Universitat Oberta de Catalunya

uoc.edu

Diseño e implementación de la base de datos para una aplicación de control de procesos de seguridad informática

David Giráldez Sánchez
Grado de Ingeniería Informática
TFG - Bases de Datos

Consultor/a: Jordi Ferrer Durán
Profesor/a responsable de la asignatura: Xavier Baró Solé

10/6/2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Diseño e implementación de la base de datos para una aplicación de control de procesos de seguridad informática*
Nombre del autor:	*David Giráldez Sánchez*
Nombre del consultor/a:	*Jordi Ferrer Durán*
Nombre del PRA:	*Xavier Baró Solé*
Fecha de entrega (mm/aaaa):	06/2022
Titulación:	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*Bases de Datos*
Idioma del trabajo:	*Castellano*
Palabras clave	*Gestión de ciberseguridad, base de datos, vulnerabilidad*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

Los objetivos del presente trabajo de fin de grado son aplicar los conocimientos adquiridos durante los estudios del grado de ingeniería informática y, en base a estos, diseñar y desarrollar una base de datos a partir de unos requerimientos iniciales fijos. El sistema permitirá al cliente gestionar de forma ágil toda la información relativa al estado de la ciberseguridad de la empresa, y visualizar los indicadores que ha estimado oportunos en el documento de requerimientos en un tiempo de respuesta conocido.

Se ha utilizado la metodología en cascada para el desarrollo del sistema, ya que partimos de un enunciado cerrado y que no presentará cambios a lo largo del proyecto, y que resulta más adaptado al desarrollo por parte de un equipo unipersonal.

El resultado final cumple los requisitos solicitados, ya que permite la gestión íntegra mediante procedimientos almacenados, sin necesidad de permisos de acceso directo a los datos por parte de los usuarios, y controlando y manteniendo de esta forma la integridad de estos a partir de los requisitos funcionales[4] obtenidos del estudio del enunciado.

Se han utilizado técnicas de datawarehousing para usar la base de datos como repositorio de los datos correspondientes al estado de la ciberseguridad de la empresa, permitiendo consultas que respondan en tiempo constante.

En conclusión, ha resultado un proyecto más complejo y exigente de lo esperado inicialmente, pero enriquecedor personalmente, y me ha permitido obtener unos conocimientos más amplios de lo que supone un proyecto de este tipo.

Abstract (in English, 250 words or less):

This final degree's job's objectives are to apply the acquired knowledge obtained throughout the computer engineering degree studies, and based on them, to develop a database implementation, starting from a given document of requirements. The system will allow the client to manage in an agile way all the information regarding the cybersecurity of the company, along with the possibility to visualize the indicators that the client required in the initial document, all in a known response time.

Cascade methodology has been used for this development, given we start from a closed requirements document that is not going to change throughout this project, and that it's a more suitable methodology for a one person's team development.

The final result meets the requested requirements, since it allows that all the required data management is performed using stored procedures, with no need to assign direct data access permissions to users and forcing to fulfill all of the integrity requirements obtained in the initial analysis by the use of the system’s incorporated procedures.

Data Warehousing techniques have been used to provide access to the repository data, allowing constant time responses.

In conclusion, it has been a more complex and demanding project than it was initially expected, but it has also been personally fulfilling, and it has allowed me to obtain more in-depth knowledge about this kind of projects.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

El objetivo del presente trabajo de fin de grado es diseñar e implementar un producto para un cliente, que cumpla con las especificaciones recogidas en el documento aportado inicialmente. Será necesario desarrollar una base de datos para el control del estado de ciberseguridad de la empresa, unificando en todos los procesos de gestión de la información relativa a la misma, presentando, procesando y permitiendo la explotación de dicha información con propósitos estadísticos.

La gestión (tanto la detección, lo más temprana posible, como la corrección, en lo posible) de las vulnerabilidades en los sistemas informáticos en la actualidad se ha convertido en una parte fundamental de los procesos de seguridad de toda empresa, la ciberseguridad.

Para gestionar la información sobre las vulnerabilidades descubiertas y su número, los procesos de la empresa a los que afectan, así como el estado de las correcciones recomendadas por los expertos y de las políticas de prevención definidas por la empresa, vamos a diseñar, a petición de la empresa cliente, un modelo de datos que permita tanto el almacenamiento como la explotación de dicha información. Para ello, se utilizarán técnicas de datawarehousing. Además, el sistema deberá soportar la gestión de cualquier volumen de datos.

Del mismo modo, la formación del personal para evitar en lo posible ciberataques, tanto los más técnicos como los más simples (mediante técnicas de phishing es posible obtener mucha información de una empresa sin tener conocimientos informáticos especiales) ha pasado a ser una de las preocupaciones de toda empresa. El modelo de datos que vamos a diseñar permitirá también almacenar la información sobre la formación impartida por la empresa, la recibida por cada empleado, y adicionalmente, el resultado de las simulaciones de ataques que, de forma preventiva, realizará la propia empresa para comprobar el grado de cumplimiento de las políticas de seguridad (definidas por la empresa) por parte de los empleados.

La gestión de todo el sistema se llevará a cabo (a falta de una aplicación de usuario que lo haga) mediante procedimientos almacenados PL/SQL[6], controlando de esa manera la corrección semántica de los datos de entrada. También será necesario implementar

mecanismos de seguridad que permitan realizar un seguimiento de qué usuarios lanzan qué procesos.

1.2 Objetivos del Trabajo

El presente trabajo está orientado a plantear una posible solución para gestionar la información relativa a la seguridad informática de los sistemas de una empresa, almacenando además toda la información relativa a la formación impartida, las políticas de seguridad definidas, así como su cumplimiento.

Además, planteará un modelo de datos que permitirá almacenar toda la información que se ha considerado relevante, de forma que permita realizar consultas y obtener resultados sin necesidad de acceder a diferentes almacenes de datos, y en tiempo constante 1.

Otro requisito que deberá cumplir el sistema es el control y seguimiento de los diferentes accesos, para cumplir con las leyes de protección de datos existentes. Para ello, el sistema será gestionado exclusivamente a través de los procedimientos programados incluidos, cuyas ejecuciones se registrarán en un log del sistema, junto con el usuario que lo ejecuta y la fecha, no permitiendo ni necesitando el acceso directo a los datos.

El diseño deberá ser adaptable para incorporar posibles funcionalidades que el cliente solicitase en el futuro.

1.3 Enfoque y metodología

Dado que partimos de un enunciado cerrado (aunque se tiene la opción de pedir aclaraciones al consultor, que actuará como "cliente"), y que mi experiencia tanto en diseño como en programación es nula, he optado por desarrollar el producto solicitado siguiendo una metodología en cascada, definiendo varias fases, cada una de las cuales deberá estar finalizada antes de pasar a la siguiente. Esto obliga a que el resultado de una fase sea lo más acertado posible, para no tener que volver atrás en el desarrollo.

He considerado que son necesarias las siguientes fases:

1. Una primera fase, de **análisis y obtención de requisitos**, a partir del análisis del enunciado proporcionado. Finalizamos esta fase disponiendo de un listado de requisitos, tanto funcionales como no funcionales, que la aplicación debe cumplir.

2. Posteriormente pasamos a la fase de **diseño**: trataremos de plasmar los requisitos obtenidos a partir del análisis realizado en la primera fase, primero en el diseño conceptual, del que

posteriormente obtendremos un diseño lógico, en el cual basaremos finalmente la implementación del modelo físico sobre el SGBD elegido.

3. Una vez definido el modelo de datos, procederemos a la **implementación** sobre el SGBD elegido (en este caso, Oracle 12.2 [1]) del mismo. Se crearán las estructuras físicas (datafiles) y lógicas (tablespaces) necesarias para almacenar los datos (tablas) y demás objetos (procedimientos, disparadores, etc.) del usuario (esquema).

4. Con el producto en su versión final beta, se realizarán diferentes **pruebas** para validar que el resultado obtenido es el deseado, y se realizarán las correcciones del código que se estimen necesarias.

5. Finalizaremos con la redacción / corrección / actualización de la **documentación** que se ha ido generando a lo largo de todo el proyecto.

