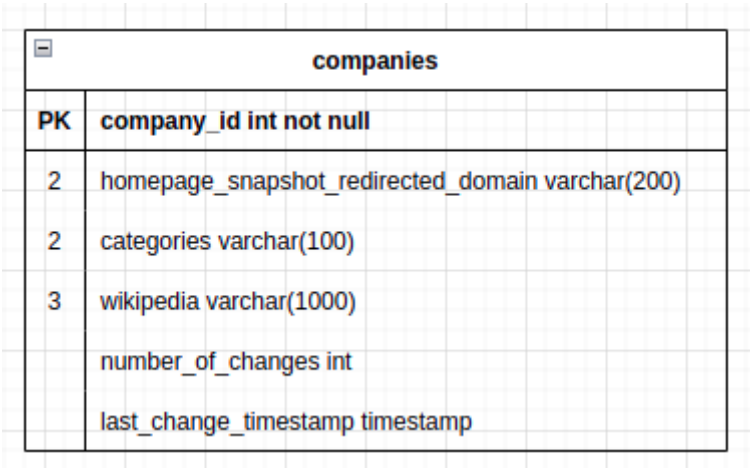

Imagen 6: tabla* companies*.

- Almacenar la información útil de las políticas de privacidad. Se crea la tabla ***policies*** para cubrir esta necesidad, la información importante que se almacenará es la siguiente: (Se indica con 1,2 o 3 cada campo dependiendo de qué fuente de datos provenga).

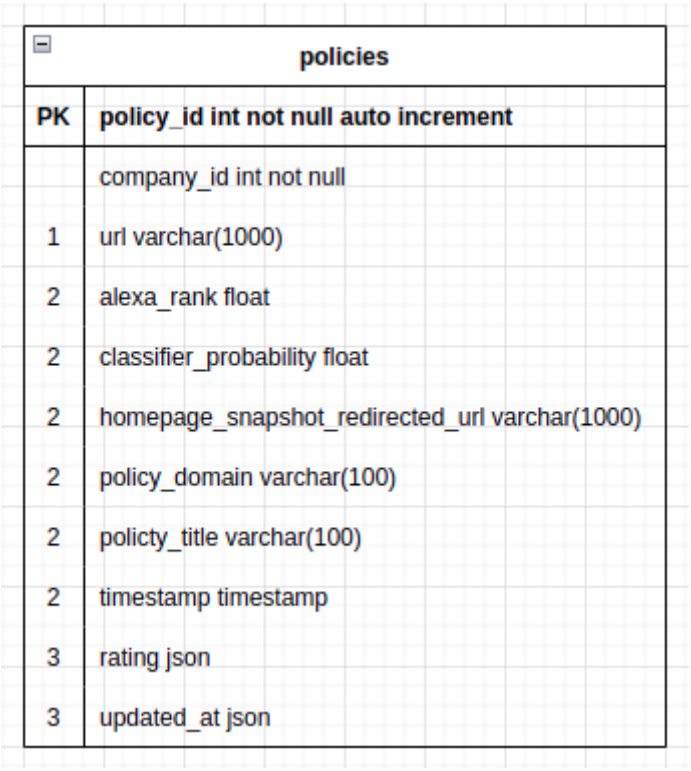

Imagen 7: tabla policies.

- Tener la posibilidad de ver cuántas veces una compañía ha cambiado sus políticas de privacidad, y si estos cambios han influido en alguno de los índices recogidos de las fuentes, como por ejemplo en la confianza del usuario. Para dicha necesidad, se crea la tabla ***company_changes*** donde almacenaremos aquellos cambios en las políticas ocurridas para cada una de las empresas de todo el conjunto de datos, especificando el ID de la política anterior, el ID de la nueva política, y la fecha del cambio en los casos en los que sea posible.

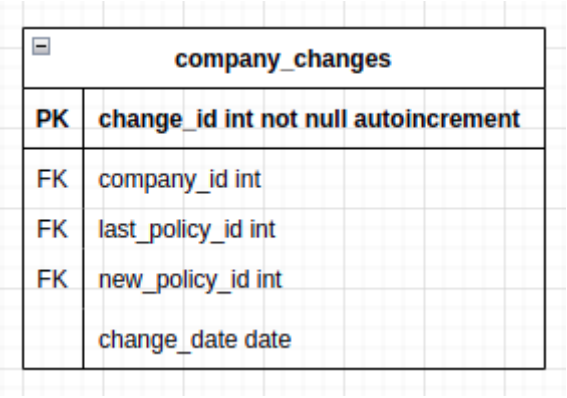

Imagen 8: tabla company_changes,. donde almacenaremos los cambios en las políticas de privacidad que ha sufrido o sufran las empresas de nuestro set de datos

- Ser capaces de registrar cada una de las ejecuciones de nuestro proceso integrador de los datos, así como los mensaje de registro propios de un sistema de integración tales como, cuántas líneas han sido actualizadas o en caso de suceder cualquier error dejar constancia del mensaje en dicha tablas.

