Trabajo Fin de Máster
Modalidad B

Enmascaramiento Dinámico de Datos

Alba Gude Santos

Máster en Técnicas Estadísticas

Curso 2022-2023

Propuesta de Trabajo Fin de Máster

Título en galego: Enmascaramento dinámico de datos
Título en español: Enmascaramiento dinámico de datos
English title: Dynamic Data Masking
Modalidad: Modalidad B
Autor/a: Alba Gude Santos, Universidad de Santiago de Compostela
Director/a: Guillermo López Taboada, Universidade da Coruña;
Tutor/a: Gabriel Rocamador Murillo, SDG Group;
Breve resumen del trabajo: En este trabajo trataremos y aplicaremos la técnica del enmascaramiento dinámico de datos. Este método tiene como principal objetivo ocultar campos de determinadas tablas que se consideran confidenciales. Además, permite mantener la estructura de los datos y, por tanto, podremos realizar acciones con ellos pero sin revelar nunca el dato original. Finalmente, se mostratá el caso que he tratado en la empresa de prácticas, SDG Group, que son datos de la compañía de infraestructuras y telecomunicaciones Cellnex.
Recomendaciones:
Otras observaciones:

Don/doña Guillermo López Taboada, Catedrático de la Universidade da Coruña, don/doña Gabriel Rocamador Murillo, Executive Manager de SDG Group, informan que el Trabajo Fin de Máster titulado

Enmascaramiento dinámico de datos

fue realizado bajo su dirección por doña Alba Gude Santos para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En Santiago de Compostela, a 05 de Junio de 2023.

LOPEZ TABOADA GUILLERMO - 44088774M

Firmado digitalmente por LOPEZ TABOADA GUILLERMO - 44088774M
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-44088774M, givenName=GUILLERMO, sn=LOPEZ TABOADA, cn=LOPEZ TABOADA GUILLERMO - 44088774M
Fecha: 2023.06.05 21:51:50 +02'00'

El/la director/a:
Don/doña Guillermo López Taboada

El/la director/a:
Don/doña

El/la tutor/a:
Don/doña Gabriel Rocamador Murillo

El/la tutor/a:
Don/doña

El/la autor/a:
Don/doña Alba Gude Santos

Declaración responsable. Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, Disposición 2978 del BOE núm. 48 de 2022), **el/la autor/a declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,...sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

Índice general

Resumen

Resumen en español

En la era digital actual, donde la recopilación y el análisis de grandes cantidades de datos se ha vuelto tan común, es muy importante garantizar que los datos confidenciales no sean expuestos a terceras personas no autorizadas.

En este trabajo explicaremos la técnica del **enmascaramiento dinámico de datos**, que implica la generación de datos sintéticos o transformados que mantienen las caracteristicas clave y estructura de los datos originales, pero sin la necesidad de relevar la información confidencial. Por otra parte, debido a la evolución del almacenamiento y la transmisión de la información, hablaremos de la explotación de los almacenes de datos, donde presentan especial relevancia los almacenes de datos en la nube, que son los más utilizados en la actualidad por las empresas y organizaciones, al permitirles guardar grandes cantidades de datos y el acceso a los mismos en cualquier lugar y momento si tienen una conexión estable a Internet. Además, hablaremos de los procesos ETL y del modelado de datos, donde se explican sus beneficios y los diferentes tipos de modelos o bases de datos que existen. Introduciremos también un capítulo de soluciones de enmascaramiento dinámico de datos, donde indicaremos las partes más relevantes del código en lenguaje SQL que hemos implementado para crear y aplicar las políticas de enmascaramiento a nuestros datos. Para terminar, aplicaremos dichas soluciones a una determinada base de datos de la compañía española de telecomunicaciones **Cellnex Telecom S.A**, logrando enmascarar los campos confidenciales requeridos de las tablas indicadas por la compañía.

Para finalizar el trabajo, mostraremos las conclusiones que hemos podido extraer al realizar este proyecto durante el período de prácticas en la empresa **SDG Group** sobre la importancia de la seguridad de los datos y las diferentes técnicas que existen para protegerlos.

English abstract

In today's digital age, where the collection and analysis of large amounts of data has become so commonplace, it is very important to ensure that sensitive data is not exposed to unauthorized third parties.

In this paper we will explain the technique of dynamic data masking, which involves the generation of synthetic or transformed data that maintains the key characteristics and structure of the original data, but without the need to relieve confidential information. On the other hand, due to the evolution of information storage and transmission, we will talk about the exploitation of data warehouses, where data warehouses in the cloud are particularly relevant, as they are the most used nowadays by companies and organizations, allowing them to store large amounts of data and access them anywhere and anytime if they have a stable Internet connection. In addition, we will talk about ETL processes and data modeling, explaining their benefits and the different types of models or databases that exist. We

will also introduce a chapter on dynamic data masking solutions, where we will indicate the most relevant parts of the SQL language code that we have implemented to create and apply masking policies to our data. Finally, we will apply these solutions to a specific database of the Spanish telecommunication company Cellnex Telecom S.A., masking the required confidential fields of the tables indicated by the company.

To finish the work, we will show the conclusions that we have been able to draw from this project during the internship in the company SDG Group on the importance of data security and the different techniques that exist to protect them.

Prefacio

En el mundo actual, los datos se han transformado en uno de los activos más valiosos para las empresas y organizaciones. Como consecuencia de su valor, existe la necesidad de proteger la confidencialidad de estos datos.

Para ello, el enmascaramiento de datos es un método fundamental que se utiliza hoy en día para proteger los datos que se consideran privados. Esta técnica consiste en la sustitución de la información original sensible por datos anonimizados, pero manteniendo siempre la utilidad de esta información para el análisis que deseemos llevar a cabo.

Las principales causas que nos motivan a realizar un trabajo sobre el enmascaramiento de datos se basan en los siguientes aspectos clave: privacidad, seguridad, investigación, análisis y utilización de la información.

Con el aumento de información e intercambio de datos de carácter personal, lo fundamental es garantizar la privacidad de los usuarios. Las reglas y normas de protección de datos establecen unos requisitos que debemos cumplir para tratar de forma segura la información. En cuanto a esto, el enmascaramiento de datos cumple con estos requisitos ya que reduce el riesgo de acceso no autorizado a la información sensible.

Por otra parte, la pérdida o acceso no autorizado a información condifencial puede tener grandes consecuencias, entre las que podemos destacar el fraude financiero o el robo de identidad. En este caso, lo que hace el enmascaramiento de datos es reducir la exposición de datos privados, minimizando así estos riesgos.

En muchas ocasiones necesitamos compartir estos datos confidenciales para realizar investigaciones conjuntas. El papel que juega aquí esta técnica es la facilidad de mantener la confidencialidad de la información durante el intercambio de estos datos sensibles.

En cuanto a su utilización y análisis, el enmascaramiento de datos nos permite realizar estas dos acciones ya que ofrece métodos que mantienen la estructura de los datos y las posibles relaciones entre ellos. De esta forma se asegura la privacidad de los individuos.

En resumen, la técnica del enmascaramiento de datos se ha convertido en una necesidad hoy en día, debido a que la privacidad de la información es una prioridad fundamental. En este trabajo nos centraremos principalmente en este tema, introduciendo también algunos conceptos previos que nos ayuden a entender mejor la funcionalidad del enmascaramiento de datos y de los almacenes de datos en la nube.

Para entenderlo mejor, terminaremos con un ejemplo práctico que ha sido realizado en mi período de prácticas en la empresa SDG Group.

Capítulo 1

Explotación de Almacenes de Datos

Antiguamente la información que recibíamos se guardaba en revistas, periódicos, libretas, carpetas, discos, etc. Hoy en día, debido a la gran cantidad de información que recibimos cada día, esta forma de mantener y guardar la información sería ineficaz ya que perderíamos algunos datos, sería difícil tener un lugar para almacenar la información con el paso del tiempo, nos arriesgamos a que alguien los robe o tenga acceso a ellos, entre otras consecuencias que podría tener guardarla de esta forma. También es diferente la manera en la que se comparte la información a día de hoy

Es por eso que, con la llegada de Internet, ha surgido la necesidad de tener cada vez más espacio para los datos y también mayor seguridad. Por eso empezamos hablando de lo que son los **almacenes de datos** y qué utilidad tienen hoy en día.

En el ámbito de la informática, se define un **almacén de datos** o **data warehouse** (DW) como un conjunto de datos que está orientado a una determinada empresa u organización y que se caracteriza por su **no volatilidad**, **integridad** y **variabilidad en el tiempo** [1].

Los almacenes de datos son utilizados para guardar grandes cantidades de información y tomar una determinada decisión en el ámbito en el que se utiliza. Estos datos pueden provenir de múltiples flujos de datos, Internet, bases de datos relacionales y sistemas de información.

Actualmente, la **información digital** aparece en todas partes en nuestra sociedad, fluye a través de las redes y finalmente es consumida por los usuarios a través de pantallas, radios y teléfonos.

Dado que los **almacenes de datos locales** tienen una capacidad de almacenamiento inflexible, dificultades técnicas y altos gastos operativos debido a las necesidades de mantenimiento del hardware, muchas empresas están trasladando su almacenamiento de datos a la nube.

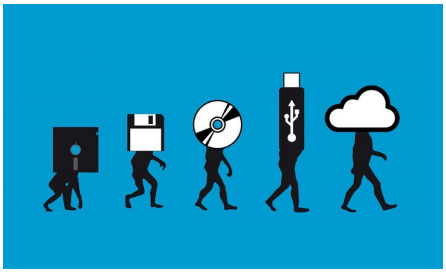


Figura 1.1: Evolución del almacenamiento de datos

1.1. Almacén de datos en la nube

Como hemos comentado, los almacenes de datos locales tienen la desventaja de que su capacidad es limitada. Es por esta razón que el almacenamiento físico ha sido reemplazado por los **servicios web** de almacenamiento de datos.

Esta nueva arquitectura de almacenamiento de la información recibe el nombre de **"computación en la nube"** y los datos y aplicaciones con las que trabaja se distribuyen a través de Internet a diferentes sistemas en la nube que pueden pertenecer a Google, Microsoft, Oracle o Amazon, entre otros.

La computación en la nube o **Cloud computing** se ha hecho popular a partir de 2008 y desde entonces ha evolucionado para llegar a una amplia audiencia que necesita este tipo de arquitectura para procesar grandes cantidades de datos [2].

Empezamos hablando, por ejemplo, del **almacenamiento**, uno de los servicios más populares a la hora de consumir servicios en la nube, y muchas empresas recurren al alquiler de espacio de almacenamiento en grandes centros de datos porque no tienen el espacio necesario para trabajar con toda la información que necesitan.

Es por esta razón que a las empresas y organizaciones les beneficia invertir en las nubes de servidores, ya que de esta manera logran trasladar partes importantes de sus operaciones a la nube.

Antes de que aparecieran estos servidores, las empresas necesitaban mucha memoria, software complejo y mucha potencia informática. En general, utilizar un almacén de datos en la nube u online ofrece grandes beneficios, entre los que destacan costes, evitar problemas de mantenimiento y la continua necesidad de mejorar las infraestructuras.

A continuación, presentamos una serie de ventajas específicas del *cloud computing* que justifican los motivos de que se esté convirtiendo en una de las mejores opciones para particulares y empresas:

- Menor coste, debido a que el coste del almacenamiento online es menor que los gastos que conlleva el almacenamiento físico. El coste variará en función de la capacidad de almacenamiento que necesitemos.
- Facilidad de acceso a los datos y su intercambio, ya que teniendo conexión a Internet podemos acceder de forma sencilla a ellos. La desventaja es la calidad o velocidad que tengamos en nuestra red en ese momento.
- Mayor seguridad, debido a que la mayor parte de estas plataformas tienen funciones de seguridad integradas. Además, se puede restringir a determinados usuarios para que no puedan acceder a la información.
- No ocupa espacio físico. Esto es especialmente importante en el caso de las empresas.
- Sincronización y automatización.
- Mayor protección de los datos. Existen nubes privadas y públicas.
- Un almacén de datos promueve el uso eficiente de la información mediante la eliminación de controles entre industrias que impiden el flujo de información.

En resumen, las plataformas actuales de almacenamiento de datos en la nube permiten a las empresas y organizaciones ventajas económicas, así como desarrollar el análisis de datos en un entorno seguro para mejorar y facilitar la toma de decisiones y las estrategias que llevarán a cabo para lograr sus objetivos.

Los almacenes de datos basados en la nube aprovechan los principales beneficios asociados con la informática bajo demanda, entre los que destacan:

- El acceso de **usuarios de gran alcance**.
- El **almacenamiento "ilimitado"**.
- Mayor **capacidad informática**.
- Opción de **pagar** solamente por el **espacio que se utiliza**.

Los almacenes de datos más conocidos en la actualidad son Amazon Redshift, Microsoft Azure y SnowflakeDB, entre otros.

Figura 1.2: Almacenamiento de datos en la nube

Cuando trabajamos con almacenes de datos en la nube, las herramientas de integración de datos convierten a estos en información útil que, posteriormente, será procesada.

Cuánto más conocido y necesario se hace el almacenamiento web en nuestros objetivos del día a día, más importante es saber elegir un procesador que sea lo más potente posible, pero que a su vez sea económico, ya que pagaremos por la cantidad de espacio utilizado.

A pesar de todas las innovaciones informáticas que surgieron en la primera década del sigo veintiuno, ha sido el almacenamiento en la web el que ha provocado más impacto social y económico.

Como podemos observar, las Tecnologías de la Información y las Comunicaciones (TIC) están en continuo cambio y se transformarán en una industria global que provocará importantes cambios en el trabajo de las empresas y de las personas, así como en nuestra vida cotidiana.

1.1.1. Estado de la computación en la nube

La introducción de Internet en nuestra vida diaria requirió ampliar el espacio de almacenamiento. Las computadoras son utilizadas por los ISP (Internet Service Providers) como plataformas de hardware.

Para aumentar la flexibilidad de estas computadoras, actualmente se están desarrollando varias tecnologías de software. Esto resultó en el desarrollo de tres modelos principales de computación en la nube basados en metodologías de extracción de recursos subyacentes [6].

- **Amazon Cloud**: la computación en la nube de Amazon se basa en la tecnología de virtualización de servidores. Este modelo de almacenamiento en la nube se lanzó en 2006 con Elastic Compute Cloud (EC2) basado en Xen, el servicio de almacenamiento de objetos (S3) y el servicio de almacenamiento de datos estructurados (SimpleDB). En 2007, lanzó Amazon Web Services (AWS), un importante precursor de la infraestructura como servicio (IaaS) de bajo costo.

  Este modelo ofrece almacenamiento, potencia de proceso de computación y sistemas de gestión de base de datos, entre otros servicios, para descargar y ejecutar los datos y a los que accedemos mediante internet.

  El primer servicio proporcionado por Amazon como computación en la nube fue S3, permitiendo un almacenamiento de datos ilimitado. El siguiente fue EC2 con una mejor capacidad de proceso y que permitía elegir a los clientes diferentes configuraciones del servidor. Con el tiempo, además de una gran empresa de comercio, es también una gran empresa tecnológica especializada en tecnología de la información.