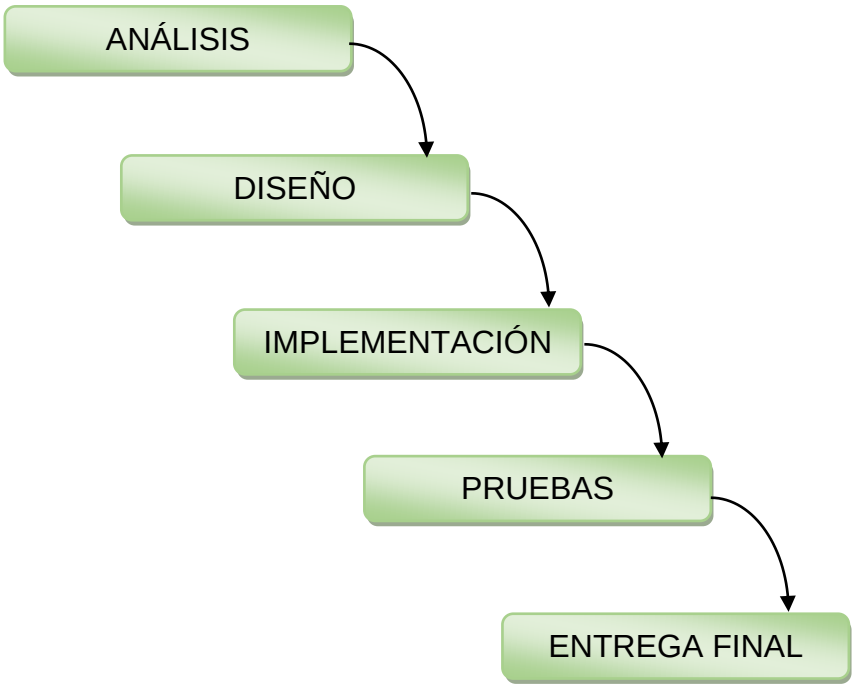
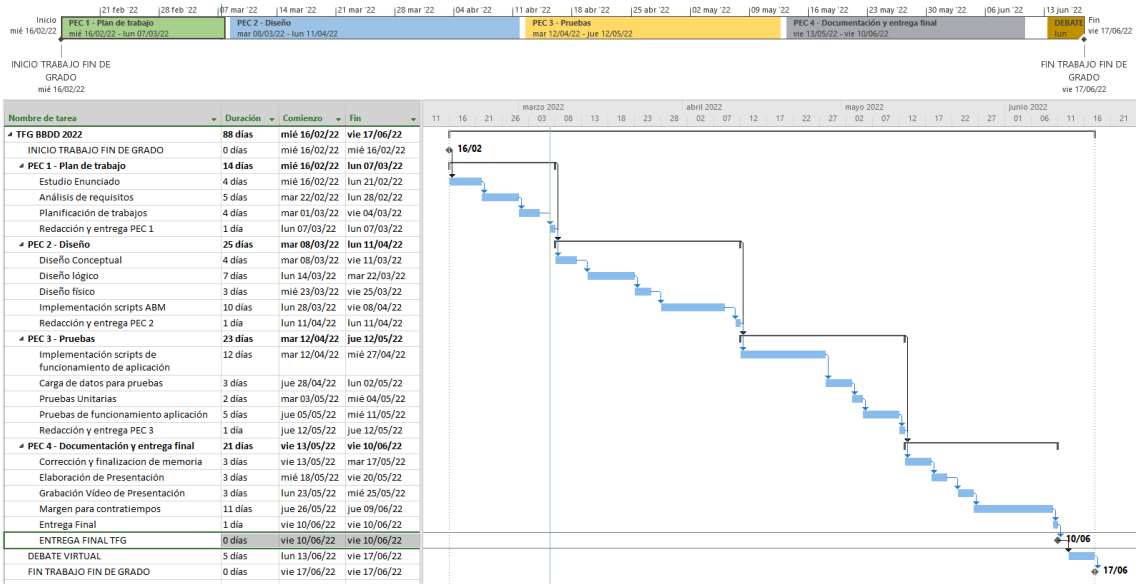


Ilustración 1 Fases

1.4 Planificación del Trabajo

He realizado la planificación teniendo en cuenta pasadas experiencias, por lo que he planificado un trabajo diario entre semana, dejando los fines de semana para recuperar retrasos puntuales, así como un periodo de dos semanas al final para recuperar problemas más graves que se puedan dar a lo largo del semestre, o correcciones importantes que se detecten ya avanzado el mismo.



Nombre de tarea	Duración	Comienzo	Fin
TFG BBDD 2022	**88 días**	**mié 16/02/22**	**vie 17/06/22**
INICIO TRABAJO FIN DE GRADO	0 días	mié 16/02/22	mié 16/02/22
PEC 1 - Plan de trabajo	**14 días**	**mié 16/02/22**	**lun 07/03/22**
Estudio Enunciado	4 días	mié 16/02/22	lun 21/02/22
Análisis de requisitos	5 días	mar 22/02/22	lun 28/02/22
Planificación de trabajos	4 días	mar 01/03/22	vie 04/03/22
Redacción y entrega PEC 1	1 día	lun 07/03/22	lun 07/03/22
PEC 2 - Diseño	**25 días**	**mar 08/03/22**	**lun 11/04/22**
Diseño Conceptual	4 días	mar 08/03/22	vie 11/03/22
Diseño lógico	7 días	lun 14/03/22	mar 22/03/22
Diseño físico	3 días	mié 23/03/22	vie 25/03/22
Implementación scripts ABM	10 días	lun 28/03/22	vie 08/04/22
Redacción y entrega PEC 2	1 día	lun 11/04/22	lun 11/04/22
PEC 3 - Pruebas	**23 días**	**mar 12/04/22**	**jue 12/05/22**
Implementación scripts de funcionamiento de aplicación	12 días	mar 12/04/22	mié 27/04/22
Carga de datos para pruebas	3 días	jue 28/04/22	lun 02/05/22
Pruebas Unitarias	2 días	mar 03/05/22	mié 04/05/22
Pruebas de funcionamiento aplicación	5 días	jue 05/05/22	mié 11/05/22
Redacción y entrega PEC 3	1 día	jue 12/05/22	jue 12/05/22
PEC 4 - Documentación y entrega final	**21 días**	**vie 13/05/22**	**vie 10/06/22**
Corrección y finalizacion de memoria	3 días	vie 13/05/22	mar 17/05/22
Elaboración de Presentación	3 días	mié 18/05/22	vie 20/05/22
Grabación Video de Presentación	3 días	lun 23/05/22	mié 25/05/22
Margen para contratiempos	11 días	jue 26/05/22	jue 09/06/22
Entrega Final	1 día	vie 10/06/22	vie 10/06/22
ENTREGA FINAL TFG	0 días	vie 10/06/22	vie 10/06/22
DEBATE VIRTUAL	5 días	lun 13/06/22	vie 17/06/22
FIN TRABAJO FIN DE GRADO	0 días	vie 17/06/22	vie 17/06/22

Ilustración 2 Diagrama de Gantt

Con la evolución del proyecto ha sido finalmente necesario hacer uso del periodo de dos semanas final por ajustes necesarios del modelo que han sido detectados en la última fase.

1.5 Breve sumario de productos obtenidos

El producto final de este proyecto será una base de datos que permita la gestión y el control de los procesos de seguridad informática del cliente.

La entrega se compondrá de:

- Scripts de creación de la base de datos: tablas, procedimientos almacenados, roles y usuarios, pruebas de los procedimientos del sistema, y un juego de datos de prueba del sistema.
- Memoria del proyecto.
- Video con la presentación de este.
- Informe de autoevaluación.

1.6 Breve descripción de los otros capítulos de la memoria

En los siguientes capítulos desarrollaremos las diferentes fases del proyecto:

- Análisis del enunciado y obtención de requisitos: a partir de la lectura del enunciado del proyecto, se obtiene un listado de los requisitos que deberá cumplir la aplicación a desarrollar.
- Diseño:
  - Conceptual: identificaremos las entidades y sus atributos, así como las relaciones entre ellas.
  - Lógico: convertiremos el modelo conceptual previamente obtenido en un modelo relacional.
- Implementación del diseño físico: transformaremos el modelo lógico de la fase anterior en un modelo que podremos implementar en el SGBD elegido. Distinguiremos los siguientes apartados:
  - Seguridad. Roles. Usuarios.
  - Base de datos. Esquema.
  - Tablas
  - Procedimientos almacenados. Paquetes.
  - Secuencias y disparadores.
  - Módulo estadístico.
- Pruebas de validación del producto: definiremos un plan de pruebas para comprobar el correcto funcionamiento de cada uno de los procedimientos generados (pruebas unitarias), así como del sistema general.

En los últimos capítulos desarrollaremos las conclusiones a las que hemos llegado tras la ejecución del proyecto, un glosario de los principales términos técnicos utilizados, así como un listado de la documentación consultada para la elaboración del proyecto.

2. Análisis del enunciado. Requisitos.

En esta primera fase del proyecto, el objetivo es obtener una idea clara de los requerimientos del sistema a desarrollar, a partir del enunciado proporcionado, y de la interpretación realizada por mí. Podemos destacar la importancia de esta primera fase para minimizar la posibilidad de tener que retocar el diseño y volver a atrás en el proyecto.

Tras un estudio reiterado del enunciado, he tomado un número de decisiones que han determinado los requisitos funcionales[4] que deberá cumplir el proyecto, así como el diseño final obtenido.