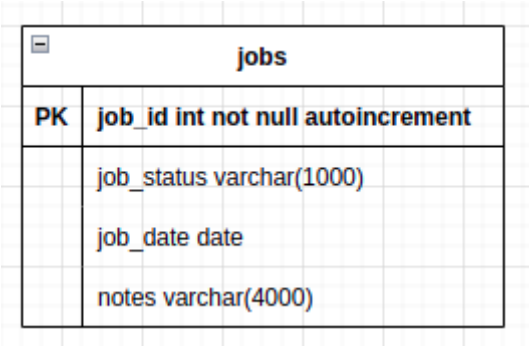

Imagen 9: Tabla con información de cada una de las ejecuciones.

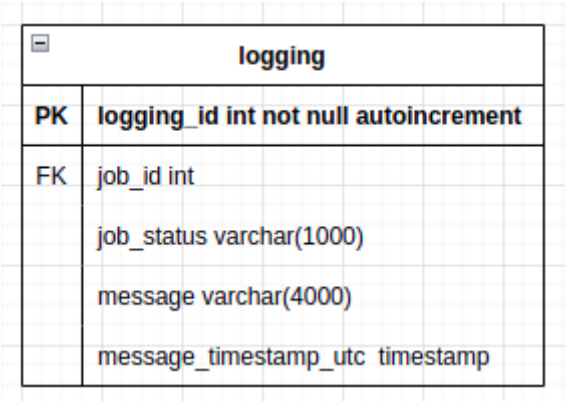

Imagen 10: Tabla de* logging *con cada uno de los mensajes registrados en cada una de las ejecuciones del sistema integrador de la información.

4.2. Paradigma datawarehouse:

Se conoce data warehouse como una base de datos centralizada. Típicamente en él, se resuelve el problema de la integración de la información, dicha información puede venir de distintas fuentes de datos, o ser distintos conjuntos de datos provenientes de una misma fuente. Un ejemplo de esto último podría ser una empresa, la cuál ha decidido construir un data warehouse para almacenar sus datos sobre contabilidad, inventario y empleados, con el fin de obtener una visión distinta integrando dichos conjuntos de datos. Este es el caso de uso más común para un datawarehouse, la obtención de nueva información a partir de la unificación de distintos conjuntos de datos.

Para que la información llegue desde la fuente al data warehouse, son necesarios un conjunto de funciones y procedimientos que extraen, transforman y cargan los datos desde la fuente al data warehouse, a estos conjuntos se los conoce como ETL.

Según el libro *Principles of Data Integration* de Anhai Doan[10], un ETL puede realizar distintas tareas, como por ejemplo:

- Filtrados en la importación: Estos incluyen analizadores para distintos formatos de archivos o procesos para ver si los datos vienen de una base de datos relacional.

- Transformaciones de datos: Pueden unir, agregar o filtrar datos.

- Deduplicación: dichas herramientas buscan determinar cuándo múltiples registros se refieren a una misma entidad.

- Perfilado: Estos procesos crean perfiles que constan de tablas u otra información que resume las propiedades de los datos que hay en el warehouse.

- Gestión de la calidad: Además de la deducción se pueden incluir otras pruebas como restringir ciertas combinaciones de valores o ciertos caracteres que no se ajustan a la codificación de nuestro data warehouse.

En nuestro caso, el datawarehouse será una base de datos relacional MySQL donde se integrará información sobre webs/empresas y sus políticas de privacidad desde 3 fuentes distintas. El objetivo es sacar valor de cuantos cambios y/o actualizaciones ha recibido la política de privacidad de una web/empresa a lo largo del tiempo.

Para integrar toda esta información, la cargamos, sin ninguna modificación, dentro de nuestro sistema de almacenamiento, por lo que en nuestro *data warehouse* tendremos almacenado tanto la información integrada en nuestro modelo de datos como la información que recibimos de las fuentes.

4.3. Modelo de datos unificado

El modelo de datos final consta de dos tablas principales, una con información relativa a la web/empresa (***companies***) y otra con información relativa a cada política de privacidad (***policies***).

En la tabla ***companies*** encontramos una clave primaria que es un índice único asignado por nuestro sistema a una web/empresa, dicho índice será una clave foránea en la tabla *policies*.

Por otro lado, en la tabla ***policies*** tendremos también un identificador único, asignado por nuestro sistema según el orden en el que la información sobre las políticas ha llegado a él.

[10] *Principles of Data Integration*, Anhai Doan, Alon Halevy and Zachary Ives. Ed Morgan Kaupmann ISBN: 978-0-12-416044-6

Como nuestro objetivo es poder rastrear la evolución de las políticas de privacidad a lo largo del tiempo, se crearon también dos tablas donde se registran cuando han sido los cambios para cada web/empresa, el nombre de la tabla es ***companies_changes***.

Dicha tabla contendrá información sobre cuando fue el cambio, cuál fue la política que se introdujo, cuál fue la anterior política, en caso de haberla y de no ser relativa a una nueva la empresa en el sistema, y la fecha cuando se cambió. Se utilizan además como claves foráneas las claves primarias de ***companies*** y ***policies*****,** para hacer más fácilmente rastreables los cambios y pertenencias de las políticas de privacidad.