- **Google Cloud**: este modelo está basado en la creación de pruebas especiales de tecnología. Entre 2003 y 2006, Google anunció varios proyectos de investigación relacionados con la computación en nube de plataforma como servicio (PaaS). En 2008 se lanzó la plataforma Google App Engine (GAE).

  A mediados de 2008 presentó Street view, un servicio de mapas en línea, que ofrece información sobre mapas de ciudades, así como de sus calles y plazas.

- **Microsoft Azure**: lanzado en octubre de 2008 y utiliza Windows Azure Hypervisor (WAH) como infraestructura de nube subyacente y .NET como contenedor de aplicaciones. También proporciona objetos BLOB y SQL.

Acabamos de hacer un resumen de los tres proveedores principales, así reconocidos los tres como líderes del último cuadrante de Gartner sobre proveedores de servicios en la nube, pudiendo observar como la virtualización de los servidores es flexible y compatible con el software y las aplicaciones. Es por eso que la computación en la nube es uno de los métodos de minería de datos más utilizado en la actualidad.

Además de estos servicios de nuble pública, muchas organizaciones y empresas han implementado sistemas de computación en la nube patentados o propios. Podemos ver como la computación en la nube se convierte en una importante estrategia para los proveedores de servicios de tecnología.

1.2. Desventajas del Cloud Computing

Así como el Cloud Computing tiene muchas ventajas, también presenta algunos inconvenientes que presentamos a continuación:

- Es importante tener la mayor seguridad posible ante corrupción de datos y amenazas externas.
- Aumenta el peligro ya que al almacenar los datos en la nube, estos pueden residir en cualquier centro de datos.
- Se necesita estudiar la compatibilidd de software bajo licencia con el software en la nube.
- Interoperabilidad: necesitamos que esté garantizada esta característica entre todos los servicios, lo que lo hace menos seguro.
- SLA (Servile Level Agreement): se requiere el cumplimiento de tratos a nivel de servicio antes de confiar a una empresa las aplicaciones de la misma. Posteriormente, se deben cumplir los tiempos especificados para cada tarea.
- Las aplicaciones que utiliza el cloud computing deben estar diseñadas para que se puedan dividir en diferentes servidores.

1.3. Snowflake como almacén de datos en la nube

Hoy en día, las plataformas públicas en la nube ofrecen una computación virtual totalmente ilimitada y recursos de almacenamiento bajo las necesidades de cada persona o empresa. Además, el modelo de software como servicio (SaaS) ofrece soluciones a sistemas empresariales y usuarios independientes que antes no podían permitirse debido al coste y complejidad.

Los sistemas de almacenamiento de datos han sido diseñados para recursos fijos y no pueden aprovecharse de la elasticidad de la nube. Además, su dependencia de complejos procesos ETL y el ajuste físico están en desacuerdo con los requisitos de flexibilidad de los nuevos tipos de la nube de datos semiestructurados y cargas de trabajo en rápida evolución.

Podemos introducir el almacén de datos en la nube Snowflake como un sistema multiusuario, seguro, transaccional, escalable y elástico con soporte SQL y extensiones integradas para datos semiestructurados y sin un esquema determinado.

En nuestro caso particular, este almacén se presenta como un servicio de pago por uso en la nube del proveedor **Microsoft Azure**. Esto permite que los usuarios carguen sus datos en la nube y puedan consultarlos de manera inmediata y desde cualquier parte del mundo utilizando determinadas herramientas que serán necesarias.

Snowflake comenzó a implementarse a finales del año 2012, pero no fue hasta 2015 cuando estuvo disponible para su uso. A partir de ahí, esta plataforma es utilizada cada vez por más empresas, tanto grandes como pequeñas, ya que es una de las mejores actualmente que utiliza los almacenes en la nube. Durante este capítulo, explicaremos el diseño de Snowflake y su arquitectura multiclúster de datos compartidos.

Para ello, terminamos la introducción de la sección con una definición que será útil:

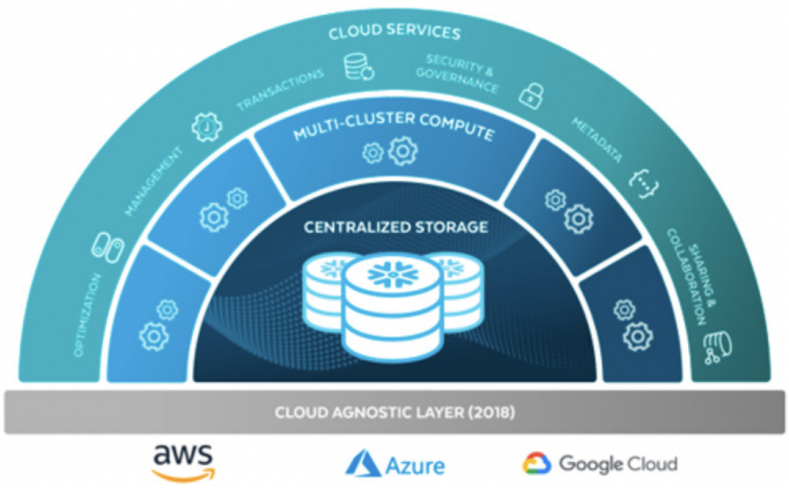
Definición 1.1 *Un **clúster** es un sistema que agrupa o relaciona una o más computadoras para que puedan trabajar de manera unificada y poder conseguir el objetivo propuesto de la forma más eficiente posible. Estas máquinas comparten entre sí las tareas de procesamiento y las ejecutan de manera simultánea.*

Es por esta razón que diremos que Snowflake es una plataforma multiclúster.

1.3.1. Arquitectura

La plataforma Snowflake ha sido diseñada para ser un servicio completo para una determinada empresa u organización, ofreciendo una alta usabilidad e interoperabilidad y también una alta disponibilidad.

Su principal ventaja es la fácil accesibilidad a la información y la posibilidad de trabajar a la vez desde diferentes ordenadores o máquinas. Además, la misma información puede ser utilizada y modificada por varios usuarios a la vez, aunque se encuentren en lugares diferentes.



Además, la plataforma Snowflake es una arquitectura que dispone de ciertos servicios que son tolerantes a determinados fallos y escalables de manera independiente. Estos servicios de los que hablamos se forman por tres capas diferentes, donde la primera de ellas puede variar según el almacén de datos en la nube que elija la empresa [3]:

1. **Almacén de datos**: en nuestro caso práctico, la plataforma hace uso de uno de los principales proveedores, **Microsoft Azure**, para almacenar la información necesaria para llevar a cabo el proyecto.
2. **Almacenes virtuales**: son las máquinas virtuales donde se ejecutan las consultas necesarias dentro de los clústeres elásticos.
3. **Servicios en la nube**: es considerado el cerebro del sistema y está constituído por un conjunto de servicios que gestionan almacenes virtuales, consultas y todos los metadatos que les rodean, como esquemas de bases de datos, claves de cirado, estadísticas de uso, etc.

A diferencia de los almacenes virtuales, que son recursos específicos de cada usuario, la capa de **servicios en la nube** es multiusuario, es decir, la pueden compartir a la vez y desde distintas ubicaciones diferentes usuarios de una determinada organización o empresa.

Además, en la capa de servicios en la nube aparecen diferentes servicios como el **control de acceso**, la **optimización de consultas** y la **administración de transacciones**, entre otros.
Una de las ventajas de la plataforma es que cada uno de estos servicios es compartido a través de muchos usuarios, lo que hace más eficiente la utilización del servicio en cuestión y de esta forma conseguimos reducir los gastos administrativos.

Para obtener una alta escalabilidad y una alta disponibilidad, se replica cada uno de los servicios. Esto tiene la ventaja de que si falla alguno de los nodos de algún servicio, no provoca la pérdida de datos o la pérdida de disponibilidad de los mismos.

A continuación, hablaremos un poco sobre la gestión y optimización de consultas en estos servicios en la nube, ya que todas ellas pasan por esta capa.
En este lugar podemos realizar: el análisis de datos, la resolución de objetos, crear un plan de mejora y el controlar el acceso de usuarios.

También cabe mencionar que, debido a que Snowflake no utiliza índices, la cantidad de almacenamiento requerido para guardar la información necesarias es más reducido que la de otros sistemas.

Todo este diseño de la plataforma hace que sea más difícil que el optimizador tome una mala decisión, lo que hace que sea más robusto aunque nos arriesguemos a una pequeña pérdida del rendimiento máximo.

Una vez se haya completado el optimizador, el plan de ejecución se reparte entre los diferentes nodos trabajadores que forman la consulta que estamos ejecutando. Mientras esta se va ejecutando, los servicios en la nube llevan a cabo un seguimiento continuo del estado de la consulta o query para obtener información sobre los contadores de rendimiento.

Control de concurrencia:

La finalidad del **control de concurrencia** es comprobar la consistencia de nuestros datos cuando estamos ejecutando consultas. También nos sirve para asegurarnos de que cada acción se completa en un período de tiempo finito.

Como hemos mencionado anteriormente, el control de concurrencia se realiza en la capa de servicios en la nube. Cabe destacar que Snowflake es una plataforma que soporta cargas de trabajo analíticas que normalmente requieren grandes lecturas o inserciones y actualizaciones masivas que, además, pueden ser lentas.

Lo que haremos en este caso es llevar a cabo un **MVCC** (**Multiversion concurrency control**), o **control de concurrencia mediante versiones múltiples**, para que cada copia de un objeto modificado de nuestra base de datos se conserva durante un determinado tiempo.

Definición 1.2 *El control de concurrencia mediante versiones múltiples (**MVCC**) es una técnica que permite controlar el acceso, normalmente, utilizado por un gestor de una base de datos para proporcionar un acceso concurrente a los datos o implementar concurrencia [3].*

En lo que se refiere a una base de datos, este método consiste en implementar las actualizaciones de los datos sin borrar los datos antiguos, marcando los antiguos como obsoletos y añadiendo los nuevos datos obtenidos. De esta forma, existirán varias versiones de los datos, donde la válida es la más reciente.

Lo que se consigue con esto es que el sistema gestor de la base de datos no tiene que rellenar huecos en memoria recorriendola para eliminar los objetos que ya no se utilizan.

Además de esto, la lectura de transacciones en MVCC usan normalmente un ID o índice de transacción para saber qué estado de la base de datos leer. De esta forma, las lecturas se pueden independizar de las escrituras deseadas de la conservación de versiones antiguas, evitando procesos de bloqueo o exclusión de información. Con esto, las escrituras afectan a versiones futuras pero el ID de transacción garantiza la consistencia de las mismas. Con otras palabras, el MVCC proporciona a cada usuario conectado a la base de datos una instantánea de la misma para él solo. Por último, comentar que cualquier cambio que se haga no será visible a los demás usuarios hasta que la transacción se haya completado.

1.3.2. Características de Snowflake

Como hemos visto hasta ahora, Snowflake cuenta con ciertas características propias de un almacén de datos relacional, como puede ser un soporte completo de SQL, transacciones ACID, interfaces estándar, estabilidad y seguridad, rendimiento y escalabilidad.

A continuación, presentaremos ciertas características que hacen que esta plataforma sea una de las más utilizadas por las empresas actualmente [3]:

Experiencia pura de software como servicio

Snowflake admite interfaces de bases de datos estándar y funciona con varias herramientas y servicios de terceros, as herramientas de visualización de datos, y explotación de Business Intelligence, Tableau, Informatica o Looker.

Además, esta plataforma nos permite interactuar con el sistema a través de un navegador web.

La interfaz de usuario web hace muy fáil el acceso a Snowflake desde cualquier lugar y a cualquier hora. También permite a muchos usuarios guardar sus datos en el almacén de datos en la nube para poder consultarlos a distancia sin la necedidad de descargar un software.

Además de permitir operaciones SQL, la interfaz del usuario tiene acceso al catálogo de la base de datos, a la administración del sistema, a la información de uso, entre otras cosas.

Disponibilidad continua

Debido a que el análisis de datos se volvió muy relevante para tareas comerciales y empresariales, un requisito imprescindible que nos ofrece Snowflake es la disponibilidad continua e instantánea de nuestros datos.

Esta disponibilidad que ofrece Snowflake cumple con las dos carácterísticas técnicas principales que son la resistencia a determinados fallos y las actualizaciones en línea.

- Resistencia a fallos.

  Snowflake tolera fallos de nodos individuales y correlacionados en cualquier nivel de su arquitectura. Hoy en día, la capa de almacenamiento de datos de Snowflake replica los objetos en unos determinados centros de datos que se conocen como “zonas de disponibilidad”.

  Esto es una gran ventaja ya que si un nodo falla, el resto de nodos puedan continuar con las actividades correspondientes sin causar un gran impacto sobre los usuarios finales.

  El resto de servicios de la capa de servicios en la nube están formados por nodos sin estados en diferentes zonas de disponibilidad con una carga equilibrada que reparte las solicitudes de los usuarios entre estos nodos. De ahí que el fallo en un nodo no causa un gran impacto sobre el resto del sistema. Si un determinado nodo falla, el usuario se redirige a otro nodo diferente para realizar la siguiente consulta.

  Por otra parte, los Almacenes Virtuales no se distribuyen a través de AZ (Availability Zone) debido a motivos de rendimiento. En este caso, si uno de los nodos falla mientras se ejecuta la consulta, la consulta falla y se vuelve a ejecutar de forma transparente, pudiendo ser con el nodo reemplazado de manera inmediata o con un número reducido de nodos temporales.

  Para agilizar el proceso de reemplazamiento de nodos, Snowflake tiene un grupo de nodos ya destinados a eso, mientras que en un Almacén Virtual necesitaría ser reaprovisionado activamente por un usuario en una AZ diferente.

- Actualización en línea.

  Snowflake permite una disponibilidad continua cuando se producen fallos, pero también durante las actualizaciones de software, debido a que su sistema permite multitud de versiones de los distintos servicios que se ejecutan en paralelo, tanto Cloud Services, componentes y almacenes virtuales.

  Esto es posible debido a que los estados se mantienen en un almacén de clave-valor transaccional y se puede acceder a él a través de una capa de mapeo que se encarga de los metadatos, del control de versiones y de la evolución de los esquemas.

  Cuando realiza una actualización del software, Snowflake implementa una versión nueva del servicio junto con la anterior. Después, las cuentas de los usuarios se pasan de manera progresivamente a la nueva versión, así como las consultas realizadas por cada uno de los usuarios.

  Cuando todas las consultas y usuarios han terminado de usar la versión anterior, todos los servicios de esa versión son terminados y dados de baja.

  Ambas versiones de Cloud Services comparten el mismo almacén de metadatos. Además, los almacenes de datos de diferentes versiones pueden compartir los mismos nodos de trabajo y sus respectivos espacios de almacenamiento.

  Lo que ha provocado la actualización en línea es aumentar la velocidad de desarrollo a la hora de manejar y solucionar problemas en Snowflake.

Datos semiestructurados y sin esquema.