2.1 Análisis del enunciado.

El sistema que diseñemos deberá almacenar toda la información que consideremos de interés relativa a la ciberseguridad de la empresa. En concreto, por un lado, almacenaremos la información relativa a todas las **vulnerabilidades** que se descubran en los **procesos informáticos** de la empresa durante los **análisis** realizados, y de las **acciones de mitigación** que se recomienden para corregirlas. Por otro lado, se gestionará la información relativa a las **políticas de seguridad** que se definan por parte de la empresa, de los **incumplimientos** de estas que se descubran en las **auditorías** que se realicen, y de las **sesiones de formación** al respecto que se impartan.

He considerado que la información exhaustiva de los diferentes elementos (tales como descripciones completas de vulnerabilidades, planes de ejecución de acciones de mitigación, enunciado completo de políticas de seguridad, documentación de los análisis y auditorías realizados, etc.) no será incluida en base de datos. Para cada elemento que lo requiera, se almacenará la URL del documento relevante alojado en el gestor documental de la empresa, que queda fuera del objetivo de este proyecto.

Con respecto a las **vulnerabilidades** que se descubran, además de asignarles un identificador único, almacenaremos el **análisis** que la ha detectado, la fecha de detección, el **estado** (que irá cambiando en el tiempo, y en principio se ha definido que podrá ser uno de los siguientes: *Identificada, No Mitigada, Parcialmente Mitigada* – si hay al menos una acción de mitigación asociada a la vulnerabilidad que está en estado acabada - o *Totalmente Mitigada* – si todas las acciones de mitigación asociadas a la vulnerabilidad están acabadas -), si es considerada **crítica** o no, y el **proceso** de la empresa sobre el que se ha detectado la vulnerabilidad. Además, he considerado oportuno registrar un nombre descriptivo de la vulnerabilidad, así como el mencionado enlace a la documentación completa de la vulnerabilidad en el gestor documental de la empresa.

Si bien una misma *vulnerabilidad informática* podrá afectar a uno o varios procesos de la empresa, a efectos de nuestro sistema las

consideraríamos vulnerabilidades diferentes, ya que toda la información relativa que nos interesa almacenar puede ser distinta (tanto su estado como su criticidad podrán ser diferentes para los distintos procesos, podrán afectar a diferentes procesos y ser descubiertas en distintos análisis y fechas, etc.). Del mismo modo, las **acciones de mitigación** definidas para una misma vulnerabilidad descubierta en diferentes procesos podrían ser diferentes.

Para cada una de las vulnerabilidades descubiertas se definirán **acciones de mitigación proactiva** de las mismas (podrán ser una o varias acciones las que estén asociadas a una misma vulnerabilidad), con el objetivo de corregir dichas vulnerabilidades. Para dichas acciones les asignaremos un identificador único, y almacenaremos un nombre identificativo, así como el empleado designado como responsable de coordinar la ejecución de la acción, la fecha límite que la empresa se plantea para la implantación, el **estado** en que se encuentren (que en el enunciado se ha definido que podrá ser uno de los siguientes: *Definida, En proceso, Acabada,* o *En Revisión* – para aquellas que no se pueda ejecutar por algún motivo), la fecha en que se alcanzó dicho estado, y la URL relativa a la documentación completa de la acción de mitigación.

En cada momento deberá ser posible conocer las acciones de mitigación que están en ejecución para aquellas vulnerabilidades que aún no estén totalmente mitigadas, y así conocer el estado de resolución de las vulnerabilidades identificadas.

Con respecto a los **análisis** en busca de vulnerabilidades en los procesos, almacenaremos su identificador único, la indicación de si se trata de un análisis externo o interno, el nombre del equipo que lo ha realizado, así como las fechas de inicio y finalización (que normalmente no se conocerá al comienzo del análisis). También le asignaremos y almacenaremos un nombre identificativo, y registraremos la URL de la documentación relativa al análisis en el gestor documental de la empresa. Además, deberemos poder obtener el listado de las vulnerabilidades descubiertas por cada análisis, y de los procesos de la empresa analizados.

Respecto a los **procesos internos de la empresa**, almacenaremos solamente un identificador único, una breve descripción, la URL de la documentación completa del mismo, y si han sido analizados o no en algún momento. Además, almacenaremos por separado los diferentes análisis y fechas en los que se haya **estudiado** el proceso.

Por otro lado, por parte de la empresa se definirán unas **políticas de seguridad**, que serán de obligado cumplimiento por parte de todos los empleados y departamentos. De las políticas registraremos un identificador, nombre, y la URL donde podemos encontrar toda la documentación al respecto. También registraremos el número de incumplimientos de la política que se han detectado, actualizando el

valor de forma automática por parte del sistema para la gestión de las consultas estadísticas.

Será necesario poder controlar desde la aplicación los **incumplimientos** de dichas políticas. Se registrarán en el sistema los incumplimientos (identificador) y su fecha de detección, así como el empleado que lo ha cometido, la política incumplida, y la auditoría que lo ha descubierto. Tras cada inserción, se actualizarán los campos estadísticos afectados de forma automática mediante triggers.

En el enunciado se indica que el incumplimiento de las políticas conllevará consecuencias administrativas por parte del departamento de recursos humanos para las personas implicadas. Sin embargo, como se indica igualmente en el enunciado, en nuestra aplicación solo almacenaremos el número total de incumplimientos por política por empleado y en cada departamento de la empresa, y no se almacenarán las sanciones que se les hayan aplicado a los empleados, quedando esa información en manos del departamento de RRHH y fuera de nuestra aplicación.

Como tan importante es detectar las vulnerabilidades existentes en los procesos de la empresa como concienciar y formar a los empleados de ésta para que conozcan y eviten las que puedan originarse por su lado – mediante técnicas de phishing, por ejemplo – la empresa realizará tanto **sesiones de formación** como simulaciones controladas de ataques, de forma que se descubra a aquellos empleados que no cumplan las políticas de seguridad definidas.

De nuevo, en nuestra base de datos solo registraremos la cantidad de dichos incumplimientos, anotándoselos al empleado y al departamento al que pertenezca, y dejando en manos del departamento de recursos humanos (fuera de nuestro proyecto) la penalización que el incumplimiento suponga para el empleado infractor.

Con respecto a las **sesiones de formación**, deberemos registrar todas las que se impartan en la empresa en relación con los temas de seguridad que nos ocupan, indicando su título, la modalidad (*presencial* o *telemática*), su fecha de inicio, los **empleados** que participen en ellas y su grado de finalización.

También deberemos registrar en el sistema la información relativa a las **auditorías** que se realicen, y todos los **muestreos** que dichas auditorías realicen. Los muestreos irán asociados a la política de seguridad que se esté auditando, a la auditoría que lo realiza, y a los empleados que son auditados; además, conservaremos para los muestreos realizados la fecha en que se ejecutaron. No se almacenará el resultado del muestreo, debido a las restricciones de confidencialidad que se le exigen al nuevo sistema.

Para poder recuperar la información relativa a los **empleados**, así como para la relativa a los **departamentos** de la empresa, será

necesario almacenar la información de todos ellos. No considero necesario guardar de los empleados más que su número de empleado (identificador único), nombre y apellidos, departamento al que pertenece, email y teléfono de contacto, así como el dato de si sigue o no perteneciendo a la empresa, y el número de incumplimientos que haya cometido. He considerado que los posibles cambios de departamento, por infrecuentes, no los vamos a tener en cuenta y no conservaremos el histórico, por lo que solo guardaremos el departamento al que pertenece el empleado en la actualidad. En cuanto a los departamentos, con un identificador y un nombre tendremos suficiente.

Para poder controlar la seguridad de la gestión de la aplicación, toda la gestión de la información se llevará a cabo mediante procedimientos almacenados, sin permitir la manipulación directa de los datos, de forma que controlemos la calidad de los datos almacenados. De esta manera, solo los usuarios de base de datos que tengan permisos de ejecución sobre los paquetes podrán manipular los datos.

Además, todas las ejecuciones de los diferentes procedimientos se registrarán en un **log de operaciones**, que guardará la fecha de ejecución, el usuario de base de datos que lo ha ejecutado, el nombre del procedimiento ejecutado, los parámetros de entrada del procedimiento, y una indicación de si el resultado de la ejecución ha sido correcto o no (indicando en este caso el error producido).

Finalmente, con el objetivo de controlar de manera rápida la situación de la seguridad de los sistemas, por parte del cliente se han definido un conjunto de indicadores que permitan conocer en cada momento el estado real del mismo. Para ello, se definirán unas tablas estadísticas que almacenarán los datos que nos interesan, que serán actualizados de forma automática mediante disparadores y procedimientos. Esto nos permitirá obtener la información que nos interese en un tiempo constante 1, sin necesidad de realizar consultas complejas sobre la información almacenada y esperar el resultado.