La finalidad de tener una tabla así, es el valor que podemos sacar de analizar cuantas veces, cada cuánto y qué sentimiento se le otorga desde distintas fuentes a las distintas políticas de privacidad publicadas por una web/empresa. Por ejemplo, que una empresa haya actualizado su política de privacidad muchas veces en los últimos años obteniendo mejor valoración por nuestras fuentes, puede indicarnos que dicha web/empresa se molesta por la transparencia y el uso que hace de los datos de sus clientes, o lo contrario, una web/empresa con un mal sentimiento en su única política de privacidad publicada y sin haberla actualizado en años.

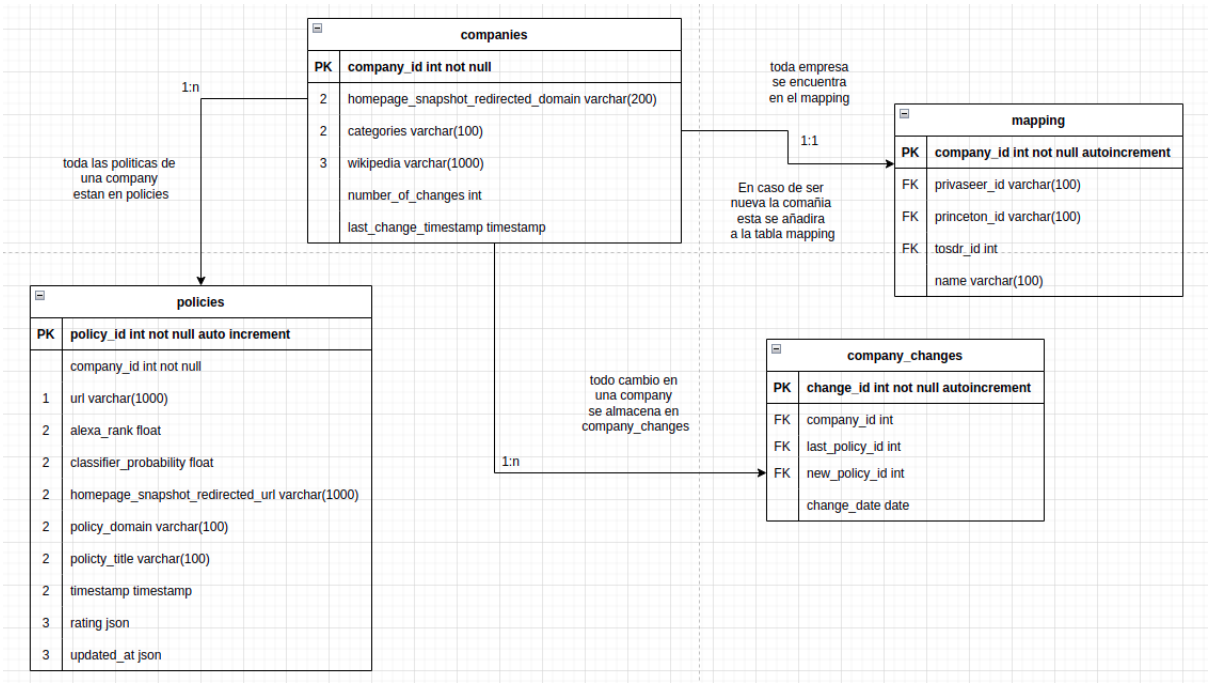

Imagen 11: Modelo de datos unificado

4.4. Histórico con cambios:

Para registrar todos los cambios que se realizan sobre las políticas de privacidad de una web/empresa, se ha construido un *procedure*[11] que se ejecuta en la carga de datos dentro del sistema de almacenamiento. Dicho *procedure* se encarga de introducir la información relativa a la nueva política de privacidad para la empresa en concreto que se ha cargado en el sistema para, de esta forma, introducir durante la misma ejecución la información sobre el cambio en la web/empresa y sobre la relación anterior política-política actual.

4.5. *Mapping*:

Para conseguir tener relacionadas todas las empresas de las distintas fuentes de datos, se construye una tabla de *mapping* donde almacenaremos el ID de cada fuente de datos para una misma empresa junto con el identificador que le asigna nuestro propio sistema, de esta forma podremos rastrear fácilmente los datos originales de cualquiera de los registros en nuestro modelos de datos final. En esta tabla es donde se genera el identificador único para cada empresa además de extraer el nombre de la empresa del dominio/url para así poder resolver el problema de la heterogeneidad del nombre. Dicho problema se refiere a los casos donde los entidades o realidades que son la misma se denominan de forma distinta en dos fuentes de datos, como el dominio/url es único para cada empresa, con la ayuda de expresiones regulares se extrae el nombre y así poder identificar una empresa en dos fuentes distintas antes de introducirse sus políticas en el sistema de almacenamiento.

4.6. Descripción de alto nivel de los ETL:

Cada fuente de datos tendrá su propio método de extracción y carga de datos dependiendo, por ejemplo, si se siguen actualizando o no. El análisis de los dicho métodos es el siguiente:

4.6.1 Privaseer:

Al ya no recibir actualizaciones, se hará una descarga manual del conjunto de datos y una carga manual del mismo. La integración en el sistema de almacenamiento junto con las demás fuentes se realizará a través del transformador.