Snowflake considera tres tipos de datos estructurados que son: **VARIANT**, **ARRAY** y **OBJET**. Ofrecemos ahora una breve descripción de cada uno de ellos:

Los valores o datos que son de tipo VARIANT pueden almacenar cualquier valor de tipo SQL, ya sea date, varchar, etc. Además, su longitud es variable. Sin embargo, los datos de tipo ARRAY u OBJECT son mapas de cadenas asociados a valores de tipo VARIANT.

Dicho de otra forma, los elementos ARRAY y OBJECT son solo restricciones de tipo VARIANT.

Aún así, la representación interna de estos tres tipos de datos estructurados es la misma, ya que es una autodescripción, con una serialización binaria compacta que admite un valor clave de búsqueda rápida. Introducimos ahora una breve definición de las técnicas hash [4]:

Definición 1.3 *Las **técnicas hash** se definen o reconocen como métodos que fueron creados mediante una ecuación para generar posiciones en una tabla que será la que contendrá los datos que nos interesan para realizar el análisis.*

Continuando con lo anterior, entendemos entonces que las columnas VARIANT se pueden utilizar como claves de agrupación, claves de combinación y claves de orden. Además, este tipo de datos permite a Snowflake utilizar una versión **ELT** (**extract, load and transformate**) en vez del método tradicional, que es el modo ETL (extract, transformate and load).

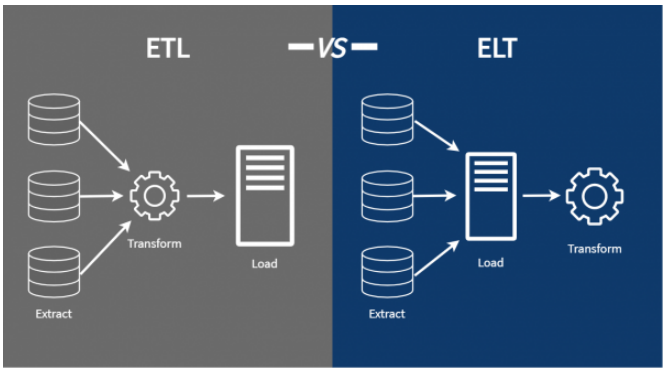


Figura 1.3: Modelo ETL y modelo ELT

Una de las mayores ventajas de Snowflake es que no necesitamos especificar un esquema de documentos ni llevar a cabo diferentes transformaciones en la carga, ya que los usuarios pueden cargar los datos de entrada, desde diferentes formatos, directamente en una columna de tipo VARIANT. Esto se debe a que la plataforma Snowflake presenta la capacidad de manejar el anaálisis y la inferencia de tipos de datos.

Este enfoque permite la evolución del esquema, ya que desvincula a los productores de información de los consumidores de información y cualquier intermediario. Por otra parte, cualquier cambio que se realice en los esquemas de datos en un sistema ETL necesita la coordinación entre varios departamentos de una organización.

Otra ventaja del método ELT y Snowflake es que, si se desea transformar los esquemas de datos, se pueden transformar utilizando toda la potencia de una base de datos SQL paralela, incluídas algunas operciones que normalmente faltan o son ineficientes en las cadenas ETL convencionales.

Además, Snowflake cuenta con funciones de procedimiento definidas por el usuario, que denotaremos por UDF, con sintaxis JavaScript e integración con los datos de tipo VARIANT.

El soporte para los procedimientos definidos por el usuario permite aumentar el número de tareas ETL que se pueden insertar en Snowflake.

- Operaciones post-relacionales.

  La principal operación en los documentos es la **extracción** de los datos, ya sea por nombre de campo, para objetos, o por desplazamiento, para array. En cuanto a esto, Snowflake permite extraer los datos tanto en lenguaje SQL funcional como en lenguaje JavaScript. Su codificación interna permite que la extracción sea muy eficiente.

A menudo, se realiza la extracción mediante una conversión del valor VARIANT resultante a un SQL de tipo estándar. De nuevo, debido a la codificación interna hace que este proceso resulte muy eficiente.

Además de la extracción, existe otra operación muy importante, que es el aplanamiento, que es la acción de dividir un documento anidado en varias filas. Para ello, Snowflake utiliza vistas laterales SQL para representar estas operaciones. Este método de aplanamiento puede ser recursivo, permitiendo la conversión completa de la estructura jerárquica del documento en una tabla relacional de procesamiento SQL.

- Almacenamiento y procesamiento en columnas.

  El uso de la representación binaria para datos semiestructurados permite integrar esta información en bases de datos relacionales. Lamentablemente, la representación por filas hace que el almacenamiento y procesamiento de dichos datos sea menos eficiente que el de los datos relacionales en columnas, que es la principal razón para transformar datos semiestructurados en datos relacionales.

  Para lograr tanto la flexibilidad de un sistema sin esquema y la mejora del rendimiento, Snowflake introduce un nuevo enfoque para el almacenamiento en columnas, es decir, Snowflake almacena datos en un formato columnar híbrido.

  Al almacenar datos semiestructurados, el sistema realiza automáticamente un análisis estadístico para. Posteriormente, las columnas se eliminan de los documentos y se almacenan por separado, utilizando el mismo formato de columnas comprimido que los datos relacionales originales.

  Durante un escaneo de los datos, las columnas se vuelven a juntar en una columna de tipo VARIANT. Aun así, la mayor parte de las consultas solamente afectan a un subconjunto de todas las columnas del documento original. Lo que hace Snowflake en estas situaciones es proyectar las empresiones en el operador de escaneo de manera que solamente se puede acceder a las columnas necesarias .

  Las mejoras descritas anteriormente se realizan de forma independiente para cada archivo de tabla, lo que permite un almacenamiento y extracción eficientes incluso bajo la evolución del esquema.

- Conversión optimista.

  Debido a que algunos de los campos o valores re representan en formatos externos, como JSON o XML, es necesario convertir dichos valores en su tipo real en lenguaje SQL ya sea mientras se insertan o actualizan los datos, o durante las consultas.

  Otro problema con los datos que no tienen un tipo determinado es la falta de metadatos adecuados para la poda, que es especialmente importante en el caso de las fechas.

  En cuanto a esto, Snowflake resuelve el problema realizando una conversión de datos optimista, y preservando tanto el resultado de la conversión como el cadena original en columnas separadas.

- Rendimiento de ejecución.

  Hasta ahora hemos hablado de los efecto del almacenamiento en columnas, la conversión optimista y la poda de datos semiestructurados. Ahora evaluaremos estos efectos sobre el rendimiento de las consultas.

  Lo que hace esta plataforma es crear dos tipos de esquemas de datos, primero un esquema relacional convencional y luego un modelo de datos "sin esquema", donde cada tabla estaba formada por una única columna de tipo VARIANT. A continuación, se forman grupos con los datos, se almacenan estos grupos que vienen en formatos externos sin formato y se cargan esos datos en Snowflake.

  Al ejecutar una consulta, los errores estándaros son insignificantes y se eliminan del resultado.

  En resumen, el rendimiento de las consultas sobre datos semiestructurados con esquemas relativamente estables y simples , es casi igual al rendimiento de los datos relacionales convencionales, disfrutando de todos los beneficios del almacenamiento en columnas, ejecución columnar y poda de una forma automática, sin intervención manual.

Capítulo 2

Técnicas de Enmascaramiento de datos

A lo largo del tiempo, la forma en la que se han utilizado los datos y la privacidad de los mismos han supuesto un problema para la seguridad de estos, el cual se ha solucionado haciendo uso de diferentes técnicas de enmascaramiento muy específicas para proteger los datos que se consideran más sensibles o privados.

Empezamos introduciendo la definición de datos sensibles [5]:

Definición 2.1 *Nos referiremos a los* ***datos sensibles*** *como aquellos datos que afectan a lo más propio de la persona, a su intimidad.*

Podemos pensar que los datos sensibles son aquellos datos que, al divulgarse o compartirse, pueden desencadenar una situación de vulnerabilidad a la persona afectada en su ámbito social, personal, de trabajo, familiar,etc.

Dependiendo del tipo de dato del que se trate, distinguiremos dos grupos diferentes dentro de los datos sensibles. Por ejemplo, diremos que serían datos **sensibles de tipo personal** algunos datos como la identificación, el nombre, el teléfono, el email. En otro caso, llamaremos **datos sensibles financieros** a un número de cuenta o tarjeta, al saldo, etc.

Las medidas de enmascaramiento de las que hablaremos a lo largo de este trabajo se encuentran agrupadas e implementadas en complejas plataformas de seguridad de datos.

En este trabajo podremos ver, entre otras cosas, como ha ido evolucionando el mercado de las medidas de enmascaramiento de datos y cuáles son las más utilizadas y las que más destacan en el mundo tecnológico hoy en día.

Lo que intentaremos explicar en esta sección es cómo lograr la desidentificación de los datos empleando el **enmascaramiento de datos** o **data masking** (DM). Con ello lograremos ocultar o enmascarar esos datos que se consideran confidenciales o privados y que no pueden ser consultados por cualquier persona.

2.1. Conceptos previos

En esta primera sección introduciremos los conceptos principales que serán de gran utilidad para facilitar la lectura de este trabajo. Introducimos inicialmente la definición de **base de datos**.

Definición 2.2 *Una base de datos es un conjunto de información organizada y estructurada de manera lógica en un formato específico para que pueda ser accedida, gestionada y actualizada de manera eficiente.*

Según el uso que podamos darle, podemos diferenciar dos tipos diferentes de bases de datos [8]:

1. Bases de datos estáticas: son únicamente de lectura y se utilizan para almacenar datos históricos que pueden servir para el estudio de dichos datos a lo largo del tiempo.
2. Bases de datos dinámicas: la información almacenada puede ser modificada. En estas bases de datos se permite la inserción, actualización y borrado de datos.

Ahora que sabemos que una base de datos es una colección de datos que se organizan y almacenan en un sistema informático para facilitar el acceso y la gestión de dichos datos, podemos introducir los conceptos de **tabla**, **registro** y **campo** dentro de una base de datos.

Definición 2.3 *Una **tabla** es un conjunto de datos organizados en filas y columnas que representan un conjunto de entidades en un sistema. Cada fila de la tabla representa una instancia de la entidad, mientras que las columnas de la tabla representan los atributos o propiedades de dicha entidad.*

Introducimos también las definiciones de **registro** y **campo** que se corresponden con las filas y columnas de una tabla, respectivamente [7].

Definición 2.4 *Los **registros** en una base de datos son los items que hay en cada tabla. Es decir, cada registro se corresponderá con una fila de la tabla.*

Definición 2.5 *Además, las tablas de una base de datos se dividen en columnas, las cuales hacen referencia a un atributo de un determinado registro. A estas columnas (atributos) se le denominan **campos** de la base de datos.*

Los registros suelen identificarse mediante un número o *id* distinto, mientras que los campos representan diferentes características de estos registros.

Podemos representar también una tabla de una base de datos como una **matriz** compuesta por n filas y m columnas donde la fila i, con $i = 1, \ldots, n$, se corresponde con un registro de la tabla y la columna j, con $j = 1, \ldots, m$, se corresponde con un campo o atributo.

Si denotamos por $\mathbf{A}$ a la matriz que define la tabla, el elemento a_{ij} correspondería a la descripción de la característica j para el registro i, donde $i = 1, \ldots, n$ y $j = 1, \ldots, m$.

CAMPOS

REGISTROS

DS_VIEW_NAME	DS_VIEW_FIELD_NAME	DS_SECURITY_CLASSIFICATION	AUDIT_LOAD_DATE	AUDIT_LOAD_JOB	AUDIT_RECORD_SOURCE	AUDIT_LAST_SEEN_DATE
VENDOR	DS_STREET	Restricted	2023-03-17 10:13:45.753	JOB	SOURCE	2023-03-17 10:13:45.753
REQUEST	DS_AGENT_NAME	Public	2023-03-17 10:13:45.744	JOB	SOURCE	2023-03-17 10:13:45.744
REQUEST	DS_AGENT_MAIL	Restricted	2023-03-17 10:13:45.744	JOB	SOURCE	2023-03-17 10:13:45.744
SERVICE	DS_ONE_OFF_TYPE	Public	2023-03-17 10:13:45.733	JOB	SOURCE	2023-03-17 10:13:45.733
SERVICE	ID_CIRCUIT_ORIGIN_CAP	Public	2023-03-17 10:13:45.733	JOB	SOURCE	2023-03-17 10:13:45.733
USER_ALIAS	DS_TELEPHONE	Restricted	2023-03-17 10:13:45.684	JOB	SOURCE	2023-03-17 10:13:45.684
USER_ALIAS	DS_NEDAP	Public	2023-03-17 10:13:45.684	JOB	SOURCE	2023-03-17 10:13:45.684
USER_ALIAS	DS_LOGIN_NAME	Public	2023-03-17 10:13:45.684	JOB	SOURCE	2023-03-17 10:13:45.684
BLOCK	DS_LOCK_TYPE	Public	2023-03-17 10:13:45.639	JOB	SOURCE	2023-03-17 10:13:45.639

Figura 2.1: Registros (filas) y campos (columnas) de una tabla de datos

En la Figura 2.1 podemos ver un ejemplo de una tabla de auditoría en donde se registranlas operaciones de acceso a un conjunto de datos.En este caso,tenemos $n = 9$ filas o **registros** y $m = 7$ columnas o **atributos**.

2.2. Enmascaramiento

Aunque ya hemos hecho una introducción al enmascaramiento de datos, trataremos ahora de proporcionar una definición más formal:

Definición 2.6 *El **enmascaramiento de datos** es una técnica que consiste en cambiar una parte de los datos, o todo el dato, en un almacén de datos mientras se asegura de que la estructura siga siendo la misma. En el ámbito empresarial, el enmascaramiento de datos permite la exclusión de información confidencial del cliente del entorno de producción.*

Es decir, el enmascaramiento de datos es un método con el que podemos crear una versión similar de la estructura que tienen nuestros datos originalmente pero que no es la auténtica, ya que lo que haremos será modificar una parte de estos datos para protegerlos a la vez que creamos un sustituto para utilizar en algunas ocasiones que no necesitemos emplear los datos reales. Puede utilizarse, por ejemplo, para realizar pruebas de software o formación de usuarios.

Los datos pueden ser modificados de diferentes formas entre las que destacan la encriptación, la mezcla de caracteres o la sustitución de palabras. Utilicemos la técnica que utilicemos, debemos garantizar que los datos reales son transformados por ciertos valores de forma que sea imposible descubrir el valor real o realizar el proceso inverso, es decir, a partir del valor que tenemos, obtener el dato real.

Diremos entonces que el enmascaramiento de datos consiste en ocultar registros específicos dentro de la base de datos, asegurando que los datos sensibles son reemplazados con datos que parecen reales pero no lo son, de forma que pueden ser utilizados en entornos de pruebas con la seguridad de que las

pruebas son válidas, mientras que se garantiza la protección de los datos confidenciales.

La diferencia principal que existe entre la encriptación y el enmascaramiento de datos es que, en la primera técnica, los datos están protegidos contra un ataque a la privacidad cuando está en reposo pero para manipularlos necesitamos desencriptarlos primero, exponiéndolos. Sin embargo, con los datos enmascarados se pueden hacer operaciones manteniendo su estructura y comportamiento.