2.2 Requisitos funcionales.

Los requisitos funcionales[4] reflejan funciones qué deberá cumplir el sistema para dar respuesta a las necesidades indicadas en el enunciado. En base al análisis indicado anteriormente, indicamos los siguientes requisitos funcionales[4] para el sistema a desarrollar:

1. Se almacenará toda la información que se considere relevante para la empresa. En concreto, la relativa a las entidades:

    a. Análisis realizados, en busca de vulnerabilidades.
    b. Vulnerabilidades descubiertas en los análisis.
    c. Acciones de mitigación proactiva definidas de las vulnerabilidades.

d. Procesos informáticos de la empresa.
e. Políticas de seguridad definidas por la empresa.
f. Auditorías realizadas para comprobar el grado de cumplimiento de las políticas definidas.
g. Incumplimientos de dichas políticas.
h. Sesiones de formación impartidas.
i. Empleados de la empresa.
j. Departamentos de la empresa.

2. Será necesario reflejar las relaciones entre las entidades definidas, y en su caso, los datos asociados a dichas relaciones. En concreto:
    a. Para cada análisis realizado, las vulnerabilidades descubiertas.
    b. Para cada análisis realizado, los procesos de empresa que ha estudiado.
    c. Para cada vulnerabilidad, el proceso de empresa al que afecta.
    d. Para cada vulnerabilidad, las acciones de mitigación definidas para remediarla.
    e. Para cada acción de mitigación, el empleado que es responsable de su implantación.
    f. Para cada política de seguridad definida, los incumplimientos detectados.
    g. Para cada política de seguridad definida, los muestreos realizados.
    h. Para cada auditoría de seguridad llevada a cabo, las políticas de seguridad sobre las que se ha llevado a cabo.
    i. Para cada auditoría realizada, los incumplimientos de políticas detectados.
    j. Para cada auditoría, los muestreos realizados, por cada política y empleado.
    k. Para cada sesión formativa impartida, los empleados a los que se les ha asignado.
    l. Para cada departamento, todos los empleados que pertenecen al mismo.

2.3 Requisitos no funcionales.

Como requisitos no funcionales[5] señalamos los siguientes:

a. La base de datos será relacional.
b. Como SGBD se utilizará Oracle 12.2 [1]
c. El sistema será escalable a cualquier volumen de datos, y deberá permitir incorporar modificaciones y nuevas funcionalidades.
d. El único método permitido de acceso al sistema serán los procedimientos almacenados que se implementen.

e. El sistema incluirá un repositorio estadístico que deberá devolver los resultados de las consultas solicitadas por el cliente en tiempo constante 1.
f. El sistema almacenará información relativa a la ejecución de los procedimientos mediante una tabla de log, en la que se registrará toda la información que se considere relevante, incluyendo el resultado de la ejecución y el usuario que lo ha ejecutado.
g. Todos los procedimientos almacenados implementados incluirán el tratamiento de excepciones.

2.4 Restricciones de integridad.

Las restricciones de integridad serán en algunos casos implementadas en la propia definición de las tablas afectadas, y en otros casos se comprobarán durante la ejecución de los procedimientos involucrados.

1. Los estados por los que puede pasar una **vulnerabilidad** son los siguientes (podrían añadirse más estados con el tiempo):
   - *Identificada*
   - *No Mitigada*
   - *Parcialmente Mitigada* (si hay al menos una acción de mitigación asociada a la vulnerabilidad que está en estado acabada)
   - *Totalmente Mitigada* (si todas las acciones de mitigación asociadas a la vulnerabilidad están acabadas)
2. La fecha de detección de una **vulnerabilidad** siempre será anterior o igual a la fecha actual.
3. La fecha de detección de una **vulnerabilidad** deberá ser siempre posterior o igual a la fecha de inicio del análisis en el que se detecta, y anterior o igual a la fecha de fin de dicho análisis (si está registrada en el sistema).
4. El indicador de si una **vulnerabilidad** es crítica o no será de tipo booleano: 0/1, S/N
5. Los estados por los que puede pasar una **acción** de mitigación son (podrían añadirse más estados con el tiempo, pero de momento son estos los que se indican en el enunciado):
   - *Definida*
   - *En proceso*
   - *Acabada*
   - *En Revisión* (para aquellas que no se pueda ejecutar por algún motivo)
6. La fecha límite de implantación de una **acción** de mitigación siempre será mayor que la fecha actual.
7. La fecha en la que una **acción** cambia de estado siempre será anterior o igual a la fecha actual.
8. El responsable de una **acción** deberá ser un empleado en activo.

9. La fecha de **estudio** de un **proceso** durante un análisis de vulnerabilidades siempre será mayor o igual a la fecha de inicio del análisis, y menor o igual a la fecha de fin del análisis (si está registrada).
10. El indicador de si un **proceso** ha sido analizado o no será de tipo booleano: 0/1, S/N
11. Los **análisis** podrán ser de tipo interno o externo.
12. La fecha de inicio de un **análisis** siempre será menor que su fecha de fin. Solo se comprobará si la fecha de fin está registrada.
13. La fecha de detección de un **incumplimiento** debe ser anterior o igual a la fecha actual.
14. La fecha de detección de un **incumplimiento** debe estar entre las fechas de inicio y de finalización de la auditoría en la que se ha detectado.
15. Las modalidades de las **sesiones** formativas podrán ser *presencial* o *telemática*.
16. El indicador de si un **empleado** ha finalizado o no una **sesión** de formación asignada será de tipo booleano: 0/1, S/N
17. La fecha de fin de una **sesión** por parte de un empleado siempre será posterior a la fecha de inicio de dicha sesión formativa.
18. Si se registra la fecha de finalización de una **sesión** de formación asignada a un empleado, se debe reflejar también que ha finalizado la sesión (valor 1, S, …)
19. El indicador de si un **empleado** sigue o no en activo en la empresa será de tipo booleano: 0/1, S/N
20. El número de **incumplimientos** de un empleado siempre será mayor o igual que 0.
21. El número de **incumplimientos** de una política siempre será mayor o igual que 0.
22. La fecha de inicio de una **auditoría** siempre será menor que su fecha de fin, si es que esta está registrada.
23. La fecha de realización de un **muestreo** debe estar entre las fechas de inicio y de finalización (si está registrada) de la auditoría durante la que se ha realizado.

3. Diseño.

3.1 Diseño Conceptual.

El primer paso para obtener un modelo de datos que sea consistente con los requisitos funcionales[4] que hemos recogido en el análisis del enunciado es realizar un diseño conceptual adecuado, en el que queden reflejadas las entidades que hemos identificado anteriormente, y las relaciones que existan entre ellas.

A partir de las decisiones indicadas y los requerimientos anotados, hemos llegado al siguiente modelo E/R:

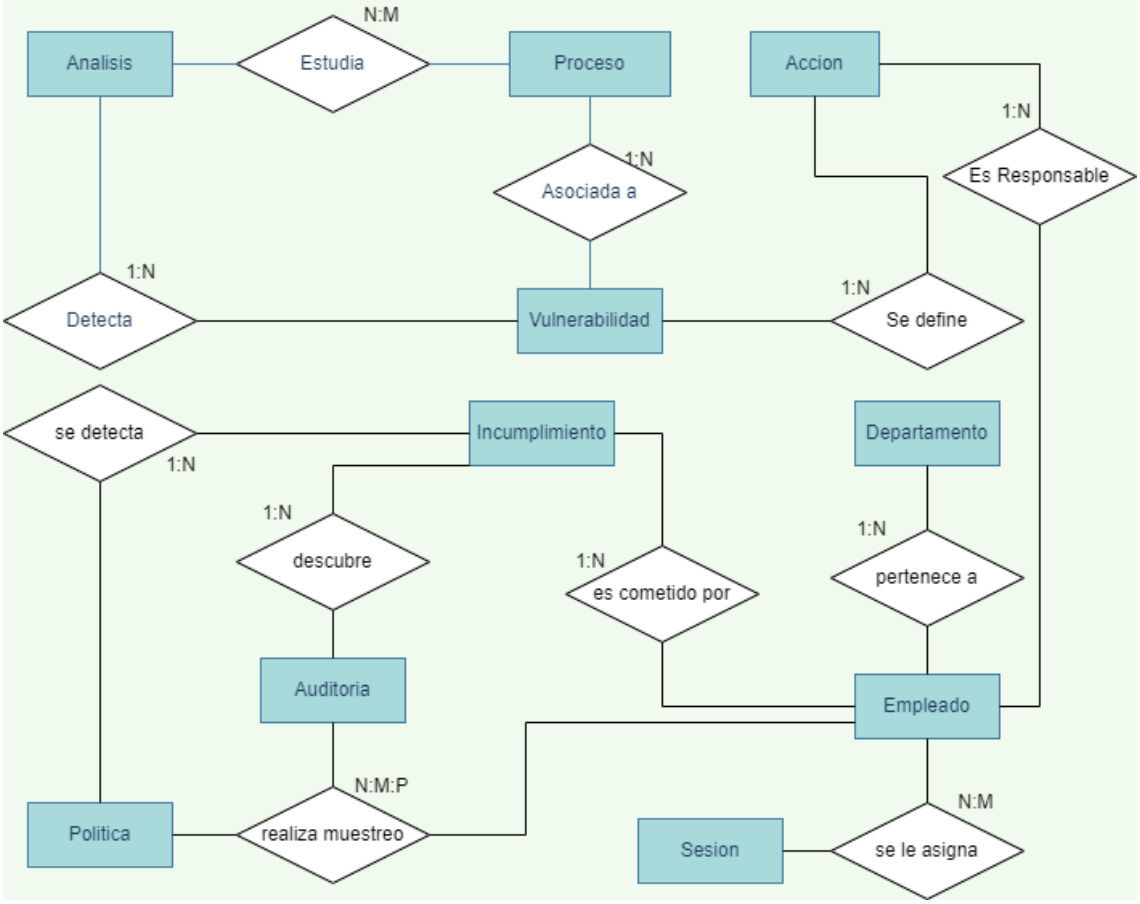


Ilustración 3 Diseño Conceptual

3.2 Diseño Lógico.

A partir del diseño conceptual obtenido previamente, elaboramos el siguiente modelo lógico, ajustado al modelo relacional que hemos elegido como nuestro SGBD.

Con respecto a las relaciones reflejadas en el diseño conceptual, realizaremos las siguientes conversiones:

- Las relaciones binarias con cardinalidad 1:N se resuelven añadiendo como un nuevo atributo a la entidad con multiplicidad N la clave primaria de la entidad con multiplicidad 1 en la relación. Será una clave foránea (FK).
- Las relaciones binarias con cardinalidad N:M se convierten en nuevas entidades, que tendrán como clave primaria la unión de las claves primarias de las dos entidades relacionadas. Además, los atributos de la PK serán FK.
- Las relaciones ternarias con cardinalidad N:M:P se convierten en nuevas entidades, cuya clave primaria será la unión de las claves primarias de las entidades relacionadas. Del mismo modo, serán también claves foráneas.

Para representar el diseño lógico, utilizaremos la notación:

- Los atributos que formen las claves primarias irán subrayados con línea continua.
- Los atributos que formen las claves candidatas (en caso de existir) irán subrayadas con línea discontinua.
- Los atributos que no sean parte de ninguna clave (primaria o candidata) y para los que no admitiremos valores nulos se marcarán en negrita.
- Los atributos que sean claves foráneas se marcarán entre llaves ({}), indicando a qué tabla hacen referencia.

Con todo esto, llegamos al siguiente modelo lógico:

Analisis (id_analisis, **nombre_analisis**, **tipo_analisis**, **nombre_equipo**, **f_inicio_analisis**, f_fin_analisis, url_analisis)

Proceso (id_proceso, **nombre_proceso**, url_proceso, **ha_sido_analizado_proceso, num_vuln_detectadas**)

EstudioProceso (id_analisis, id_proceso, **f_estudio**)
{id_analisis} es FK de Analisis
{id_proceso} es FK de Proceso

Vulnerabilidad (id_vulnerabilidad, **id_analisis, id_proceso, f_deteccion_vuln, id_estado_vuln, es_vuln_critica, nombre_vuln,** url_vuln)
{id_analisis} es FK de Análisis

{id_proceso} es FK de Proceso
{id_estado_vuln} es FK de EstadoVulnerabilidad

EstadoVulnerabilidad (id_estado_vuln, **estado_vuln**)

Accion (id_accion, **id_vulnerabilidad**, **nombre_accion**, **responsable**, **f_implantacion_acc, id_estado_acc**, **f_estado_acc**, url_accion)
{id_vulnerabilidad} es FK de Vulnerabilidad
{responsable} es FK de Empleado
{id_estado_acc} es FK de EstadoAccion

EstadoAccion (id_estado_acc, **descripción_estado_accion**)

Politica (id_politica, **nombre_politica**, url_politica, **num_incumplimientos**)

Incumplimiento (id_incumplimiento, **id_politica**, **id_auditoria**, **id_empleado**, **f_deteccion_incumplimiento**)
{id_politica} es FK de Politica
{id_auditoria} es FK de Auditoria
{id_empleado} es FK de Empleado

Sesion (id_sesion, **titulo_sesion**, **modalidad_sesion**, **f_inicio_sesion**)

AsignadaSesion(sesión_id_sesion, empleado_id_empleado, **ha_finalizado_sesion**, f_fin_sesion)
{empleado_id_empleado} es FK de Empleado
{sesión_id_sesion} es FK de SesionFormacion

Auditoria (id_auditoria, **nombre_auditoria**, **tipo_auditoria, f_inicio_auditoria**, f_fin_auditoria)

Muestreo (id_auditoria, id_politica, id_empleado**, f_muestreo**)
{id_auditoria} es FK de Auditoria
{id_politica} es FK de Politica
{id_empleado} es FK de Empleado

Empleado (id_empleado, **dni_empleado**, **nombre_emp**, **apellido1_emp**, **apellido2_emp**, **id_departamento**, **email_empleado**, **tfno_empleado**, **es_empleado_en_activo**, **incumplimientos_emp**)
{id_departamento} es FK de Departamento

Departamento (id_departamento, **nombre_depto, incumplimientos_empleados, empls_con_sesiones_pendientes**)

Además, se han definido una entidad en la que recogeremos el histórico de ejecuciones de los procedimientos en el sistema, y otras tres que recogerán la información intermedia que será utilizada por el módulo

estadístico para lograr el tiempo constante 1 en las consultas estadísticas:

LogOperaciones(id_log, **f_ejecucion, usuario, nombre_procedimiento, params_entrada, resultado**)
Estadistica(id_consulta, **descripción_consulta, resultado_consulta**)

IncumplimientosAnyoDepto(anyo, id_departamento, **num_incumplimientos**)
{id_departamento} es FK de Departamento

IncumplimientosAnyoEmpleado(anyo, id_empleado, **incumplimientos_propios, incumplimientos_departamento, incumplimientos_totales**)
{id_empleado} es FK de Empleado

Así, obtenemos el siguiente esquema lógico[1], en el que distinguiremos las tablas del sistema (azules), las que podríamos denominar tablas maestras de estados (grises), las tablas creadas para el módulo estadístico (verdes), y la tabla de log donde anotaremos las ejecuciones de los procedimientos y sus resultados (amarilla):

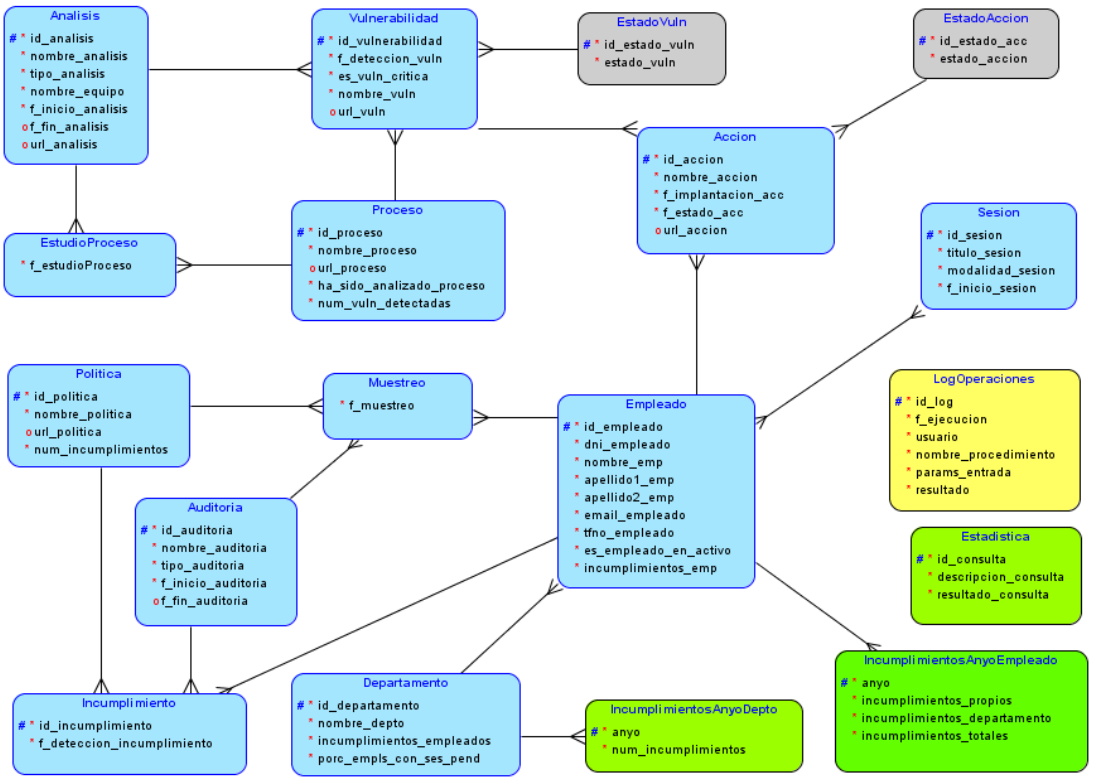


Ilustración 4 Diseño Lógico

[1] Generado mediante Oracle SQL Developer Data Modeler [3]