[11] *Procedure*: subrutina escrita en un lenguaje de scripting almacenada en la base de datos, para este proyecto, escrito en PL/SQL. https://dev.mysql.com/doc/refman/8.0/en/create-procedure.html

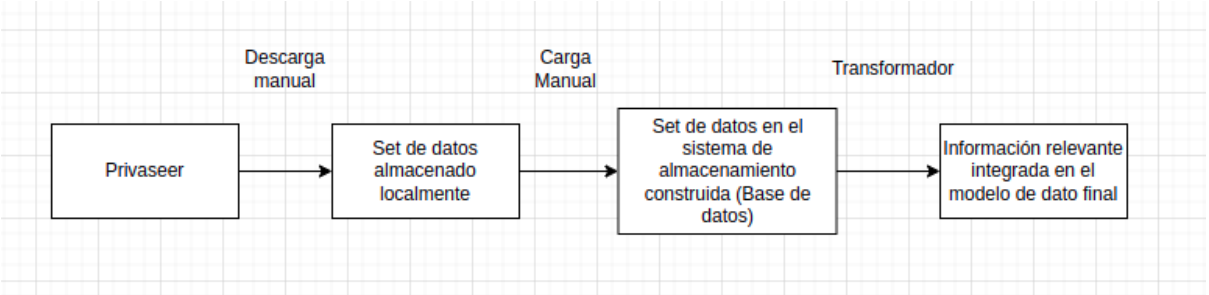

Imagen 12: Descripción de alto nivel del ETL para la fuente de datos privaseer.

4.6.2 Princeton:

Esta fuente de datos tampoco recibe actualizaciones, por lo que se seguirá el mismo proceso descrito anteriormente para Privaseer.

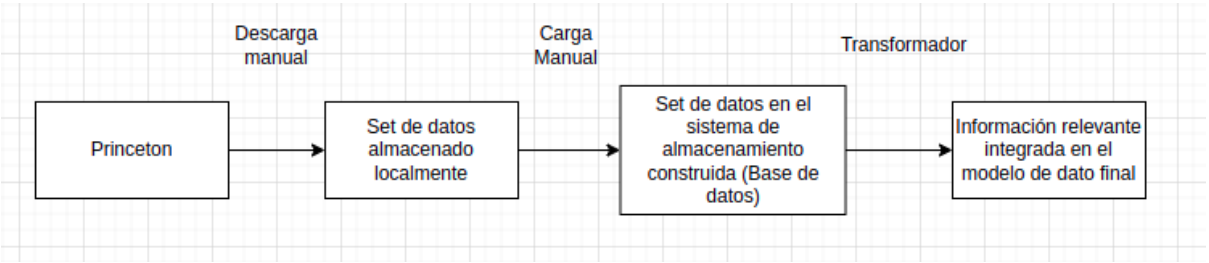

Imagen 13: Descripción de alto nivel del ETL para la fuente de datos Princenton.

4.6.3 TOSDR:

Dicha fuente si que recibe actualizaciones, por lo que se ha construido un script para poder automatizar la descarga de datos desde la propia fuente de datos hacia el servidor donde se ha construido el sistema de almacenamiento. En el script, además se realiza una transformación en cuanto a la estructura de los datos obtenidos, dividiéndolos en ficheros unitarios para cada política de privacidad, ya que es la forma en la que importar estos datos en el sistema de almacenamiento. No se realiza ninguna modificación relativa al contenido. Posteriormente se genera un fichero con la cláusulas sql necesarias para insertar la información en el propio sistema de almacenamiento.

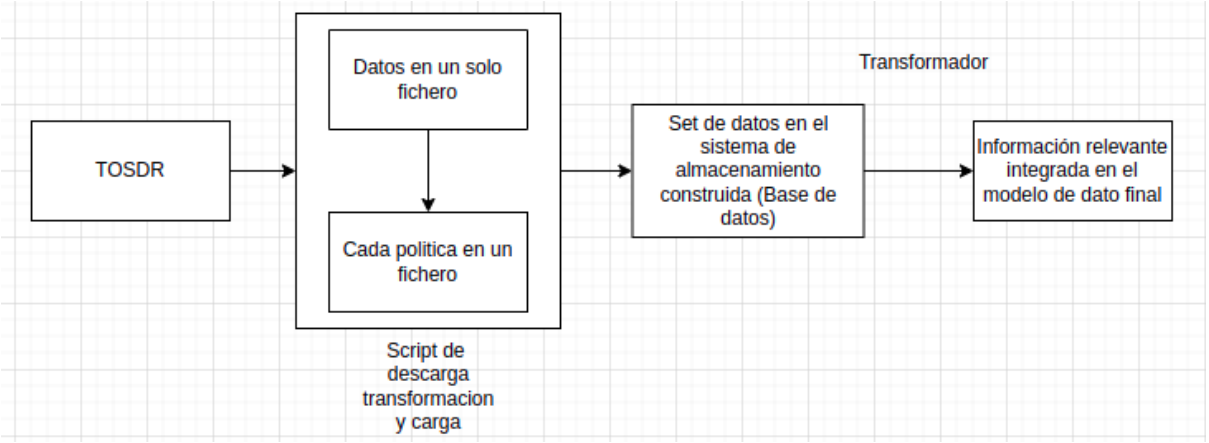

Imagen 14: Descripción de alto nivel del ETL para la fuente de datos TOSDR.