2.3. Tipos de Enmascaramiento de Datos

La aparición de las políticas de enmascaramiento de datos se debe, especialmente, a la expansión de las regularizaciones de privacidad. Esto provoca que sea más complicado implementar una plataforma unificada para administrar los requisitos de seguridad de datos y enmascaramiento de datos empresariales.

Entre las principales características que tiene la implementación de estas políticas de enmascaramiento, se encuentran el aumento de confianza en los datos, la rapidez de la creación, la puesta en marcha de modelos analíticos y la mejora en la eficiencia de los procesos.

Los responsables de seguridad de estas aplicaciones y de los datos deben considerar la política de enmascaramiento como un control de limpieza de seguridad fundamental y también aumentar la política de enmascaramiento utilizando controles más complejos, entre los que se encuentran la privacidad diferencial o los datos sintéticos, para evitar que los atacantes vuelvan a identificarse y conservar altos niveles de utilidad de los datos.

Definición 2.7 *La **privacidad diferencial** es un conjunto de técnicas que nos permiten recopilar y compartir datos con una "garantía matemática" de que las personas que proporcionaron esos datos no se van a ver afectadas de ninguna manera.*

Podemos introducir también el concepto de dato sintético:

Definición 2.8 *Los **datos sintéticos** son datos generados artificialmente que imitan los patrones existentes en los datos reales.*

Estos líderes también deben adoptar las funciones de dichas políticas que se encuentran en las plataformas de seguridad de datos para reducir el riesgo de los datos al evitar la división de sus estrategias de enmascaramiento de datos en diferentes fuentes de datos.

Hablaremos a continuaciñon lo que se conoce como **controles BDSP**.

Definición 2.9 *Los **controles BDSP** son un grupo de medidas de seguridad utilizadas para proteger datos privados o confidenciales. Con el fin de disminuir el riesgo de exposición y proteger la privacidad de las personas, estos controles se realizan para enmascarar o anonimizar datos personales o confidenciales.*

Para proteger la seguridad y la privacidad de los datos de los usuarios, los BDSP emplean con frecuencia métodos como la **tokenización**, el **cifrado**, la **supresión** y el **enmascaramiento de datos**.

Podemos proporcionar una breve definición de cada uno de los tres primeros conceptos.

- **Tokenización**: es el proceso que permite proteger datos sensibles, sustituyéndolos por otros que no lo son (tokens). Un token no tiene valor, es un identificador que permite volver al dato sensible. Se utiliza, por ejemplo, con las tarjetas de crédito.
- **Cifrado**: el cifrado surge a partir de la criptografía, que es la ciencia encargada de ocultar información ante personas no autorizadas.
- **Supresión**: se conoce así a la eliminación directa de los datos que pueden resultar confidenciales.

Para disminuir el impacto de rendimiento del enmascaramiento de datos en los equipos de desarrollo y prueba, los controles BDSP inadecuados pueden reemplazarse con mecanismos de enmascaramiento de datos en la virtualización de datos o en la capa de aplicación.

Estos son los dos tipos que se conocen de técnicas de enmascaramiento de datos:

- SDM (Static Masking Data) o Enmascaramiento de Datos Estático
- DDM (Dynamic Masking Data) o Enmascaramiento de Datos Dinámico

A continuación, se proporcionarán unas definiciones de estos dos tipos de enmascaramiento de datos y veremos las diferencias que existen entre ambos así como cuál de los dos es el más utilizado en la actualidad.

2.3.1. Enmascaramiento de datos estático y dinámico.

En esta sección presentaremos las principales características de cada uno de ellos y las propiedades que los diferencian.

Enmascaramiento de datos estático (SDM):

Este tipo de enmascaramiento ayuda a crear una copia limpia de nuestra base de datos alterando los campos que consideremos confidenciales hasta que consigamos que se pueda compartir a una copia de la base de datos de manera completamente segura. Normalmente, este método requiere crear una copia de respaldo de una base de datos en producción, cargarla en un entorno aparte, eliminar los datos que no necesitaremos y enmascarar los que sí queremos. Una vez hecho esto, enviamos la copia enmascarada a la ubicación de destino.

Enmascaramiento de datos dinámico (DDM):

Esta versión de data masking ofrece enmascaramiento en tiempo real y basado en roles de datos sensibles en los entornos de producción [1]. El objetivo de esta variante es reemplazar los datos confidenciales en tránsito dejando los datos originales en reposo y sin cambios. Es decir, los datos no se almacenan en un almacén de datos secundario. Lo que se hace en este caso es transmitir directamente desde el sistema de producción y se consume desde otro sistema en un entorno de prueba o desarollo.

[1] Consideraremos entornos de producción aquellos en los que se encuentran los datos necesarios para las aplicaciones finales.

En resumen, el enmascaramiento de datos estático (SDM) reemplaza permanentemente los datos confidenciales al alterar los datos en reposo, mientras que el enmascaramiento dinámico de datos (DDM) está diseñado para reemplazar los datos confidenciales en tránsito al dejar los datos originales intactos.

Para el enmascaramiento de datos será necesario designar roles o usuarios con diferentes privilegios, donde solo los usuarios con determinados privilegios tienen acceso a la información confidencial.

Evolución de DDM frente a SDM

En concordancia con lo visto en el apartado anterior, el enmascaramiento de datos estático, SDM, se utiliza cuando queremos proporcionar datos realistas que puedan utilizarse en entornos de desarrollo y prueba sin que se pueda acceder a la información confidencial.

Lo que diferencia a esta técnica es la realidad de los datos, por eso tenemos que conseguir que sean los más realistas posibles para permitir que la fase de desarrollo y prueba sea más efectiva logrando identificar posibles errores. Esto se traduce en una reducción de costes de producción y aumenta la calidad del enmascaramiento estático.

Además, en el enmascaramiento dinámico de datos se eliminan aquellos que son confidenciales de la base de datos enmascarada por lo que resulta imposible ver la información confidencial en caso de que se quisiese acceder a la base de datos, ya que los datos originales fueron sustituidos por otros que son semejantes.

Por otra parte, el enmascaramiento estático de datos no afecta al rendimiento ya que el enmascaramiento de datos se realiza al principio, por lo que el rendimiento no se ve perjudicado una vez que hayamos enmascarado la base de datos y está disponible para diferentes funciones. La base de datos de producción queda totalmente protegida e inaccesible.

Si bien las tecnologías SDM y DDM han crecido en estos últimos años, se puede observar una ligera evolución en los productos DDM. El interés en el enmascaramiento de datos está impulsado por tendencias, que incluyen:

- La expansión y mayor madurez de las leyes de privacidad en jurisdicciones de todo el mundo.
- Crecimiento y mayor concienciación relacionada con la seguridad de los datos para proyectos de análisis avanzado e inteligencia artificial/aprendizaje automático (AI/ML).
- Su migración a lagos de datos basados en la nube.
- Mayor atención a la seguridad de los datos internos utilizando principios de confianza cero en los que todos los accesos deben definirse y autorizarse con precisión.

En las regiones afectadas por las regulaciones de privacidad, el nivel de interés en la tecnología DM se mantiene estable. Presentamos dos citas sacadas de libros que reflejan este hecho:

"La cantidad de consultas de clientes que recibió Gartner se mantuvo estable en los mercados europeo y chino, y las consultas continuaron indicando un interés creado en los casos de uso de DM".

"Los niveles de interés en América del Norte han bajado solo levemente año tras año, con un enfoque en las verticales que dependen en gran medida de los datos personales, como los servicios financieros diversificados y la atención médica".

Con enmascaramiento estático de datos, la mayor parte de los administradores, programadores y probadores de bases nunca utilizan la base de datos de producción, ya que el trabajo se realiza sobre la base de datos de desarrollo, lo que proporciona un mayor nivel de protección y es necesario para muchos entornos. A pesar de esto, no es una solución completa, ya que no protege acerca de que los usuarios autorizados puedan ver y extraer información no autorizada.

Las soluciones estáticas requieren la extracción de todos los datos antes de que se enmascaran, es decir, los datos salen de la base de datos desenmascarados.

Un punto crítico en el enmascaramiento estático es el proceso estándar ETL (extracción, transformación y carga, de esto hablaremos más adelante). Es decir, la información de base de datos se extrae tal como está desde la base de datos, y sólo se transforma después. Los riesgos del enmascaramiento dinámico es que dicho enmascaramiento no ha eliminado correctamente los datos reales, mientras que en el enmascaramiento estático el riesgo es no estar utilizando una plataforma segura.

Aún así, la base de datos no está totalmente protegida ya que las personas como administradores, control de calidad o desarrolladores, sí tienen permisos de acceso a la base de datos en directo y pueden ver estos datos sin enmascarar.

Para llevar a cabo el enmascaramiento de datos se deben realizar dos pasos: primero realizamos las actividades en un entorno de prueba y luego lo implementamos en un sistema de producción. En el caso de que el sistema de producción no funcione o presente algún problema, algunas de los usuarios que mencionamos anteriormente (administradores, controles de calidad, desarrolladores, etc) deben depurar el sistema y para ello necesitarán unos determinados permisos que son específicos para acceder a la base de datos original.

La principal ventaja del enmascaramiento dinámico de datos (DDM) frente al estático (SDM) es que ha sido diseñado para asegurar o enmascarar datos en tiempo real en sistemas en el resto de sistemas.
Lo que hace esta técnica es enmascarar los datos sensibles a los que se quiere acceder y, al hacerlo en tiempo real, la información sensible nunca sale de la base de datos por eso, cuando alguna persona autorizada intenta acceder a ellos, los datos son ilegibles y es imposible desenmascararlos, lo que provoca una mayor seguridad en los datos originales.

Además, de esta forma, se consigue que los datos sean totalmente funcionales con fines de desarrollo o prueba, sin mostrarse a los usuarios no autorizados.

Al utilizar reglas avanzadas de enmascaramiento de datos, podemos saber si un campo concreto se puede mostrar a una persona en particular. Por ejemplo, una persona podría ser capaz de acceder a un registro de la base de datos mediante un usuario determinado con unas credenciales específicas, mientras que el acceso se haría imposible mediante una orden directa de la base de datos.

2.3.2. Enmascaramiento según la parte del dato a enmascarar

Además de tener dos técnicas distintas de enmascaramiento dependiendo si la técnica es estática o dinámica, enmascaramiento estático de datos (SDM) y enmascaramiento dinámico de datos (DDM),

respectivamente, dependiendo de qué parte del dato queremos enmascarar, aparecen otros dos tipos de enmascaramiento:

1. Enmascaramiento parcial: lo que hacemos con esta técnica es enmascarar únicamente una parte del dato, pudiendo ver otra parte del mismo (como la extensión en los correos electrónicos, el prefijo en los teléfonoc, etc.)
2. Enmascaramiento total: en este caso el dato se encuentra enmascarado completamente y no tenemos información ninguna acerca de él. Solamente las personas con determinados permisos podrían acceder a estos datos.

Ejemplo 2.10 *Dado un correo personal, alba@gmail.com, un enmascaramiento parcial sería, por ejemplo, ****@gmail.com, donde solo se podría ver la extensión de este, mientras que el enmascaramiento total sería ********** dejando así el dato original completamente enmascarado y descifrable para cualquier persona que intente consumir el dato real y no tenga los permisos necesarios para hacerlo.*

Nota: En el ejemplo anterior, hay tantos asteriscos (*) como caracteres tenía la dirección de correo electrónico pero también es común, en el enmascaramiento total, poner una cantidad fija de caracteres que hagan de máscara (en este caso *) independientemente de la longitud que tenga el dato que queremos enmascarar. Es decir, si queremos enmascarar el dato albagude@gmail.com y alba@gmail.com, utilizaríamos el mismo número de asteriscos.

Otro ejemplo importante podrían ser los números de teléfono donde, en un tipo de enmascaramiento parcial, nos interesaría que se viera solamente la extensión del número, por ejemplo +34 si es España.

2.4. Modelado de datos

En esta sección, empezaremos hablando del modelado de datos y las relaciones entre diferentes entidades.

Los diagramas de esquema que se utilizan fueron formalizados por Charles Bachman en 1960, quién utilizó rectángulos para denotar tipos de registros y flechas dirigidas desde un determinado registro hacia otro u otros para denotar una relación entre esos registros específicos. Existen las relaciones de uno a uno (de un registro a otro) o de uno a muchos (de un registro a más de un registro de destino).

El enfoque entidad-relación (ER) para el modelado conceptual de bases de datos fue descrito por primera vez en 1976 por Peter Chen. El énfasis primordial en el modelado de ER está en la simplicidad de la lectura.

Definición 2.11 *Un modelo de datos es la representación gráfica de un conjunto de datos con finalidad empresarial.*

Un modelo de datos es la combinación de los siguientes componentes:

1. Una colección de tipos de estructura de datos.

2. Un conjunto de operadores o reglas que se puedan aplicar a cualquier instancia válida de los tipos de datos en el punto anterior, para recuperar datos de cualquier parte de esas estructuras.

3. Una colección de reglas generales de integridad para definir el conjunto de estados consistentes de la base de datos.

Un modelo de datos permite organizar los elementos de la base de datos y establecer cómo se relacionan entre sí. Además, este debe estar compuesto por un conjunto de ítems (o registros) que sean de fácil comprensión y que combinan texto y símbolos.

Un modelo de datos es una entrega fundamental en proyectos de software operativo y analítico, que se realiza durante la fase de análisis o diseño de la solución [10].

Definición 2.12 *El modelado de datos es el proceso de creación de una representación visual o esquema que define los elementos o entidades de los que disponemos datos, su atributos o características y la relación entre ellas, así como la relación con los sistemas gestores de los datos, siendo su objetivo representar cómo se estructuran y organizan los datos en una base de datos.*

2.4.1. Beneficios del Modelado de Datos

Algunas de las ventajas de un modelado de datos son:

- Modelar permite definir el problema y proponer diferentes escenarios, eligiendo el que sea mejor para nosotros.
- Permite anticipar errores.
- Tiene un alcance más claro: permite enfocar el alcance a algo tangible que ayuda a los desarrolladores y consumidores de datos a decidir qué incluir.
- Menos errores, ya que el modelado de datos requiere definir bien los conceptos y evitar confusiones. Además permite identificar los casos redundantes e incompletos.
- Proporcionan un mejor rendimiento al ser un modelo bien propuesto. Para ello, debemos incluír conceptos que sean claros y coherentes y las necesarias reglas para traducirlo a un modelo de base de datos.
- Estos modelos documentan conceptos importantes y sirven para materializar ideas.
- Una mejor gestión de los riesgos, ya que permite estimar la complejidad de la solución y tener más información sobre el riesgo de un proyecto.

Para crear un modelo de datos primero tenemos que crear un modelo conceptual y entenderlo. Un modelo conceptual define lo que debe estar en la estructura del modelo para asegurar los conceptos de negocio. Este modelo está basado en entidades, atributos y las relaciones entre entidades y está diseñado por un arquitecto de datos. Además, es también una perspectiva de modelado que se dirige a los usuarios de negocio principalmente.