3.3 Diseño Físico.

A partir del diseño lógico que hemos obtenido en la fase anterior, pasamos a definir el diseño físico para el SGBD de nuestra elección (en este caso, Oracle 12.2 [1]), para el que hemos generado[2] el siguiente diagrama relacional:

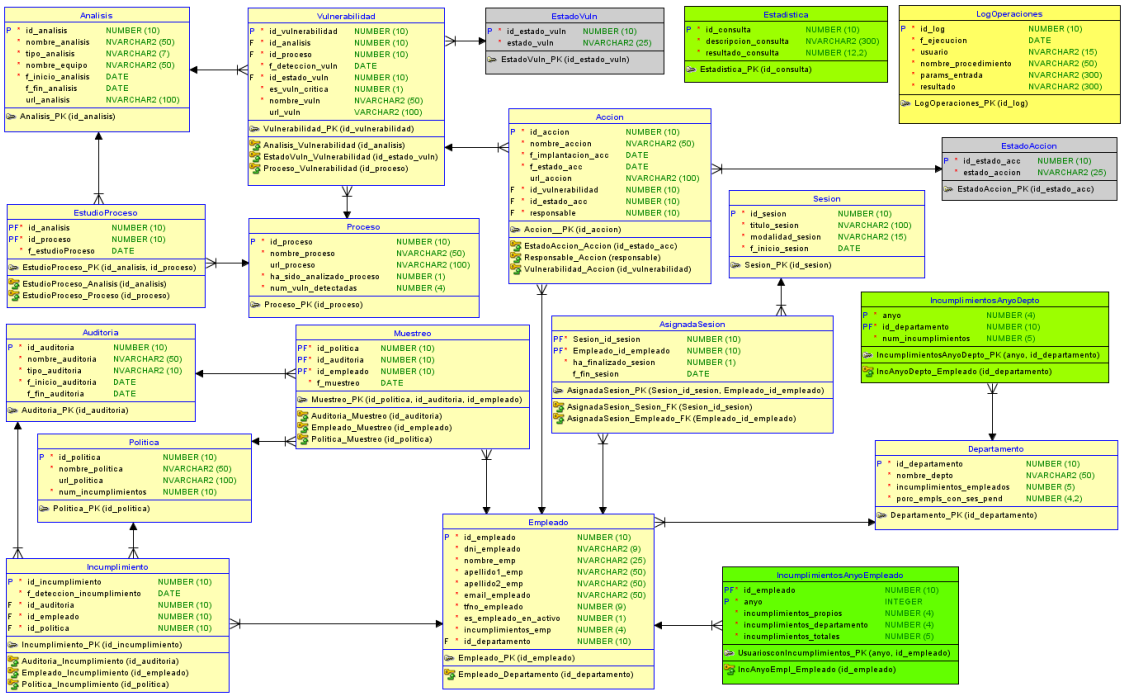


Ilustración 5 Diseño Físico

[2] Haciendo uso de nuevo de las funcionalidades que ofrece Oracle SQL Developer Data Modeler [3]

4. Implementación.

4.1 Seguridad. Roles. Usuarios.

No debemos olvidar que los datos que almacenaremos contienen información extremadamente sensible para la empresa, y que además estará protegida por las leyes de protección de datos tanto nacionales como europeas. Es por ello por lo que debemos controlar al máximo los accesos que se hagan al sistema, así como las inserciones, bajas y modificaciones de datos, ya que el sistema reflejará todas las debilidades de la empresa en términos de ciberseguridad.

Si un atacante lograse acceso a la información de la base de datos, tendría en primer lugar información sobre todas las vulnerabilidades del sistema informático de la empresa, sin necesidad de estudio previo. Además, podría marcar como resueltas vulnerabilidades que no lo estuvieran, para explotarlas posteriormente con tranquilidad.

En nuestro caso, podremos tomar medidas orientadas a restringir al acceso a los datos solo al personal autorizado, y a controlar los accesos y modificaciones de los datos. La indisponibilidad de los datos por ataques al gestor de bases de datos, así como la obtención de credenciales autorizadas por parte de usuarios no autorizados se sale del objetivo de este proyecto.

Para evitar el acceso a los datos por parte de personas no autorizadas, utilizaremos las funcionalidades que nos proporciona Oracle[7] a la hora de definir roles y usuarios.

Del estudio de nuestro caso podemos ver que hay un tipo de usuarios que deben poder insertar y modificar datos: los **operadores**, que introducirán los datos en el sistema (resultados de auditorías y análisis, tanto internos como externos, actualizaciones de estados, incumplimientos detectados, …). Y habrá otro tipo de usuarios (no excluyente), los **usuarios de solo lectura**, que podrá ver el resultado de las consultas preestablecidas del módulo estadístico, para comprobar el estado de la ciberseguridad de la empresa.

Tal y como se indica en el enunciado, el sistema debe poder soportar cualquier tamaño de datos, además de ser escalable en el caso de solicitarse nuevas funcionalidades. Para asegurar que el volumen de datos a gestionar no influya en la gestión del resto de la instancia, se asignará un almacenamiento propio, que permitirá el crecimiento (hasta las capacidades físicas dela máquina donde se aloje).

Además, el uso de la aplicación deberá ser exclusivamente mediante los procedimientos almacenados creados, lo que además permitirá mantener el log de todas las acciones que se ejecuten.

De cara a preparar la instancia para alojar nuestro sistema, comenzaremos por definir el esquema (usuario) que alojará los objetos que procederemos a crear (tablas, triggers, procedimientos, …). Por separar totalmente los objetos y datos de los de otras posibles aplicaciones existentes en la instancia, además de permitir al sistema un crecimiento que no interfiera con el de otros esquemas existentes en la misma instancia, primero definiremos una unidad de almacenamiento lógica (denominada *tablespace* en terminología Oracle[7]) propia, que será la que asignaremos por defecto para nuestro esquema principal. Este esquema podrá realizar la creación de las tablas y demás objetos, pero no será (o no debería ser) el utilizado para hacer uso de la aplicación.

Para la creación del tablespace y el esquema principal disponemos del script 1_TABLESPACE_Y_ESQUEMA_PRINCIPAL.sql, el cual deberá ser ejecutado por un usuario con permisos de creación de tablespaces y esquemas, así como de asignación de permisos. Este usuario típicamente será sys, el usuario administrador de la instancia que se crea durante la instalación, pero puede ser cualquier otro al que se le hayan asignado previamente los permisos necesarios.

Además, vamos a asumir que la instancia sobre la que se va a cargar el script ha sido creada con la opción de OMF[3]. En caso contrario, será necesario indicar en la sentencia de creación del tablespace la ruta y nombre del datafile a crear.