4.7. Transformador:

El transformador es el proceso que se ha construido en este proyecto para integrar la información desde las distintas fuentes de datos al modelo final creado. Para conseguir dicho objetivo, el trabajo se divide en 3 partes:

- Parte 1: Filtrar la información de las fuentes y quedarnos solo con la información relativa a las políticas de privacidad que aún no están en el sistema de almacenamiento.
- Parte 2: Comprobar si la compañía a insertar es nueva o no, si es nueva se inserta en la tabla ***mapping***.
- Parte 3: Insertar la información útil en la tabla ***companies*** y ***policies***, además de registrar los cambios en la tabla ***company_changes***.

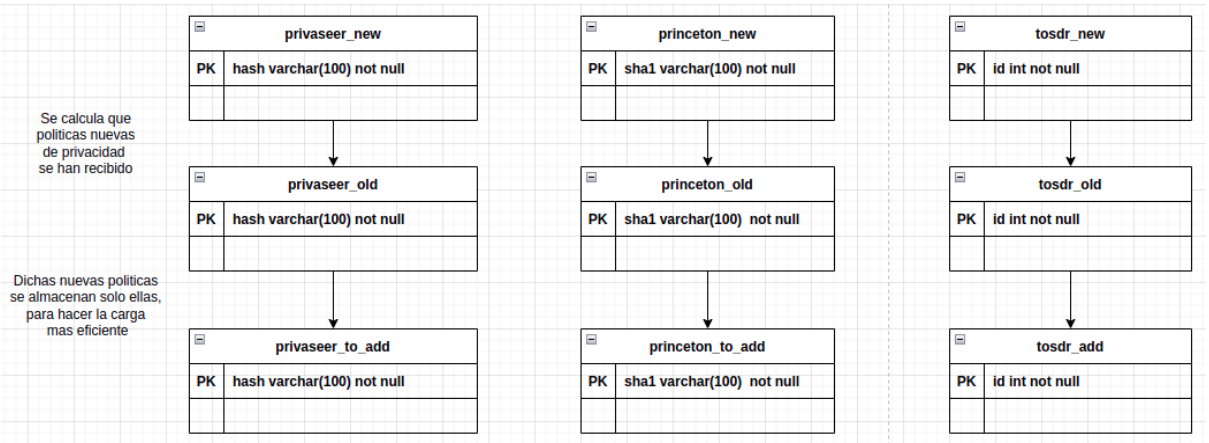

Imagen 15: Transformador Paso 1

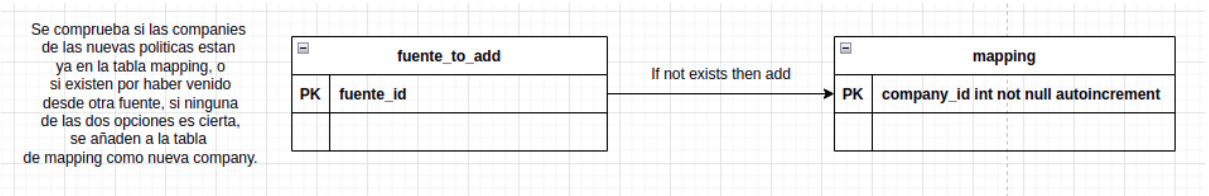

Imagen 16: Transformador Paso 2

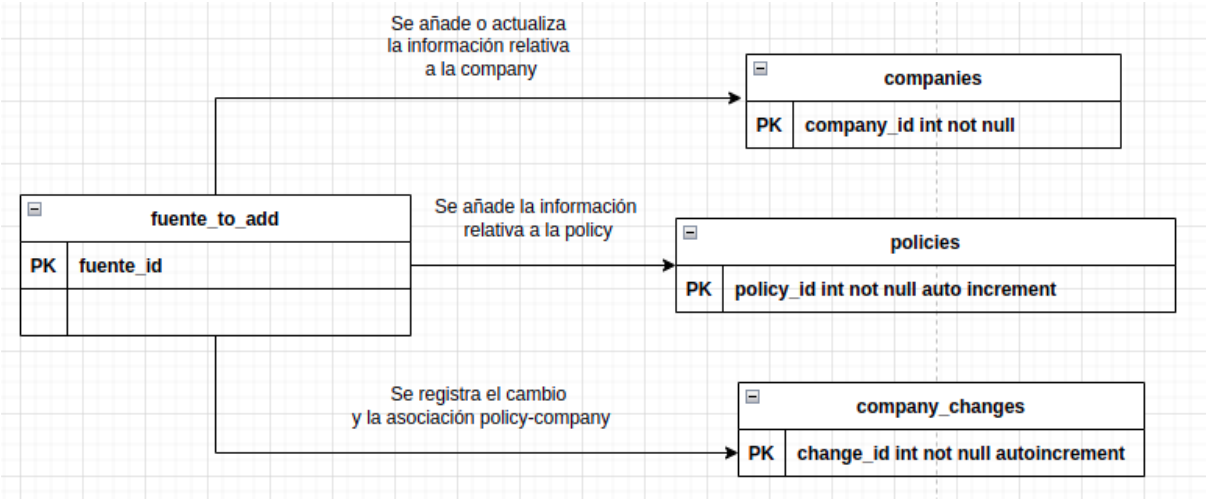

Imagen 17: Transformador Paso 3

Capítulo 5

Diseño de la solución

Para la solución se propone un sistema de extracción, carga y transformación de los datos recopilados de las distintas fuentes.