Un ejemplo de modelo entidad-relación es el siguiente:

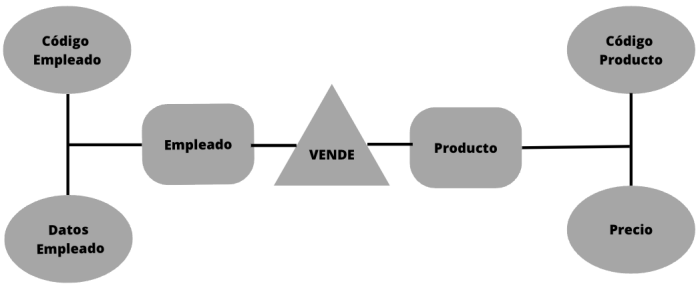


Figura 2.2: Modelo entidad-relación.

2.5. Diseño de Procesos ETL

Los procesos o herramientas ETL tienen como objetivo proporcionar los componentes software necesarios para extraer, validar, limpiar, formatear y cargar una gran cantidad de información procedente de una o más fuentes en otro sistema operacional.

Definición 2.13 *Un proceso ETL (Extract, Transformate and Load) es una actividad de Data Warehouse que consiste en extraer datos de múltiples fuentes de datos, transformarlos y cargarlos en un solo destino o almacenamiento de datos para ser consumidos por aplicaciones para la toma de decisiones [9].*

Definición 2.14 *Un almacén de datos o Data Warehouse es, según Immon, (*Inmon 02, Imhoff & Galemmo 03 et al.*), un conjunto de datos que se refieren a un determinado ámbito, no volátil, integrado en el tiempo y sujeto a cambios que es utilizado para tomar determinadas decisiones.*

Un proceso ETL consiste en una serie de actividades de preparación de datos. Estos procesos permiten:

- Supervisar el flujo de los datos transaccionales.
- Corregir datos que faltan y eliminar los errores.
- Transformar los datos de las diferentes fuentes para unificarlos.
- Dar una estructura a los datos para poder ser utilizados por los usuarios finales.
- Proporcionar una prueba medible de la calidad de los datos.

Los procesos ETL forman parte del "Business Intelligence", por lo que también tiene objetivos dirigidos a estrategias de negocio, entre los que destacan:

- Apoyo en la toma de decisiones, permitiendo a los usuarios analizar solamente aquellos datos que necesiten.
- Accesibilidad de la información.
- Orientación al usuario final, permitiendo al usuario utilizar estas herramientas sin tener conocimientos técnicos específicos.

A lo largo del tiempo y con diferentes objetivos, utilizaremos diferentes tipos de modelado de datos, entre los que destancan las bases de datos jerárquicas, en red, relacionales y orientadas a objetos. A continuación, añadiremos una pequeña definición de cada uno de los tipos y también una representación gráfica. Cabe destacar el último tipo de modelado que es el relacional, que será en el que nos centraremos a lo largo de este trabajo.

- Modelado de datos jerárquico: son los primeros utilizados en la creación de una base de datos. Son modelos semánticos e independientes del tipo de base de datos que queremos crear. En estos modelos se describen los datos tal y como los captamos.

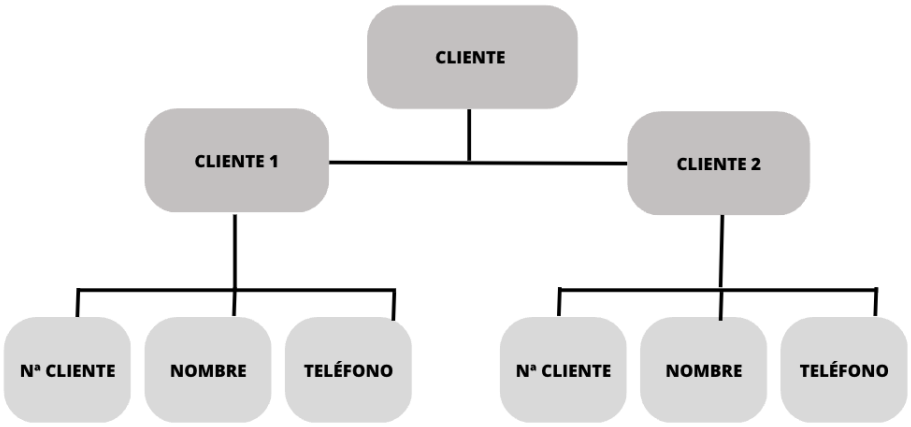


Figura 2.3: Modelo jerárquico

- Modelo de datos en red: fue propuesto por Peter Chen en 1976. Sus elementos básicos son las entidades, las relaciones y los atributos. En este caso, la base de datos está formada por un conjunto de registros que están enlazados entre sí creando una red, de ahí el nombre.

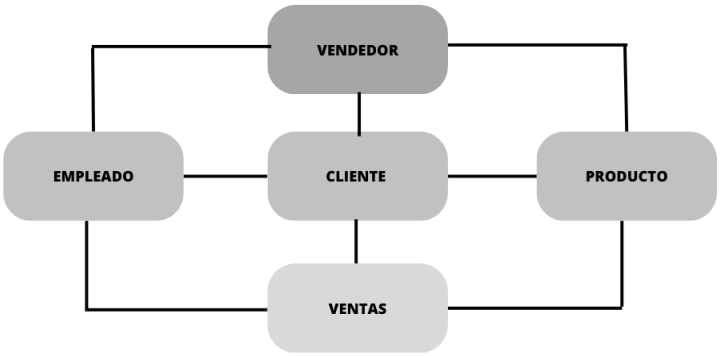


Figura 2.4: Modelo en red

- Modelos orientados a objetos: a partir de 1996 se integraron varios métodos de análsiis orientado a objetos dando lugar al Lenguaje del Modelado Unificado. Los elementos principales de este tipo de modelado son las clases, con sus métodos y atributos, y las diferentes relaciones que pueden existir entre dichas clases.

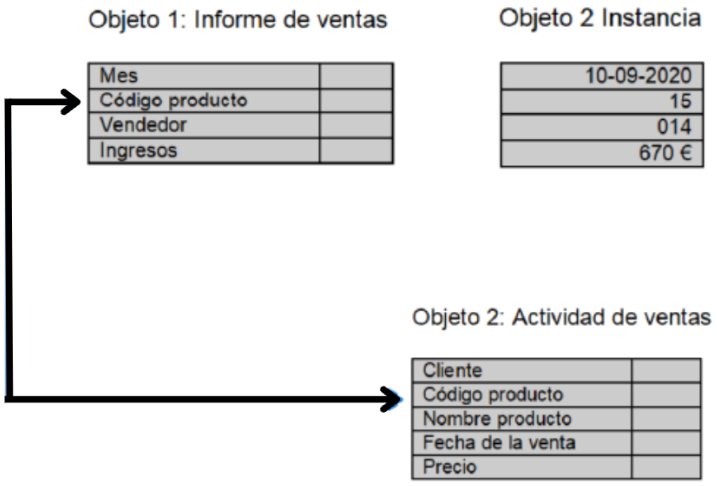


Figura 2.5: Modelo orientado a objetos

- Modelos relacionales: posterior a los descritos anteriormente y fue desarrollado en 1970 por Codd. En estos modelos se utilizan tablas para la representación lógica de los datos y las relaciones entre ellos. Son las más utilizadas hoy en día y en las que nos centraremos a continuación.

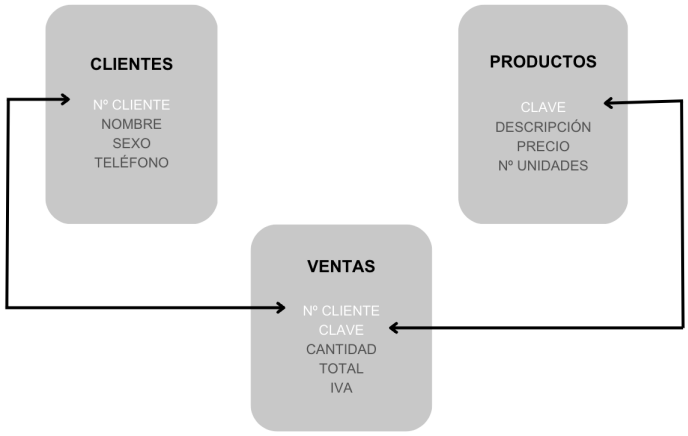


Figura 2.6: Modelo relacional

Los modelos de datos en red y los orientados a objetos permiten anidar los datos de manera que aceptan objetos que se refieren a otros como valores de sus atributos. Sin embargo, los modelos que están orientados a objetos pueden tener ciclos mientas que en el modelo de redes, estos clicos requieren de estructuras artificiales en el sistema. En la figura 2.6 los campos en blanco son las claves primarias de cada una de las tablas y las flechas representan la unión con la tabla **VENTAS**.

2.6. Modelado de Datos Relacional

A pesar de que existen diferentes tipos de modelos o bases de datos (bases de datos jerárquicas, en red, relacionales y orientadas a objetos), a lo largo de este trabajo nos centraremos únicamente en las bases de datos relacionales, ya que son con las que hemos trabajado en el proyecto durante el período de prácticas.

El modelo relacional ha sido desarrollado por Codd en 1970. Como ya hemos comentado, en este modelo se utilizan tablas para representar de manera lógica los datos, además de las relaciones entre ellos [11].
En este contexto del modelo relacional tienen especial importante las **claves**, que son campos o atributos que identifican un determinado registro de manera única. Es decir, el valor de este campo no puede aparecer repetido.

Como bien dice la palabra, el concepto más importante de **modelo relacional** es el de relación. Mostramos a continuación un conjunto de definiciones previas que son muy relevantes a la hora de hablar de los modelos relacionales, para poder definir posteriormente el concepto de **base de datos relacional**:

- Relación: como bien dijimos este es el concepto más importante de un modelo relacional y podemos imaginar una relación como una tabla de datos que guarda información en sus filas y columnas.
- Atributo: aunque ya fue definido anteriormente, un atributo es una columna o campo de la tabla que determina una característica dentro de una relación.

- Dominio: son los posibles valores que puede tomar un determinado atributo.
- Tupla: llamaremos así a un registro o fila dentro de la relación.
- Grado: número de atributos de una tabla o relación.
- Cardinalidad: el número de tuplas, filas o registros dentro de una relación.

Una vez introducidos estos conceptos, ya estamos en condiciones de definir lo qué es una **base de datos relacional**:

Definición 2.15 *Diremos que una* ***base de datos relacional*** *es un conjunto de relaciones normalizadas entre los datos que la conforman, donde cada relación se puede identificar con un determinado nombre [12].*

Podemos pensar que una base de datos relacional es un depósito de datos integrados con una determinada estructura que representa las diferentes interrelaciones y restricciones que existen entre ellos en un problema determinado que nos podemos encontrar en el día a día.

A su vez, los datos deben mantenerse independientes de dichas bases de datos y la definición y descripción de cada uno de ellos aparecen almacenadas junto a estos.

Por otra parte, en una base de datos no hay datos redundantes ni duplicados, únicamente podrá aparecer repetida aquella información que necesitaremos para representar las diferentes relaciones entre los datos.

Ejemplo 2.16 *Para entender mejor esto, podemos pensar en una persona que hace varios pedidos a una determinada tienda. En este caso, la dirección, por ejemplo, no aparecería repetida, pero si podría aparecer repetido el número de cliente.*

Es importante saber detectar cada uno de los elementos que constituyen los modelos relacionales para poder representar los metadatos de la base de datos.

En este trabajo, y en múltiples ocasiones, necesitaremos manipular nuestros datos mediante sentencias de lenguaje SQL, sobre todo para llevar a cabo los mecanismos de inserción y/o extracción de la información.

Relacionado con lo que se ha comentado anteriormente, cabe mencionar que los datos también han de ser independientes de los programas que los utilice, lo que implica que una actualización o mejora de un determinado programa no supone una modificación en la información disponible en nuestra base de datos.

Además de esto, los datos que utilicemos deben ser íntegros ya que debemos asegurarnos de que la información con la que estamos trabajando sea verdadera y real.

Como ya hemos comentado, el acceso a la base de datos solamente está disponible para determinados usuarios que tengan los permisos requeridos para ellos, así como las modificaciones u operaciones que tengan que hacer en la misma.

Para finalizar el capítulo, cabe mencionar que la base de datos tiene que estar siempre disponible para poder ser visualizada o utilizada por los programas y usuarios que lo requieran en cualquier momento.

En el siguiente capítulo mostraremos las soluciones que hemos encontrado y empleado para enmascarar los campos de las tablas que se piden. Además, indicaremos como crear el código que empleamos para lograr el objetivo deseado.

Capítulo 3

Soluciones de enmascaramiento dinámico de datos

En este tercer capítulo del trabajo presentamos las principales soluciones para enmascarar los datos de una o varias tablas determinadas utilizando, principalmente, lenguaje **SQL** combinado con lenguaje **JAVA** para poder llevar a cabo el procedimiento que enmascara los datos en la plataforma Snowflake.

El código completo del procedimiento almacenado que se ha utilizado se encuentra en el Apéndice A y en este capítulo solamente mostraremos las principales partes de código SQL que permiten llevar a cabo el enmascaramiento. Los datos utilizados en las prácticas serán introducidos en el capítulo 4 de "Aplicación del enmascaramiento dinámico de datos".

El siguiente capítulo presentará la aplicación práctica de los conceptos presentados en el presente capítulo.

3.1. Enmascaramiento Dinámico de Datos en Snowflake

Como hemos comentado en capítulos anteriores, **Snowflake** es un almacén de datos en la nube, disponible desde 2015, que tiene diversas características que lo distinguen de otros almacenes de datos y hacen que sea uno de los más utilizados por las empresas y organizaciones a día de hoy.

Figura 3.1: Snowflake como plataforma analítica

Esta plataforma se compone de un sistema multiusuarios, es decir, varias personas puedes estar accediendo y utilizando la base de datos desde diferentes usuarios y ordenadores. También es una plataforma transaccional, segura y altamente escalable con un soporte completo de SQL para datos que no tienen esquemas o datos semiestructurados.

En nuestro caso, el sistema se ofrece como un servicio de pago por la utilización en la nube de **Microsoft Azure**. De esta forma, se paga por la capacidad que se esté utilizando y cada usuario puede cargar sus datos en la nube y gestionarlos desde cualquier lugar y en cualquier momento, requiriendo únicamente acceso a Internet.

Podemos decir entonces que Snowflake destaca por su extrema elasticidad y disponibilidad, por las capacidades de tratamiento de datos semiestructurados y sin esquema y por su seguridad de extremo a extremo.

3.2. Creación de la política de enmascaramiento

Como hemos dicho, para enmascarar los datos que necesitamos, lo primero será crear un procedimiento almacenado que nos permita enmascarar y desenmascarar los datos cuando lo necesitemos.