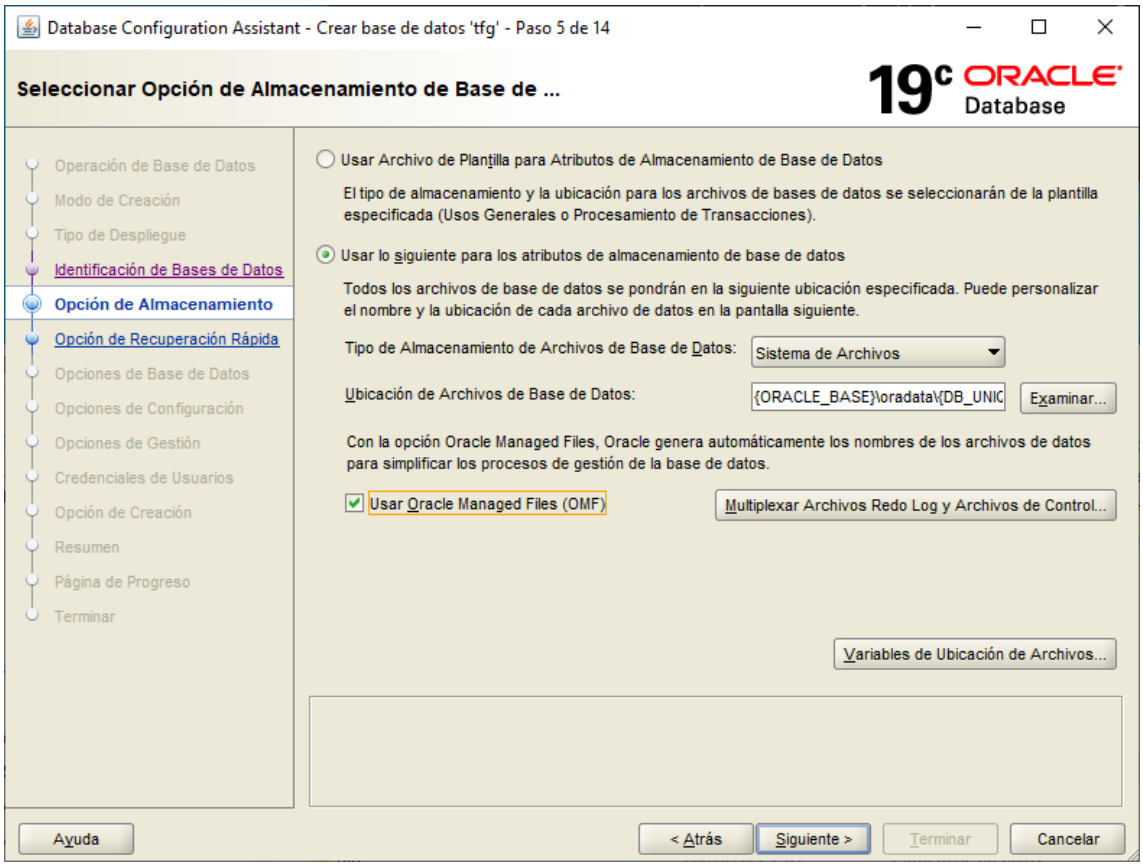


Ilustración 6 Selección de OMF durante instalación SGBD Oracle

```
-- FICHERO PARA CREACION DE USUARIOS: 1_TABLESPACE_Y_ESQUEMA_PRINCIPAL.sql
-- Autor: David Giraldez Sanchez
-- Tutor: Jordi Ferrer Duran

-- NOTA: Acentos eliminados intencionadamente para facilitar la lectura con cualquier configuracion del cliente.

-- TODOS LOS COMANDOS DE ESTE SCRIPT DEBERAN LANZARSE COMO SYS, O UN USUARIO CON LOS PERMISOS ADECUADOS (CREACION DE ESQUEMAS, Y ASIGNACION DE PERMISOS).

-- En el caso de que ya exista, descomentamos las siguientes dos líneas para eliminar tanto el esquema principal como el tablespace antes de volverlos a crear.

-- DROP USER "TFG" CASCADE;
-- DROP TABLESPACE "TS_TFG" INCLUDING CONTENTS AND DATAFILES CASCADE CONSTRAINTS;

-- CREAMOS EL TABLESPACE QUE ASIGNAREMOS POR DEFECTO AL ESQUEMA TFG;
-- ASUMIMOS QUE LA INSTANCIA SE HA CREADO CON OMF, EN CASO CONTRARIO HAY QUE INDICAR RUTA Y NOMBRE DEL DATAFILE A CREAR

CREATE TABLESPACE TS_TFG DATAFILE SIZE 512M AUTOEXTEND ON NEXT 512M MAXSIZE UNLIMITED;

-- Y EL ESQUEMA PRINCIPAL DE LA APLICACIÓN

CREATE USER "TFG" IDENTIFIED BY tfg2022 DEFAULT TABLESPACE "TS_TFG";

GRANT "CONNECT","RESOURCE" TO "TFG";
ALTER USER "TFG" DEFAULT ROLE "CONNECT", "RESOURCE";
GRANT CREATE SESSION, CREATE TABLE, CREATE VIEW, CREATE SEQUENCE, CREATE PROCEDURE, CREATE TRIGGER, CREATE MATERIALIZED VIEW TO "TFG";
ALTER USER "TFG" QUOTA UNLIMITED ON "TS_TFG";
```

Podremos lanzar el script desde cualquier cliente SQL conectado a la instancia, mi opción es SQL Developer[2]:

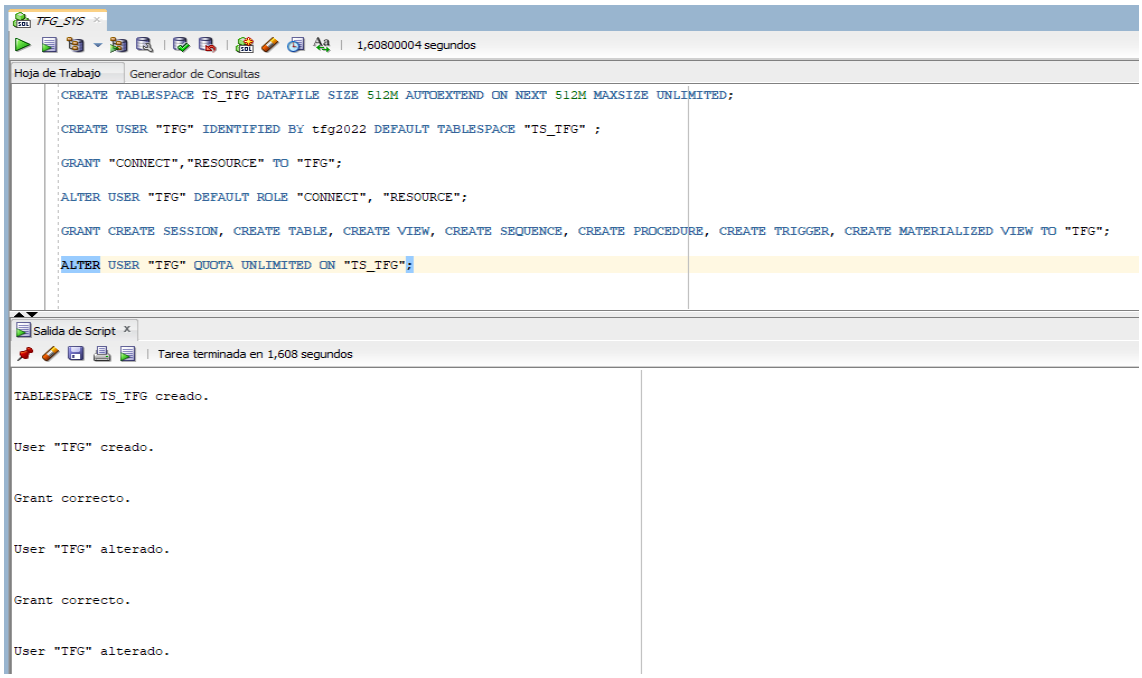


Ilustración 7 Alta de usuario

Además, como ya se ha indicado en distintos puntos del proyecto, los usuarios solo tendrán gestión sobre el sistema mediante los procedimientos y funciones previamente codificadas. En ningún caso

(con excepción del usuario de la aplicación, propietario por defecto de todos los objetos, y los usuarios con permisos de sysdba sobre la instancia) tendrán permisos para ejecutar comandos de insert, update, delete o select.

Para diferenciar sus permisos, se han definido dos roles que les permitirán “hacer su trabajo” y nada más:

- En el caso de los operadores, podrán ejecutar los procedimientos que hemos codificado en los paquetes PKG_ABM y PKG_FUNCIONAMIENTO. También se les permitirá ejecutar los proceso del paquete PKG_ESTADISTICAS.
- Y en el de los usuarios con permisos exclusivamente de lectura, solo estarán autorizados a ejecutar los procedimientos del paquete PKG_ESTADISTICAS.

Para ello, crearemos en la instancia del SGBD dos roles a los que le asignaremos permisos de ejecución sobre los paquetes de procedimientos creados, y posteriormente crearemos los usuarios (esquemas) específicos que utilizaremos para el uso de la aplicación, a los que les aplicaremos el rol previamente creado.

De esta forma, los usuarios creados no podrán realizar operaciones sobre los datos que no hagan uso de los procedimientos almacenados previamente diseñados (y que por tanto no queden reflejadas en el log). Y, en el caso de que se estime oportuno, cada usuario físico del sistema podrá tener un usuario de base de datos diferente al que se podrá asignar el rol más adecuado, por lo que se podrá seguir el trabajo realizado por cada uno en el og de operaciones.