A diferencia del conocido sistema de ETL, donde primero se extrae el dato, luego se transforma y finalmente se carga en el sistema de almacenamiento seleccionado, en este caso, primero cargaremos los datos y las transformaciones se realizarán dentro del propio sistema de almacenamiento.

Los motivos de estos son varios, el primero es tener la información que hemos recopilado sin ninguna modificación, por si cualquier problema pudiera surgir en el modelo final, tener a mano si ese dato ha venido mal desde la fuente o se ha corrompido durante la transformación. Otro motivo es que es mucho más sencillo y escalable utilizar PL/SQL que cualquier otro tipo de aplicación que habría que desplegar, recompilar fuera del sistema de almacenamiento a la hora de realizar cualquier cambio.

5.1. ELT

5.1.1. Extract:

A la hora de extraer los datos, se utilizará una descarga manual para las dos fuentes de datos que no se actualizan ya que son operaciones que sólo se realizan una vez. Para la fuente TOSDR se ha construido un script de descarga donde obtenemos todo el conjunto de datos de la fuente en un fichero JSON. Dicho fichero se dividirá en un fichero por cada política, ya que es la forma en la que poder cargar cada una en una *row* distinta con las funciones de importación de MySQL para JSON.

Para conseguir dicho fichero se utilizan cuatro utilidades existentes en Linux, como son *curl, jq, sed y split.*

- *curl*[12]: curl es un comando que nos permite descargarnos todo el contenido dada una URL, esta es la forma en la que obtenemos el conjunto de datos de la fuente TOSDR, el output está en formato JSON.

[12] Documentación sobre el comando curl: https://man7.org/linux/man-pages/man1/curl.1.html

- *jq*[13]: jq, cuyas siglas significan JSON query, sirve para consultar datos que estén en formato JSON. En nuestro caso lo usaremos para formatear la salida del comando curl a un formato JSON indexado, que es más fácilmente legible por un humano.

- sed[14]: sirve para reemplazar cadenas de caracteres en un texto por una dada, permite usar expresiones regulares.

- split[15]: sirve para dividir un fichero, por ejemplo, cada cierto número de líneas.

5.1.2. Load:

Los datos se cargarán utilizando herramientas internas de MySQL que permiten cargar datos en formato CSV[16]/JSON[17] directamente en tablas. Los datos se cargarán con su formato original en tablas que se sobreescribirán con cada carga de datos de cada fuente que se haga, ya que los datos una vez cargados dentro del sistema de almacenamiento, se integrarán en el modelo de datos final.

5.1.3 Transform:

Las transformaciones necesarias para unificar los datos de las distintas fuentes de datos se realizarán dentro del sistema de almacenamiento, a través de distintos procesos PL/SQL con los cuales se integrarán los datos coleccionados de las distintas fuentes en el modelo de datos final.

Procedimientos de transformación de la información:

Se trata de un procedimiento PL/SQL que hará una carga de los datos desde las tablas donde se cargan los datos de las fuentes sin ninguna modificación a nuestra tablas del modelo de datos final.

Para esta tarea se construye un *manager*, encargado de ejecutar el proceso completo y escribir los *logs* correspondientes en una de las tablas creada para dicho propósito.

Este proceso también consta de una tabla donde se registran los procesos ejecutados.

---

[13] Documentación sobre jq: https://code.tools/man/1/jq
[14] Documentación sobre sed: https://linux.die.net/man/1/sed
[15] Documentación sobre split: https://man7.org/linux/man-pages/man1/split.1.html
[16] Documentación para cargar ficheros CSV en MySQL: https://dev.mysql.com/doc/refman/8.0/en/load-data.html.
[17] Documentación para cargar ficheros JSON en MySQL:https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-json-import-command.html

La razón por la que se ha decidido desarrollar un sistema así es que, dado que se va a trabajar con un gran volumen de datos—se trata de *datasets* de más de un millón de entidades y varios millones de *rows*—es necesario tener constancia de en qué punto del proceso ocurren errores y poder ejecutar los procesos de transformación de forma modular para poder reiniciarlos desde el punto de chequeo previo al último error. Estos son algunos casos con los que se ahorra tiempo a la hora de administrar una base de datos con dichos volúmenes.

Las tareas principales de la transformación serían:

- **PASO 1:** Identificar nuevas políticas en los conjuntos de datos, y ver si hay nuevas web/empresas asociadas a ellas.

- En caso de que no haya, se salta al paso 2. Se escribe un mensaje de *log* informando de que no hay nuevas compañías a añadir.

- **PASO 2:** Se comprueba si las nuevas compañías no existen en la tabla de *mapping* o si existen pero provenientes de otra fuente de datos.

- Si no existen se crea un nuevo registro en la tabla *mapping* para esa nueva compañía.

- Si existen pero provienen de otra fuente de datos, se actualiza ese registro con el ID de la fuente nueva.

- **PASO 3:** Se inserta la información relativa a las empresas en la tabla *companies.* Se escribe un mensaje para avisar de que la inserción fue correcta.

- Se registran los cambios en las compañías afectadas en la tabla *companies_changes.* Se escribe un mensaje para avisar que la inserción fue correcta.