Para definir una política de enmascaramiento empezaremos mostrando el código o **query** en lenguaje SQL que permite enmascarar un determinado dato. Para ello necesitaremos:

- Definir el nombre de la política de enmascaramiento y de las tablas que emplearemos.
- Determinar el tipo de usuario que tendrá los permisos para ver los datos sin enmascaramiento.
- Seleccionar el enmascaramiento que queremos poner en lugar del dato original: palabra clave, números, cifras, dígitos, asteriscos, almohadillas, etc.

La política de enmascaramiento creada para enmascarar los datos utilizados en el siguiente capítulo es:

```
1  CREATE MASKING POLICY <nombre_politica_enmascaramiento>
2  AS (val <tipo>) RETURNS STRING ->
3  CASE
4  WHEN CURRENT_ROLE() IN ('ROLE_ADMIN') THEN val
5  ELSE <mask>
6  END;
```

Inicialmente, se crea una "masking policy" bajo el nombre de <nombre_politica_enmascaramiento>. Entre paréntesis, sustituiremos <tipo> por 'number' si el campo de datos que queremos enmascarar es de tipo numérico y escribiremos 'string' si el campo a enmascarar es una cadena de texto. En nuestro caso, indicaremos las tablas que utilizaremos para llevar a cabo en el enmascaramiento.

Un ejemplo de política de enmascaramiento que hemos creado durante el trabajo es la creada para enmascarar campos de la tabla **customer_sap**:

```
1  CREATE OR REPLACE masking policy restricted_masking_customer_sap
2  AS (val string) RETURNS string ->
3  CASE
4  WHEN EXISTS
5  (SELECT 1 FROM cap_dm_security_access a
6  LEFT OUTER JOIN cap_dm_aad_roles b
7  ON a.ID_SECURITY_ACCESS = b.ROLE
8  WHERE ID_SECURITY_ACCESS='G_SNW_MSK_CUSTOMER_SAP'
9  AND USER_ROLE = current_role()) then val
10 WHEN current_role() = 'DEVELOPER' then val
11 ELSE '*********'
12 END;
```

La máscara que elegimos en este caso para enmascarar los datos confidenciales es '*********'. El nombre de la política de enmascaramiento elegido es **restricted_masking_customer_sap** y las tablas involucradas son **cap_dm_security_access** y **cap_dm_aad_roles**.

Una vez creada la política de enmascaramiento debemos aplicarla. Para ella utilizamos el siguiente código donde indicamos la columna que queremos enmascarar y la tabla correspondiente:

```
1  ALTER TABLE <tabla> MODIFY COLUMN <campo> SET MASKING POLICY
2  <nombre_politica_enmascaramiento>;
```

Donde, en el ejemplo, sustituiríamos <tabla> por customer_sap, <nombre_politica_enmascaramiento> por restricted_masking_customer_sap y en <campo> escribiríamos la columna que queremos enmascarar de la tabla.

Con respecto a lo comentado anteriormente:

- El nombre de esta política de enmascaramiento es **customer_sap** y las tablas involucradas son la a, **cap_dm_security_access**, y la b, **cap_dm_aad_roles**.
- Le damos permisos para ver los datos originales al usuario DEVELOPER.
- La máscara que utilizaremos para nuestros datos es '*********'

La política de enmascaramiento anterior enmascara solamente datos de la tabla **customer_sap** por lo que para cada tabla distinta que queramos enmascarar necesitamos una política de enmascaramiento diferente. En el procedimiento completo, que se muestra en el Apéndice A, aparecerán las políticas de enmascaramiento **restricted_masking_vendor** y **restricted_masking_site_alias** para enmascarar los campos que se necesitaban de las tablas **vendor** y **site_alias**, respectivamente.

Además, si quisiéramos enmascarar dos campos de una tabla pero un campo que solo pudieran ver los usuarios 'DEVELOPER', por ejemplo, y que el otro campo solo pudiera verlo los que tuvieran usuario 'READER'. En este caso también debemos crear dos políticas de enmascaramiento diferentes, ya que los campos que queremos enmascarar involucran a dos roles distintos de usuarios.

En este apartado se ha presentado cómo crear un procedimiento almacenado que combina lenguaje SQL y JAVA, dentro del cual se indica la forma de obtener los datos de una determinada tabla que contiene los campos a enmascarar del resto de las tablas.

Después de crear el procedimiento almacenado, dependiendo los campos de la tabla que queramos enmascarar, el procedimiento llamará a una cierta política de enmascaramiento (que en este caso creamos 3, una para cada tabla) y posteriormente enmascará el campo deseado, que fue el que obtuvo de la primera tabla, que en nuestro caso se llamará **cap_tech_dim_field_details**.

3.3. Roles y privilegios en Snowflake

Los **roles** y **privilegios** que se otorgan a los diferentes **usuarios** de Snowflake permiten o no el acceso a una determinada información y a determinadas sentencias de código, de manera que en algunos casos solamente se permite ver la información disponible, sin poder realizar cambios ni operaciones en los datos.

Para entrar en contexto, introduciremos inicialmente unos conceptos previos [13]:

Definición 3.1 *Definimos **rol** como una entidad a la podemos atribuír privilegios. Los roles se pueden asignar a usuarios o a otros roles, creando en el último caso una jerarquía de roles.*

Definición 3.2 *Un **privilegio** es un nivel de acceso definido a un determinado objeto.*

Definición 3.3 *Un **usuario** puede ser una persona o programa y es una entidad de usuario que reconoce directamente Snowflake.*

La forma en la que trabaja Snowflake permite el acceso a objetos mediante privilegios asignados a roles, que a su vez se asignan a otros roles o usuarios.

Se otorgan los roles a los usuarios para que estos puedan realizar las operaciones necesarias para llevar a cabo su trabajo. En Snowflake se permite realizar diferentes acciones usando privilegios independientes y para eso existe la posiblidad de asignar varios roles a un mismo usuario. Como consecuencia de esto, existirán ocasiones en las que podamos identificarnos con un determinado rol para llevar a cabo una acción específica pero si tenemos que realizar otra operación, tendremos que cambiar de rol.

Hacemos referencia ahora a los dos modelos diferentes que podemos abordar en Snowflake con respecto a su control de acceso:

- Control de acceso discrecional: cada objeto tiene un propietario que puede otorgar acceso a él.
- Control de acceso basado en roles: los roles contienen diferentes privilegios que se asignarán a los diferentes usuarios.

En nuestro caso, el control de acceso se basará en roles y en las secciones posteriores explicaremos cómo se otorgan los roles y permisos y cómo se usa un determinado rol.

Como hemos comentado, para realizar este trabajo se ha creado un procedimiento almacenado en lenguaje SQL utilizando la plataforma Snowflake y la forma que tendremos de acceder será mediante un modelo basado en roles. Por eso es importante definir ahora los diferentes tipos de rol que existen en esta plataforma:

- ORGADMIN: es el administrador de la organización y gestiona las operaciones a este nivel. Este rol puede ver y crear las cuentas de la organización.
- ACCOUNTADMIN: es el administrador de cuenta y está formado por los roles SYSADMIN y SECURITYADMIN, que se definen a continuación. Solamente se puede asignar a un número limitado de usuarios por ser el nivel superior del sistema.
- SECURITYADMIN: es el administrador de seguridad y puede administrar y crear roles y usuarios. Además, hereda los privilegios del rol USERADMIN.
- USERADMIN: usuario y administrador de roles que solamente se dedica a la gestión de los mismos.
- SYSADMIN: administrador del sistema. Este rol puede crear almacenes y bases de datos en una cuenta.
- PUBLIC: rol que se otorga de manera automática a cada rol y usuarios.

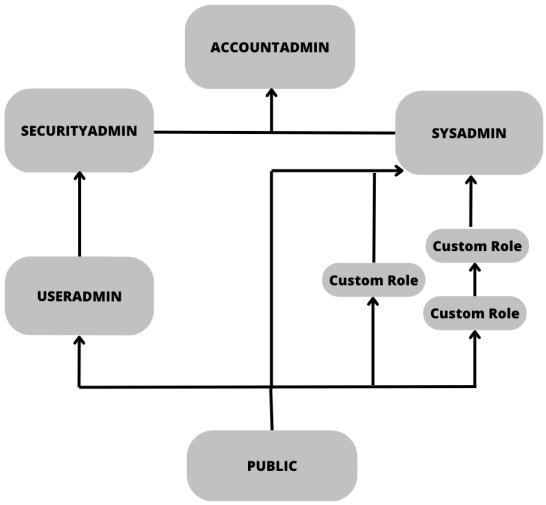


Figura 3.2: Tipos de roles que existen en Snowflake

Introducimos ahora la figura 3.3 para mostrar un ejemplo que nos facilite ver cómo se heredan y otorgan los roles y privilegios para, posteriormente, concedérselos a un determinado usuario:

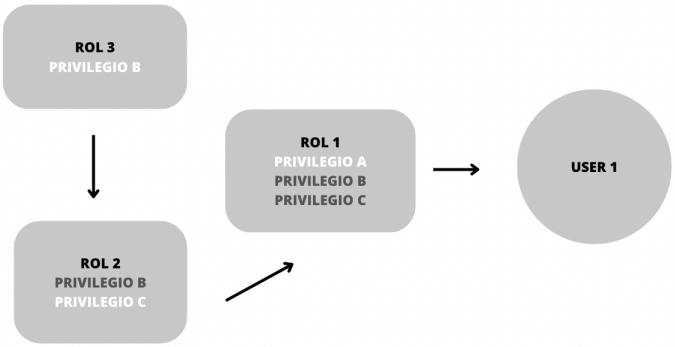


Figura 3.3: Ejemplo de roles

Los privilegios de un rol son sus propios privilegios y de aquellos roles que se le asignan. Es decir, cuando atribuímos un rol a otro rol, los privilegios del primero se otorgan directamente al segundo.

De esta forma, en la Figura 3.3, el rol 1 tiene como privilegio propio el privilegio A y tiene asignado el rol 2. El rol 2 tiene el priv. C como propio y herada el rol 3, con el privilegio B. En definitiva, el rol 3 es el que tiene menos privilegios y el rol 1 el que más.

Como hemos visto, podríamos asignar un rol a un usuario o a otro rol. En el primer caso, si otorgamos un rol a un usuario, este puede realizar todas las operaciones que están permitidas para ese rol. En el caso de asignar un rol a otro rol, lo que estamos haciendo realmente es crear una relación "principal-secundario" entre los roles, el principal sería el rol que otorga, lo que da lugar a una jerarquía de roles.

3.3.1. Otorgar roles y permisos en Snowflake

Todo lo que hemos estado hablando hasta ahora depende del tipo de usuario al que tengamos acceso en la plataforma Snowflake, ya que si no tenemos los permisos necesiarios, no podremos ver los datos sin enmascarar.

Como dijimos antes, se puede asignar un rol a un usuario o a otro rol. Para conceder un rol a otro rol se utiliza el código:

```
1  GRANT ROLE <name> TO { ROLE <parent_role_name> | USER <user_name> }
```

En este caso <name> denota el rol que se va a otorgar, <parent_role_name> otorga el rol al rol especificado y <user_name> otorga el rol al usuario especificado.

Para poder realizar esto, el rol utilizado para ejecutar el comando debe tener los privilegios de propiedad del rol que se otorga a un usuario o a otro rol.

Para evitar este problema, será más fácil utilizar el rol que tiene el privilegio global **manage grants**, y estos privilegios solo lo tienen el rol **security admin** o un rol superior.

Los roles definidos por el sistema no necesitan otorgarse a otros roles porque Snowflake define y mantiene la jerarquía de roles para estos.

Capítulo 4

Aplicación del enmascaramiento dinámico de datos

4.1. Preparación del entorno

A lo largo de este capítulo se lleva a cabo la aplicación de los conceptos presentados previamente para poner en práctica lo visto hasta ahora. Recordemos que el objetivo final de este trabajo es crear un procedimiento almacenado en lenguaje SQL para conseguir enmascarar unos determinados campos de unas tablas específicas que la empresa considera confidenciales.

Como definimos en capítulos anteriores, el enmascaramiento dinámico de datos (DDM) se realiza en tiempo real y basado en roles de datos sensibles en los entornos de producción. Lo que hace realmente este método es reemplazar los datos confidenciales en tránsito manteniendo en reposo y sin cambios los datos originales, que no se utilizarán ni transmitirán.

Empezaremos haciendo un breve resumen de los datos con los que hemos trabajado y utilizaremos un caso particular , donde explicaremos los campos a enmascarar de una determinada tabla, para llevar a cabo el enmascaramiento de datos y poder ver el proceso de una manera rápida e intuitiva.

Después explicaremos el proceso, es decir, veremos una parte del código que hemos empleado para realizar el enmascaramiento de los campos descritos que asumiremos como confidenciales.

Para cerrar el capítulo, aplicaremos el código descrito al ejemplo elegido para ver el resultado que obtendríamos en nuestro caso y hacer una breve validación del código realizado para nuestros campos y tablas.

Debemos tener en cuenta que la posibilidad de ver enmascarados o no los campos que se ocultan con el procedimiento llevado a cabo, depende de los permisos y roles otorgados a nuestro usuario.

Si poseemos el rol especificado para una determinada política de enmascaramiento, bastará con activar el rol adecuado y seleccionar la tabla que deseamos ver. Por el contrario, si no tenemos los permisos necesiaros, veremos los datos enmascarados sea cual sea el rol que utilicemos para realizar la consulta.

4.2. Descripción de los datos

Para la realización de este trabajo se han utilizado los datos de un proyecto de una compañía española de telecomunicaciones, **Cellnex Telecom SA**, y en esta sección describiremos el proceso de creación de nuestra política de enmascaramiento y mostraremos parte del código resultante para aplicarla a los datos.

La empresa **Cellnex Telecom SA**, con sede en España, proporciona servicios de infraestructura y telecomunicaciones inalámbricas en todo Europa. Sus operaciones se pueden categorizar en cuatro categorías principales: servicios para infraestructuras de telecomunicaciones, redes de difusión audiovisual, servicios de redes de seguridad y emergencia, y soluciones para la gestión inteligente de infraestructuras y servicios urbanos.

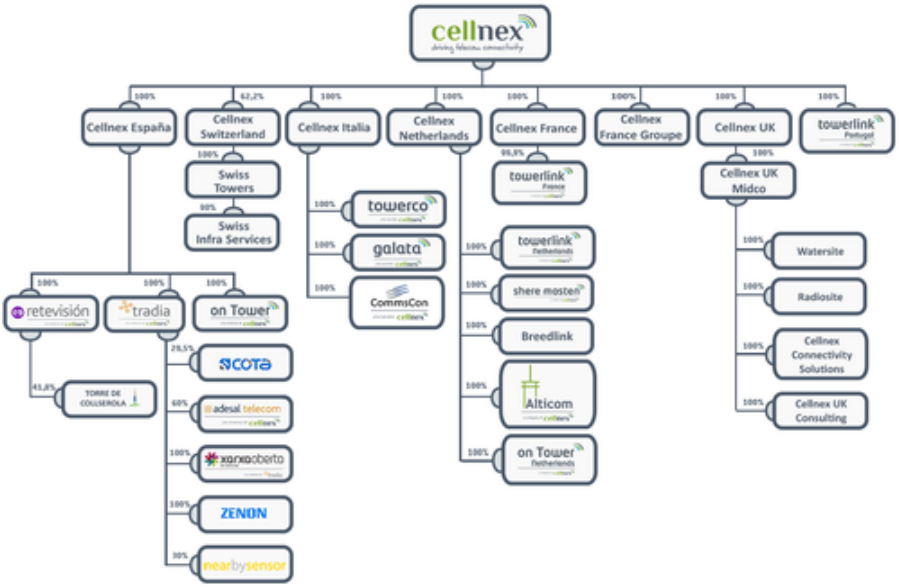


Figura 4.1: Organigrama de actividades de Cellnex

La base de datos está formada por múltiples tablas pero limitándose el enmascaramiento a las siguientes tres: **customer_sap**, **vendor** y **site_alias**.

En este caso, entre los objetivos del proyecto está crear un procedimiento almacenado en Snowflake con lenguaje SQL, principalmente, y JAVA para enmascarar los campos confidenciales o privados en cada una de las tablas.

Para llevarlo a cabo, nos apoyaremos en la la tabla **cap_tech_dim_field_details** que contiene los siguientes campos: ds_view_name, ds_view_field_name, ds_security_classification, audit_load_date, audit_load_job, audit_record_source y audit_last_seen_date.

Esta tabla contiene la información que necesitamos para el enmascaramiento ya que nos mostrará los campos que debemos enmascarar debido a que el campo **ds_security_classification** indicará si el dato es público (ds_security_classification='Public') o privado (ds_security_classification= ´Restricted').

Definimos los campos más importantes para crear el procedimiento:

- ds_view_name: nombre de la tabla o vista.
- ds_view_field_name: campo de la tabla correspondiente.
- ds_security_classification: campo relacionado con la seguridad que puede tomar los valores 'Public' o 'Restricted'. En secciones posteriores, enmascararemos los que sean **'Restricted'**.
- audit_load_date: es la fecha de carga de auditoría.
- audit_load_job: indica el tipo de carga que se realiza,
- audit_record_source: muestra el origen del registro correspondiente.
- audit_last_seen_date: fecha de última actulización o visualización.

Para llevar a cabo el enmascaramiento de nuestros datos hemos utilizado la plataforma de almacenamiento en la nube **Snowflake** aplicando lenguaje SQL.

Como hemos mencionado, la tabla **cap_tech_dim_field_details** contiene la información que debemos enmascarar de cada una de las tablas de nuestra base de datos [1].

cap_tech_dim_field_details

DS_VIEW_NAME	DS_VIEW_FIELD_NAME	DS_SECURITY_CLASSIFICATION	AUDIT_LOAD_DATE	AUDIT_LOAD_JOB	AUDIT_RECORD_SOURCE	AUDIT_LAST_SEEN_DATE
USER_ALIAS	DS_LOGIN_NAME	Public	2023-03-17 10:13:45.684	JOB	SOURCE	2023-03-17 10:13:45.684
USER_ALIAS	DS_LAST_NAME	Public	2023-03-17 10:13:45.684	JOB	SOURCE	2023-03-17 10:13:45.684
USER_ALIAS	DS_ILOQ	Public	2023-03-17 10:13:45.684	JOB	SOURCE	2023-03-17 10:13:45.684
USER_ALIAS	DS_FIRST_NAME	Public	2023-03-17 10:13:45.684	JOB	SOURCE	2023-03-17 10:13:45.684
USER_ALIAS	DS_EMAIL	Restricted	2023-03-17 10:13:45.684	JOB	SOURCE	2023-03-17 10:13:45.684
USER_ALIAS	DS_DISPLAY_NAME	Public	2023-03-17 10:13:45.684	JOB	SOURCE	2023-03-17 10:13:45.684
USER_ALIAS	DS_COMPANY	Public	2023-03-17 10:13:45.684	JOB	SOURCE	2023-03-17 10:13:45.684
USER_ALIAS	DS_CARD_NUMBER	Restricted	2023-03-17 10:13:45.684	JOB	SOURCE	2023-03-17 10:13:45.684
USER_ALIAS	AUDIT_LOAD_DATE	Public	2023-03-17 10:13:45.684	JOB	SOURCE	2023-03-17 10:13:45.684
USER_IAS	DS_EMAIL	Restricted	2023-03-17 10:13:45.550	JOB	SOURCE	2023-03-17 10:13:45.550
USER_IAS	DS_USER_COMPLETE_NAME	Public	2023-03-17 10:13:45.550	JOB	SOURCE	2023-03-17 10:13:45.550
USER_IAS	DS_DELETED	Public	2023-03-17 10:13:45.550	JOB	SOURCE	2023-03-17 10:13:45.550
USER_IAS	DS_COUNTRY_ADDRESS	Public	2023-03-17 10:13:45.550	JOB	SOURCE	2023-03-17 10:13:45.550
USER_IAS	DS_CONTACT_PREFERENCES_TELEPHONE	Public	2023-03-17 10:13:45.550	JOB	SOURCE	2023-03-17 10:13:45.550
USER_IAS	DS_CONTACT_PREFERENCES_EMAIL	Public	2023-03-17 10:13:45.550	JOB	SOURCE	2023-03-17 10:13:45.550
USER_IAS	DS_ACTIVE	Public	2023-03-17 10:13:45.550	JOB	SOURCE	2023-03-17 10:13:45.550
USER_IAS	AUDIT_LOAD_DATE	Public	2023-03-17 10:13:45.551	JOB	SOURCE	2023-03-17 10:13:45.551
VENDOR	DS_NAME_1	Public	2023-03-17 10:13:45.733	JOB	SOURCE	2023-03-17 10:13:45.733
VENDOR	DT_CREATED_AT	Public	2023-03-17 10:13:45.745	JOB	SOURCE	2023-03-17 10:13:45.745
VENDOR	DS_CREATED_BY	Public	2023-03-17 10:13:45.745	JOB	SOURCE	2023-03-17 10:13:45.745
VENDOR	ID_BPARTNER_CAP	Public	2023-03-17 10:13:45.745	JOB	SOURCE	2023-03-17 10:13:45.745
VENDOR	ID_BPARTNER	Public	2023-03-17 10:13:45.745	JOB	SOURCE	2023-03-17 10:13:45.745
VENDOR	ID_VENDOR_CAP	Public	2023-03-17 10:13:45.752	JOB	SOURCE	2023-03-17 10:13:45.752
VENDOR	ID_VENDOR	Public	2023-03-17 10:13:45.752	JOB	SOURCE	2023-03-17 10:13:45.752
VENDOR	ID_REGION	Public	2023-03-17 10:13:45.752	JOB	SOURCE	2023-03-17 10:13:45.752
VENDOR	ID_COUNTRY	Public	2023-03-17 10:13:45.752	JOB	SOURCE	2023-03-17 10:13:45.752
VENDOR	DT_EXECUTION_DATE	Public	2023-03-17 10:13:45.752	JOB	SOURCE	2023-03-17 10:13:45.752
VENDOR	DS_STREET	Restricted	2023-03-17 10:13:45.753	JOB	SOURCE	2023-03-17 10:13:45.753
VENDOR	DS_POSTAL_CODE	Public	2023-03-17 10:13:45.753	JOB	SOURCE	2023-03-17 10:13:45.753

Figura 4.2: Contiene la descripción de seguridad de los campos de las diferentes tablas.

[1] Aquí mostramos solamente una parte de los registros, por eso no aparecen todas las tablas.

Para llevarlo a cabo, solamente tenemos que obtener el nombre del campo y la tabla en la que se va a enmascarar dicho campo. Consideraremos entonces aquellos registros en los cuales ds_security_classification sea **'Restricted'**.

Para conseguir estos campos y estas tablas ejecutamos la siguiente sentencia SQL, donde le indicamos que el campo ds_security_classification toma el valor 'Restricted':

```
1  SELECT * FROM cap_tech_dim_field_details where ds_security_classification=
2  'Restricted';
```

La salida de esta consulta contendrá solamente el nombre de las tablas y los campos correspondientes que se consideran confidenciales en cada una de ellas. Estos campos serán los que tendremos que enmascarar con nuestro código.

En la siguiente sección aplicaremos nuestra política de enmascaramiento específica para la tabla **VENDOR**, ya que en la Figura 4.2, es la única que tiene un campo que debemos enmascarar.

A pesar de que el código completo del procedimiento almacenado se encuentra en el Apéndica A, en este capítulo mostraremos los pasos a seguir para crear la política de enmascaramiento correspondiente para la tabla que nos interesa, cuyo código ya vimos en el capítulo de soluciones del enmascaramiento dinámico.

Introduccimos a continuación el código que utilizaremos en la sección de validación para enmascarar los campos necesarios de la tabla **VENDOR**, que aparecen en la tabla de detalles como privados. Para ello, después de CREATE debemos señalar que se trata de una política de enmascaramiento escribiendo 'masking policy':

```
1  CREATE OR REPLACE masking policy restricted_masking_vendor
2  AS (val string) RETURNS string ->
3  CASE
4  WHEN exists
5  (SELECT 1 FROM cap_dm.cap_dm_security_access a
6  LEFT OUTER JOIN cap_dm.cap_dm_aad_roles b
7  ON a.ID_SECURITY_ACCESS = b.ROLE
8  WHERE ID_SECURITY_ACCESS='G_SNW_MSK_VENDOR'
9  AND USER_ROLE = current_role()) then val
10 WHEN current_role() = 'DEVELOPER' then val
11  else '*********'
12  end;
```

En este caso, el nombre que elegimos para la política de enmascaramiento que se va a implementar en esta tabla es **restricted_masking_vendor** y la máscara que se utiliza para ocultar los datos es '*********'.

Además, solamente los usuarios que tengan otorgado el rol de 'DEVELOPER' podrán ver los datos originales sin enmascar, mientras que el resto de usuarios, tengan el rol que tengan, no conseguirán ver dichos datos.

Notar también que, como los datos de la columna **DS_STREET** son de tipo string, debemos indicarlo entre paréntesis.

4.3. Asignación de roles en Snowflake

En el capítulo anterior hemos mostrado y explicado los diferentes roles que existen en la plataforma Snowflake y como se otorgan a un determinado usuario.

Principalmente, la diferencia entre los diferentes roles que existen son los privilegios que tienen para poder ejecutar sentencias del tipo CREATE, UPDATE, DELETE, DROP, etc. Es decir, consultas más complejas que un SELECT, que sirve para obtener unos registros determinados de la tabla que indiquemos.

Por tanto, con los privilegios básicos solamente podremos realizar la siguiente consulta:[2]

```
1  SELECT * FROM <tabla>
```

Donde en <tabla> pondremos el nombre del objeto que deseemos ver. Mediante esta consulta podemos obtener los registros de tablas, vistas, esquemas o ver los procedimientos almacenados. Podemos denominar **roles primarios** a los que tienen permisos para crear y modificar objetos y roles secundarios a los que no.

Por otra parte, cuando iniciamos sesión en la plataforma de Snowflake, tenemos que acordarnos de indicar el rol que necesitamos utilizar para realizar la acción que queremos llevar a cabo en ese determinado momento. Entonces, para iniciar sesión con el rol que queremos utilizar ejecutamos la siguiente consulta:

```
1  USE ROLE <name>
```

donde sustituiremos el parámetro <name> por el nombre del rol (public, reader, sysadmin, useradmin...)

En determinadas ocasiones, y si nuestro usuario posee más de un rol, necesitamos realizar varias consultas y debemos tener en cuenta que puede ser necesario utilizar distintos roles, dependiendo de la acción que deseemos realizar.

La principal desventaja de esto es que Snowflake solo permite tener un rol activo a la vez.
Esto quiere decir que, para cada consulta que realicemos que requiera un rol diferente al que estamos utilizando en ese momento, debemos ejecutar la sentencia anterior con el rol adecuado para esa determinada acción.

Para terminar la sección, otra observación importante es que si el nombre identificador del rol tiene espacios o algún caracter especial debemos poner el nombre entre comillas dobles al tratarse de un dato de tipo cadena (o string).

[2] El asterisco (*) significa que queremos seleccionar todos los campos de la tabla.

4.4. Validación de los datos

Para verificar la corrección del proceso de enmascaramiento se lleva a cabo un proceso de validación. El test que llevamos a cabo es, en el caso de la tabla **VENDOR**, la comprobación del campo que queremos enmascarar, como podemos ver en la Figura 4.3, es **DS_STREET**, por ser el único que en el campo referente a la seguridad tiene el valor 'Restricted':

DS_VIEW_NAME	DS_VIEW_FIELD_NAME	DS_SECURITY_CLASSIFICATION	AUDIT_LOAD_DATE	AUDIT_LAST_SEEN_DATE
VENDOR	DS_NAME_4	Public	2023-03-17 10:13:45.733	2023-03-17 10:13:45.733
VENDOR	DS_NAME_3	Public	2023-03-17 10:13:45.733	2023-03-17 10:13:45.733
VENDOR	DS_NAME_2	Public	2023-03-17 10:13:45.733	2023-03-17 10:13:45.733
VENDOR	DS_NAME_1	Public	2023-03-17 10:13:45.733	2023-03-17 10:13:45.733
VENDOR	DT_CREATED_AT	Public	2023-03-17 10:13:45.745	2023-03-17 10:13:45.745
VENDOR	DS_CREATED_BY	Public	2023-03-17 10:13:45.745	2023-03-17 10:13:45.745
VENDOR	ID_BPARTNER_CAP	Public	2023-03-17 10:13:45.745	2023-03-17 10:13:45.745
VENDOR	ID_BPARTNER	Public	2023-03-17 10:13:45.745	2023-03-17 10:13:45.745
VENDOR	ID_VENDOR_CAP	Public	2023-03-17 10:13:45.752	2023-03-17 10:13:45.752
VENDOR	ID_VENDOR	Public	2023-03-17 10:13:45.752	2023-03-17 10:13:45.752
VENDOR	ID_REGION	Public	2023-03-17 10:13:45.752	2023-03-17 10:13:45.752
VENDOR	ID_COUNTRY	Public	2023-03-17 10:13:45.752	2023-03-17 10:13:45.752
VENDOR	DT_EXECUTION_DATE	Public	2023-03-17 10:13:45.752	2023-03-17 10:13:45.752
VENDOR	DS_STREET	Restricted	2023-03-17 10:13:45.753	2023-03-17 10:13:45.753
VENDOR	DS_POSTAL_CODE	Public	2023-03-17 10:13:45.753	2023-03-17 10:13:45.753
VENDOR	DS_NAME	Public	2023-03-17 10:13:45.753	2023-03-17 10:13:45.753
VENDOR	DS_CITY	Public	2023-03-17 10:13:45.753	2023-03-17 10:13:45.753
VENDOR	AUDIT_LOAD_DATE	Public	2023-03-17 10:13:45.753	2023-03-17 10:13:45.753

Figura 4.3: En azul el campo a enmascarar de la tabla VENDOR.