- Se registra la información sobre las nuevas políticas en la tabla *policies*. Se escribe un mensaje para avisar que la inserción fue correcta.

- Si fallase en alguno de los pasos, no haría falta re-ejecutar todos de nuevo, de esta forma, la ejecución del proceso es más fácilmente arreglable pudiéndose centrar la atención en el caso de fallo.

- Para el tratamiento de excepciones, en MySQL existe un objeto llamado signal[18], es cual hace los efectos de una excepción en Oracle. Se define una signal para cada paso/posible punto de fallo, siendo diferente el mensaje de fallo, dependiendo de cuál sea el error y durante qué pasó se produzca.

- Para registrar la información sobre los procesos ejecutados en las tablas descritas en el modelo de datos del transformador, se han construido las tablas ***jobs*** y ***jobs_logging***. Las transacciones para escribir información en estas tablas deben ser pragmáticas y autónomas del resto del proceso, ya que si el resto del proceso falla, se ejecutará ***rollback*****,** y esto haría desaparecer la información de *logging* ya insertada en las tablas. Cómo solución se ha optado por construir dos funciones, una para cada tabla, donde se utilizará ***start transaction*****[19],** para evitar perder información de los *logs* en caso de que el proceso falle.

Problema de almacenar datos en formato HTML:

- Algunas de las fuentes de datos permiten la descarga de la propia política de privacidad completa, en formato HTML. Como las dos fuentes que lo permiten, son PrivaSeer y Princeton, las dos estáticas, se decide no cargar dicha información al no ser del todo actual, y no tener valor en este caso de uso, ya que se saca valor de la información de las fuentes, viendo como ha cambiado una web/compañía su política de privacidad a lo largo del tiempo, esta información es obtenible de los metadatos, sin necesidad de conocer exactamente qué cambio se hizo. Por ejemplo, interesa saber si el sentimiento del consumidor ha aumentado o disminuido después de un cambio en la política de privacidad, o ver cuantos cambios en los últimos años ha realizado una web/compañía, y esa información es obtenible de los metadatos sin necesidad del código HTML de la política de privacidad.

Creación de las secuencias/campos autoincrementales:

- Para registrar los cambios, las compañías en nuestro *mapping*, las políticas, los mensajes de un *job* y los identificadores de un *job*, se usan **campos autoincrementales**[20], ya que en MySQL no existen las secuencias.

[18] Docuentación sobre signal en MySQL: https://dev.mysql.com/doc/refman/8.0/en/signal.html
[19] Documentación sobre *start transaction* en MySQL: https://dev.mysql.com/doc/refman/8.0/en/commit.html
[20] Documentación sobre campo autoincrementales en MySQL:https://dev.mysql.com/doc/refman/8.0/en/example-auto-increment.html

- Dichos campos autoincrementales se usan para asignar un identificador único, de valor numérico, a cada nueva fila que se inserta en una tabla. En nuestro caso, esta condición se cumple para casi todas nuestras tablas excepto la tabla ***companies*** en la que los datos no representan cambios a lo largo del tiempo sino un estado actual, por lo que allí los datos se sobreescriben, no se insertan nuevos en cada ejecución, con la excepción de cuando se inserta una nueva compañía en el sistema, en ese caso si se inserta nueva información pero sigue sin representar una evolución temporal, y el identificador único se crea en la tabla ***mapping***.

5.2. Construcción del *mapping*:

Para construir la tabla de *mapping* se ha utilizado el siguiente proceso:

- Comprobar que el ID de la fuente no existe en la tabla *mapping*, en caso de existir, no se continúa con el proceso ya que la compañía ya estaría relacionada con un ID del sistema de almacenamiento.
- Extraer el nombre de la web a la que pertenece la política de privacidad.
- Utilizar expresiones regulares para guardar solamente el nombre de la web, obviando www y el dominio, por ejemplo.
- Si el nombre no se encuentra en la base de datos, se añade una nueva entrada (nueva compañía).
- Si el nombre ya se encuentra en la base de datos, se introduce el ID de la fuente en la fila correspondiente en la tabla *mapping*.

5.3 Problemas encontrados durante la integración de las datos:

A continuación se exponen que problemas técnicos se han encontrado desde la propia descarga de los datos y la solución técnica aplicada entre las distintas opciones posibles:

- Varios campos de la fuente TODSR tienen formato JSON, por lo que tenemos JSON anidados. Para resolver este problema, se plantearon dos posibles soluciones, una, transformar los datos de JSON a CSV y eliminar ese anidamiento, o intentar insertar los datos en el formato JSON, tal y cómo se recogen de la fuente. Tal y como se ha hecho durante el resto del proyecto, los datos no se han modificado fuera del sistema de almacenamiento, por lo que la opción de transformar a CSV es la opción más alejada de la realidad del proyecto. En MySQL, existe la opción de crear columnas cuyo tipo de dato sea JSON, tras una pruebas y comprobar que esto funciona correctamente, se decide cargar los datos sin modificar como para las otras fuentes, en el propio formato JSON dentro del sistema de almacenamiento.