La parte de código que ha sido ejecutada en Snowflake para obtener la tabla de la Figura 4.3 fue la siguiente:

```
1  SELECT * FROM cap_tech_dim_field_details where ds_view_name='VENDOR';
```

Donde indicamos que los registros que queremos obtener de la tabla de detalles son aquellos que toman el valor 'VENDOR' en el campo ds_view_name. De esta forma, no nos centramos en el resto de tablas ya que no las necesitamos en este momento.

Una vez obtenemos los campos que queremos enmascarar de cada tabla, debemos crear las políticas de enmascaramiento para cada una de las tablas ya que, como mencionamos en el capítulo anterior, cada política de enmascaramiento es aplicable a una única tabla.

Aunque hay más tablas que requieren el enmascaramiento de algunos campos, nos centraremos especialmente en la tabla **VENDOR** para mostrar un ejemplo de como vería los datos un usuario que tenga el rol 'DEVELOPER' y como los ven el resto de usuarios que no tienen otorgado este rol, que fue el utilizado cuando creamos la política de enmascaramiento correspondiente.

Hacemos ahora una breve descripción de los campos de la tabla que vamos a enmascarar en este apartado:

- ID_VENDOR: índice o número de proveedor.
- DS_COUNTRY: campo que describe el país del proveedor.
- DS_STREET: descripción de la calle.
- DT_CREATED_AT: fecha de creación del pedido.

Antes de hacer cualquier tipo de consulta, incluso antes de ejecutar la sentencia SELECT anterior, debemos identificarnos con el rol que necesitamos para hacer la consulta.

Como el rol que nos interesa en nuestro caso es 'DEVELOPER', ejecutamos el siguiente código:

```
1  USE ROLE DEVELOPER
```

Como mencionamos anteriormente, no basta con crear la política de enmascaramiento para una determinada tabla, ya que también debemos aplicarla.

Mostramos a continuación el código que deberíamos escribir para enmascarar el campo **DS_STREET** de la tabla **VENDOR**:

```
1  ALTER TABLE VENDOR MODIFY COLUMN DS_STREET SET MASKING POLICY
2  restricted_masking_vendor;
```

De esta forma indicamos que la columna que queremos enmascarar es **DS_STREET** de la tabla **VENDOR** utilizando la política de enmascaramiento **restricted_masking_vendor**.

En nuestro caso, como nos identificamos con el rol ´DEVELOPER', veríamos la tabla enmascarada de la siguiente forma[3]:

ID_VENDOR	DS_COUNTRY	DS_STREET	DT_CREATED_AT
188704	Netherlands	[redacted]	2020-06-05 00:00:00.000
252852	France	[redacted]	2022-12-13 00:00:00.000
238143	Poland	[redacted]	2022-05-09 00:00:00.000
255192	France	[redacted]	2022-12-16 00:00:00.000
197267	France	[redacted]	2020-12-20 00:00:00.000
198218	France	[redacted]	2020-12-20 00:00:00.000
500507	Spain	[redacted]	2006-01-04 00:00:00.000
106493	Spain	[redacted]	2006-01-03 00:00:00.000
229807	Italy	[redacted]	2022-03-30 00:00:00.000
229375	Italy	[redacted]	2022-03-30 00:00:00.000

[3]Vemos la tabla con el campo DS_STREET pixelado ya que es confidencial de cara a esta memoria pero el usuario DEVELOPER sí que lo podría ver.

Por el contrario si nos identificamos, por ejemplo, con el rol ´READER':

```
1 USE ROLE READER
```

Y volvemos a ejecutar la consulta:

```
1 SELECT * FROM VENDOR;
```

No tendríamos los permisos necesarios para ver los datos originales y, como ya hemos aplicado la política de enmascaramiento para esta tabla, obtendríamos la siguiente salida:

ID_VENDOR	DS_COUNTRY	DS_STREET	DT_CREATED_AT
175492	France	*********	2018-05-30 00:00:00.000
172850	France	*********	2018-01-22 00:00:00.000
222111	France	*********	2021-12-03 00:00:00.000
259998	France	*********	2022-12-19 00:00:00.000
260145	France	*********	2022-12-19 00:00:00.000
198609	France	*********	2020-12-20 00:00:00.000
179700	France	*********	2018-09-18 00:00:00.000
198113	France	*********	2020-12-20 00:00:00.000
196326	France	*********	2020-12-20 00:00:00.000
187764	France	*********	2020-03-17 00:00:00.000

Donde en este caso vemos que el campo que muestra la descripción de la calle aparece totalmente enmascarado y no tenemos ninguna información ni forma de saber qué es lo que pone.

En resumen, una vez aplicamos la política de enmascaramiento a los datos deseados, solamente los usuarios que tengan el rol 'DEVELOPER' podrán ver los datos originales de la tabla, es decir, sin enmascarar. Por el contrario, si no disponemos de dicho rol, veremos la tabla con el campo confidencial oculto.

Capítulo 5

Conclusiones

A lo largo de este trabajo, hemos adquirido algunos conocimientos sobre el uso de almacenes de datos y cómo se guardan grandes cantidades de información actualmente en almacenes de datos basados en la nube. Estas estructuras permiten una mayor accesibilidad y protección de los datos, a diferencia de cómo se almacenaban anteriormente.

Hoy en día, el enmascaramiento de datos se utiliza en gran medida para proteger datos confidenciales y mantener la seguridad de la información. También hemos examinado las diversas técnicas y métodos de enmascaramiento de datos en este ensayo, junto con sus ventajas e inconvenientes para la aplicación práctica.

El objetivo principal de este trabajo es, en primer lugar, analizar las técnicas de enmascaramiento de datos, en particular el enmascaramiento dinámico. A continuación se demostrará el valor de estas técnicas a través de su aplicación en los datos de una compañía de telecomunicaciones, Cellnex. Para ello, utilizamos la plataforma **Snowflake**, un proveedor de almacenamiento de datos en la nube. Además, para enmascarar los datos que se solicitan, hemos implementado un código SQL, que se incluye en el Apéndice A. Este es el **enmascaramiento dinámico** de datos llevado a cabo mediante un **procedimiento almacenado** que consta de tres políticas de enmascaramiento diferentes (una para cada una de las tablas que queríamos enmascarar) y sentencias CASE que aplican esas políticas. a los campos indicados de las tablas.

Finalmente, como vimos en el capítulo anterior, podemos ver los datos desenmascarados o no dependiendo de los roles y privilegios que tenga nuestro usuario.

Dado que la información es tan valiosa en la sociedad actual, podemos concluir que es fundamental proteger la privacidad de los datos tanto de las personas como de las empresas. Los almacenes de datos en la nube también presentan el beneficio de permitirnos acceder a la información a través de Internet desde cualquier lugar. Sin embargo, esto es algo contraproducente, ya que alguien que no deseamos puede acceder a nuestros datos si nuestra seguridad es insuficiente.

También se han definido otras medidas de seguridad de datos como la tokenización, el cifrado y la eliminación.

A lo largo de este trabajo se ha logrado demostrar los beneficios del enmascaramiento dinámico de datos frente al no dinámico. Además, también hemos visto que ahorra tiempo en disponibilizar la información, si la comparamos con otras técnicas anteriores a esta. Además, hemos podido notar la diferencia entre como se consultaban, guardaban y compartían los datos antiguamente, y como lo hacemos hoy en día.

Mi trabajo de prácticas en la empresa SDG Group ha consistido, principalmente, en el procedimiento almacenado descrito en este trabajo. Aunque no continué trabajando en este proyecto de Cellnex, el equipo que continúa trabajando en esto, ha utilizado mi procedimiento almacenado para enmascarar las tablas que mencionamos en el trabajo y, también, para tablas nuevas. Pues, se ha comprobado que es una manera segura de trabajar con miles de datos confidenciales sin necesidad de mostrarlo, y han querido llevar a cabo el proceso también en otras tablas.

En conclusión, el enmascaramiento de datos es una herramienta fundamental que se utiliza hoy en día para proteger la privacidad de los mismos. Con este método, los datos se pueden seguir utilizando y analizando de forma segura si se aplica el enmascaramiento adecuado.

Apéndice A

Código Empleado para el Enmascaramiento de Datos

En este apéndice mostraremos el código completo que se ha utilizado para realizar el enmascaramiento dinámico de nuestros datos mediante la plataforma Snowflake:

```
1  CREATE OR REPLACE masking policy restricted_masking_customer_sap
2  AS (val string) RETURNS string ->
3  CASE
4  WHEN exists
5  (SELECT 1 FROM cap_dm.cap_dm_security_access a
6  LEFT OUTER JOIN cap_dm.cap_dm_aad_roles b
7  ON a.ID_SECURITY_ACCESS = b.ROLE
8  WHERE ID_SECURITY_ACCESS='G_SNW_MSK_CUSTOMER_SAP'
9  AND USER_ROLE = current_role()) then val
10 WHEN current_role() = 'DEVELOPER' then val
11 else '*********'
12 end;
13
14 CREATE OR REPLACE masking policy restricted_masking_vendor
15 AS (val string) RETURNS string ->
16 CASE
17 WHEN exists
18 (SELECT 1 FROM cap_dm.cap_dm_security_access a
19 LEFT OUTER JOIN cap_dm.cap_dm_aad_roles b
20 ON a.ID_SECURITY_ACCESS = b.ROLE
21 WHERE ID_SECURITY_ACCESS='G_SNW_MSK_VENDOR'
22 AND USER_ROLE = current_role()) then val
23 WHEN current_role() = 'DEVELOPER' then val
24 else '*********'
25 end;
26
27 CREATE OR REPLACE masking policy restricted_masking_site_alias
28 AS (val string) RETURNS string ->
29 CASE
30 WHEN exists
31 (SELECT 1 FROM cap_dm.cap_dm_security_access a
32 LEFT OUTER JOIN cap_dm.cap_dm_aad_roles b
33 ON a.ID_SECURITY_ACCESS = b.ROLE
34 WHERE ID_SECURITY_ACCESS='G_SNW_MSK_SITE_ALIAS'
35 AND USER_ROLE = current_role()) then val
36 WHEN current_role() = 'DEVELOPER' then val
37 else '*********'
38 end;
```

```
39  -- PROCEDURE
40
41  CREATE OR REPLACE procedure MASKING()
42  RETURNS string
43  LANGUAGE javascript
44  EXECUTE AS CALLER
45  AS
46  $$
47  var ent_q = "SELECT distinct ds_view_name FROM cap_tech.cap_tech_dim_field_details
      WHERE ds_security_classification = 'Restricted';"
48  var ent = snowflake.createStatement({sqlText: ent_q}).execute();
49
50  while (ent.next()){
51
52    var desc_q = "DESC VIEW cap_dm." + ent.getColumnValue(1) + ";"
53    var desc = snowflake.createStatement({sqlText: desc_q}).execute();
54
55    var des_q = "SELECT \"name\" FROM table(result_scan(last_query_id())) A INNER JOIN
      cap_tech.cap_tech_dim_field_details B ON A.\"name\" = B.ds_view_field_name   where
      ds_view_name = '" + ent.getColumnValue(1) + "' AND \"policy name\" IS NOT NULL AND
      ds_security_classification = 'Public';"
56    var des = snowflake.createStatement({sqlText: des_q}).execute();
57
58
59    var desc_q = "DESC VIEW cap_dm." + ent.getColumnValue(1) + ";"
60    var desc = snowflake.createStatement({sqlText: desc_q}).execute();
61
62    var enm_q = "SELECT \"name\" FROM table(result_scan(last_query_id())) A INNER JOIN
      cap_tech.cap_tech_dim_field_details B ON A.\"name\" = B.ds_view_field_name where
      ds_view_name = '" + ent.getColumnValue(1) + "' AND \"policy name\" IS NULL AND
      ds_security_classification = 'Restricted';"
63    var enm = snowflake.createStatement({sqlText: enm_q}).execute();
64
65    while(des.next()){
66      var unset_q = "ALTER VIEW cap_dm." + ent.getColumnValue(1) + " MODIFY COLUMN " +
       des.getColumnValue(1)  + " UNSET masking policy;"
67      var unset = snowflake.createStatement({sqlText: unset_q}).execute();
68      unset.next()
69    }
70
71    while(enm.next()){
72      var set_q = "ALTER VIEW cap_dm." + ent.getColumnValue(1) + " MODIFY COLUMN " +
      enm.getColumnValue(1)  + " SET masking policy restricted_masking_" + ent.
      getColumnValue(1) + ";"
73      var set = snowflake.createStatement({sqlText: set_q}).execute();
74      set.next()
75    }
76  }
77  return desc
78  $$;
79
80  CALL MASKING();
```

Para aplicar las políticas de enmascaramiento a las tablas necesarias en el procedimiento almacenado, necesitamos primero crear las políticas de enmascaramiento. Es por eso que, en la página anterior y antes de crear el procedimiento, definimos las tres políticas de enmascaramiendo: restricted_masking_customer_sap, restricted_masking_vendor y restricted_masking_site_alias, que enmascaran los campos especificados de las tablas customer_sap, vendor y site_alias, respectivamente.

Una vez creadas las políticas de enmascaramiento, creamos el procedimiento almacenado con el código CREATE procedure. En este caso, antes de aplicar las políticas de enmascaramiento, le indicamos que seleccione de la tabla cap_tech_dim_field_details los campos a enmascarar de las tablas que

mencionamos en el párrafo anterior.

Finalmente, indicamos que para cada una de esas tablas, se enmascare el campo correspondiente con la política de enmascaramiento adecuada.

Bibliografía

[1] Marco de trabajo basado en ontologías para el proceso ETL. Maestro en Ciencias de la Computación Aplicada, Departamento de Computación, Centro de Investigación y de estudios avanzados del Instituto Politecnico Nacional de México, México, DF; Chávez, J. V. (2011).

[2] El nuevo paradigma tecnológico para empresas y organizaciones en la Sociedad del Conocimiento; Joyanes Aguilar, Luís.

[3] https://doi.org/10.1145/2882903.2903741

[4] https://dl.acm.org/doi/pdf/10.1145/2882903.2903741

[5] Derecho a la privacidad. Protección de los datos sensibles. Revista Colombiana de Bioética.

[6] Qian, L., Luo, Z., Du, Y., Guo, L. (2009). Computación en la nube: una descripción general. En: Jaatun, MG, Zhao, G., Rong, C. (eds) Cloud Computing. CloudCom 2009. Lecture Notes in Computer Science, vol 5931. Springer, Berlín, Heidelberg. https://doi.org/10.1007/978-3-642-10665-1_63

[7] Database Modeling and Design; Toby J. Teorey.

[8] https://repositorio.uta.edu.ec/jspui/handle/123456789/28313

[9] Modelado conceptual e implementación de un proceso de carga de información genómica basado en tecnologías ETL; Martínez Crespín, J. (2016).

[10] https://dl.acm.org/doi/pdf/10.1145/800227.806891

[11] Bases de datos relacionales y modelado de datos; Piñeiro Gómez, José Manuel.

[12] E. F. Codd. A relational model of data for large shared data banks. Commun. ACM, 13:377?387, June 1970.

[13] https://www.snowflake.com/es/

[14] Almacén de datos para el análisis de reglas de asociación (Bachelor's thesis, Universidad de las Ciencias Informáticas. Facultad 3); García Hernández, J. C. (2019).