- *Start-Transaction*: solución para la tabla de *logging* y *jobs.* Durante la construcción del *software* encargado de la integración de la información surgió el problema de ¿qué ocurre cuando el proceso falla y se hace *rollback*?¿Se pierde la información de la tabla ***logging*** y ***jobs***? Era necesario que las inserciones realizadas en dichas tablas no desaparecieran al ejecutar rollback cuando ocurriese un error durante el procedimiento, ya que perderían el sentido que tienen: ayudar a determinar qué ha fallado en caso de que algo falle. MySQL tiene la opción de crear una transacción autónoma dentro del propio proceso, la cual comienza con *start transaction* y termina una vez ejecutado *commit*; sin que este *commit* se refiera al resto de transacciones ejecutadas durante el proceso. La decisión que se ha tomado es definir dos funciones, una para escribir en ***jobs*** y otra para escribir en ***logging***, donde en cada una se creará una transacción que finalizará justo después de la inserción de la línea en la tabla.

Capítulo 6

Conclusiones y trabajo futuro

En el presente capítulo se detallan las conclusiones obtenidas tras la elaboración del proyecto y el posible trabajo futuro a realizar sobre él mismo.

Como conclusiones, cabe destacar la importancia de cómo resolver el problema de la heterogeneidad en las fuentes, o dicho más comúnmente, que el mismo objeto o entidad sea llamada o etiquetada de distinta forma dependiendo de cuál sea la fuente, de ahí la importancia que tiene el proceso de construir una tabla donde poder plasmar las relaciones que existen entre la etiqueta que recibe dicho objeto en las fuentes originales y en el sistema de almacenamiento final, para así poder relacionar el mismo objeto o entidad entre todas ellas. En este caso se utilizó una asociación por nombre.

También destacar la importancia del análisis de las fuentes y entender que representa a alto nivel cada uno de los campos de una fuente, de esta forma podremos saber qué campos nos son necesarios y cuáles no y como poder sacar valor de dichos datos.

Un buen análisis de las fuentes facilita mucho la inserción de los datos dentro del sistema de almacenamiento.

Como trabajo futuro, podría darse el caso de encontrar nuevas fuentes de datos con información sobre políticas de privacidad, en ese caso, bastaría con construir un sistema de inserción de los datos desde la fuente al sistema de almacenamiento, añadir dicha fuente al proceso de *mapping*, y añadir la funciones correspondientes al *software* encargado de la integración así como las columnas que fueran de interés a las tablas ***companies*** y ***policies*****.** Sin que exista la necesidad de alterar nada del código escrito durante este proyecto.

Así mismo, se podría, por ejemplo, querer ampliar la información sobre las empresas con datos distintos a las políticas de privacidad, por ejemplo, valor de la empresa en bolsa. Dicha información sería fácilmente integrable. Habría que definir un nuevo ETL para dicha fuente, una nueva tabla donde almacenar la información relativa a la bolsa para cada compañía a lo largo de un periodo de tiempo determinado, y las funciones pertinentes al *software* encargado de la integración.

Capítulo 7

Apéndices técnicos

En esta sección se recapitula todo el código utilizado para el desarrollo y despliegue de la aplicación, así como un manual para su uso.

7.1 Script de extracción:

URL: *https://drive.google.com/file/d/1kW68WDi_JvnBmywcMDlQrQQZKKWTiPGI/view?usp=sharing*

7.2 Procedimientos de transformación de la información:

URL: *https://drive.google.com/file/d/1oWYa4zJ9RCsQj_7DPGEFvcmJHicjZ_-R/view?usp=sharing*

7.3 Creación de tablas:

URL: *https://drive.google.com/file/d/1ASs9A_0drAn10UT6Ezt6f3KquQ4FvHI4/view?usp=sharing*

7.4 Manual de usuario:

El proceso de extracción es manual para dos fuentes de datos, y automático (un *script* ejecutable) para *TOSDR.* Para la carga de datos se puede utilizar cualquier método de MySQL, como cláusulas *LOAD FILE* o bien algún software como *MySQL Workbench*.

La transformación de los datos e integración de los mismos ocurre a través de la ejecución de un *procedure* en la base de datos, llamado *run_new_job*. A este hay que pasarle también el ID de la fuente para la que queremos ejecutar la transformación e integración.

Bibliografía

[1] Página web de la fuente de datos *PrivaSeer*. URL: *https://privaseer.ist.psu.edu/*

[2] Página web de la fuente utilizada por *PrivaSeer Common Crawl*. URL: *https://commoncrawl.org/*

[3] Página web de la fuente usada por *PrivaSeer*, *Free Company Dataset*. URL: *https://docs.peopledatalabs.com/docs/free-company-dataset*

[4] Página web de la fuente de datos *Princeton*. URL: *https://privacypolicies.cs.princeton.edu/*

[5] Página web de Alexa para el ranking de las mejores empresas *Alexa Top 100K.* URL: *https://www.alexa.com/*

[6] Página web de la fuente *Terms of Service, Didn't Read*. URL: *https://tosdr.org/*

[7] Documentación propia de *MySQL*. URL: *https://dev.mysql.com/doc/*

[8] Documentación propia de *Linux*. URLs:
- *https://linux.die.net/man/*
- *https://man7.org/*

[9] *Principles of Data Integration,* Anhai Doan, Alon Halevy and Zachary Ives. Ed. Morgan Kaupmann ISBN: 978-0-12-416044-6

Anexo 1: Imágenes y tablas