Tras la creación de los objetos, podremos crear los roles y esquemas de trabajo para la aplicación siguiendo el script 10_ROLES_Y_USUARIOS.sql

```
-- FICHERO PARA CREACION DE USUARIOS PARA LA APLICACION: 8_ROLES_Y_USUARIOS.sql
-- Autor: David Giraldez Sanchez
-- Tutor: Jordi Ferrer Duran

-- NOTA: Acentos eliminados intencionadamente para facilitar la lectura con
cualquier configuracion del cliente.

-- TODOS LOS COMANDOS DE ESTE SCRIPT DEBERAN LANZARSE COMO SYS, O UN USUARIO CON
LOS PERMISOS ADECUADOS

-- ELIMINAMOS LOS ROLES, POR SI EXISTEN PREVIAMENTE
-- DROP ROLE OPERADOR;
-- DROP ROLE SOLO_LECTURA;

-- CREAMOS EL ROL OPERADOR: SERÁ EL QUE SE ASIGNE A LOS USUARIOS DE LA
APLICACIÓN, PARA QUE SOLO PUEDAN USAR LOS PROCEDIMIENTOS CREADOS.

CREATE ROLE OPERADOR;
GRANT CONNECT,RESOURCE TO OPERADOR;
GRANT EXECUTE ON TFG.PKG_ABM TO OPERADOR;
GRANT EXECUTE ON TFG.PKG_FUNCIONAMIENTO TO OPERADOR;
GRANT EXECUTE ON TFG.PKG_ESTADISTICA TO OPERADOR;
```

```
-- CREAMOS EL ROL SOLO_LECTURA: SE ASIGNARA A LOS USUARIOS QUE SOLO VAYAN A TENER
PERMISOS DE EJECUCIÓN DE LOS PROCEDIMIENTOS DE CONSULTA DE LAS ESTADISTICAS DE LA
APLICACIÓN

CREATE ROLE SOLO_LECTURA;
GRANT CONNECT,RESOURCE TO SOLO_LECTURA;
GRANT EXECUTE ON TFG.PKG_ESTADISTICA TO SOLO_LECTURA;

-- ELIMINAMOS LOS USUARIOS, POR SI EXISTEN PREVIAMENTE
-- DROP USER USR_OPERADOR CASCADE;
-- DROP USER USR_SOLO_LECTURA CASCADE;

-- CREAMOS LOS USUARIOS Y ASIGNAMOS LOS ROLES
-- NO INDICAMOS TABLESPACE POR DEFECTO, ASI QUE LES ASIGNARA USERS. NO NOS
IMPORTA PORQUE NO ALOJARAN DATOS.
CREATE USER USR_OPERADOR IDENTIFIED BY GIRALDEZ;
GRANT OPERADOR TO USR_OPERADOR;
CREATE USER USR_SOLO_LECTURA IDENTIFIED BY QUERY;
GRANT SOLO_LECTURA TO USR_SOLO_LECTURA;
```

4.2 Base de datos. Tablas.

Detallamos el diseño de todas las tablas que vamos a utilizar en el sistema, y los campos almacenados en ellas. Las cargaremos en el sistema ejecutando como TFG el script 2_CREACION_TABLAS.sql

Significados de las columnas:

- PK: reflejaremos qué campo(s) forma(n) la clave primaria.
- FK: indicaremos (si los hay) aquel(los) campo(s) que sea(n) clave foránea.
- SEQ: indicaremos si se rellenan de forma automática mediante una secuencia definida en el sistema y un trigger “before insert”.
- UQ: reflejará los campos marcados como UNIQUE.
- NULL: indicaremos si acepta o no valores nulos.

Tabla ACCION: Nos permitirá almacenar en el sistema los registros de las diferentes acciones de mitigación definidas para las vulnerabilidades descubiertas.

Accion							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_accion	NUMBER(10)	S		S		N	Identificador único de acción de mitigación
nombre_accion	NVARCHAR2(50)					N	Nombre de la acción de mitigación
f_implantacion_acc	DATE					N	Fecha definida para tener implantada la acción de mitigación.
f_estado_acc	DATE					N	Fecha en la que se ha actualizado el estado de la acción.
url_accion	NVARCHAR2(100)						URL de la documentación de la acción de mitigación en el gestor documental de la empresa.
id_vulnerabilidad	NUMBER(10)		S				Identificador único de la vulnerabilidad para la que se ejecuta la acción de mitigación.
id_estado_acc	NUMBER(10)		S				Identificador único del estado en el que se encuentra la acción de mitigación.
responsable	NUMBER(10)		S				Identificador único del empleado que es responsable de la implantación de la acción de mitigación.

Tabla ANALISIS: Nos permitirá almacenar en el sistema los registros de los diferentes análisis realizados.

Analisis							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_analisis	NUMBER(10)	S		S		N	Identificador único de análisis.
nombre_analisis	NVARCHAR2(50)					N	Nombre descriptivo del análisis.
tipo_analisis	NVARCHAR2(7)					N	Aceptará los valores EXTERNO o INTERNO.
nombre_equipo	NVARCHAR2(50)					N	Nombre del equipo que realiza el análisis.
f_inicio_analisis	DATE					N	Fecha de inicio del análisis.
f_fin_analisis	DATE						Fecha de finalización del análisis.
url_analisis	NVARCHAR2(100)						URL de la documentación relativa al análisis en el gestor documental de

							la empresa.

Restricciones adicionales:
ALTER TABLE Analisis ADD CONSTRAINT tipos_de_analisis CHECK (tipo_analisis IN ('EXTERNO', 'INTERNO'));

Tabla ASIGNADASESION: Nos permitirá almacenar en el sistema los registros de las sesiones de formación asignadas a empleados de la empresa.

AsignadaSesion							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
sesion_id_sesion	NUMBER(10)	S	S			N	Identificador único de sesión de formación.
id_empleado	NUMBER(10)	S	S			N	Identificador único de empleado
ha_finalizado_sesion	NUMBER(1)					N	Indica si el empleado ha finalizado o no la formación.
f_fin_sesion	DATE					N	Fecha en la que el empleado finaliza la sesión de formación.

Tabla AUDITORIA: Nos permitirá almacenar en el sistema los registros de las diferentes auditorías realizadas.

Auditoria							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_auditoria	NUMBER(10)	S		S		N	Identificador único de auditoría
nombre_auditoria	NVARCHAR2(50)					N	Nombre de la auditoría
tipo_auditoria	NVARCHAR2(10)					N	Aceptará los valores EXTERNA o INTERNA
f_inicio_auditoria	DATE					N	Fecha de inicio de la auditoría.
f_fin_auditoria	DATE						Se registrará la fecha en que la auditoría se da por finalizada

Tabla DEPARTAMENTO: Nos permitirá almacenar en el sistema los registros de los diferentes departamentos existentes en la empresa.

Departamento							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_departamento	NUMBER(10)	S		S		N	Identificador único de departamento.
nombre_depto	NVARCHAR2(50)					N	Nombre del departamento.
incumplimientos_empleados	NUMBER(5)					N	Campo inicialmente 0, se actualizará automáticamente con la operativa del sistema, y reflejará el número total de incumplimientos de los empleados del departamento.
porc_empls_con_ses_pend	NUMBER(4,2)					N	Campo inicialmente 0, se actualizará automáticamente con la operativa del sistema, y reflejará el porcentaje de usuarios del departamento con sesiones pendientes en el año en curso.

Tabla EMPLEADO: Nos permitirá almacenar en el sistema la información relativa a los empleados de la empresa.

Empleado							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_empleado	NUMBER(10)	S		S		N	Identificador único de empleado
dni_empleado	NVARCHAR2(9)					N	DNI del empleado
nombre_emp	NVARCHAR2(25)					N	Nombre del empleado
apellido1_emp	NVARCHAR2(50)					N	Primer apellido del empleado
apellido2_emp	NVARCHAR2(50)					N	Segundo apellido del empleado
email_empleado	NVARCHAR2(50)					N	Dirección de email del empleado
tfno_empleado	NUMBER(9)					N	Teléfono de contacto del empleado
es_empleado_en_activo	NUMBER(1)					N	Indica si el empleado sigue perteneciendo a la empresa. Consideraremos 1=SI, 0 = NO
incumplimientos_emp	NUMBER(4)					N	Refleja el total de incumplimientos en los que ha incurrido el empleado. Se irá actualizando automáticamente cuando se detecten nuevos incumplimientos.
id_departamento	NUMBER(10)		S			N	Identificador único del departamento al que pertenece el empleado.

Restricciones adicionales:

```
ALTER TABLE empleado ADD CHECK ( es_empleado_en_activo IN ( 0, 1 ) );
```

Tabla ESTADISTICA: Tabla del módulo estadístico. Nos permitirá almacenar en el sistema los resultados de las diferentes consultas estadísticas que devuelven un único valor numérico. Se irán actualizando de forma automática, para permitir ejecutar las consultas en tiempo constante 1.

Tabla Estadistica							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_consulta	NUMBER(10)	S				N	Identificador único del registro; SIEMPRE se asignarán en el mismo orden, por eso la consideramos tabla maestra.
descripcion_consulta	NVARCHAR2(300)					N	Texto indicativo de la consulta de la que se recogerán los datos.
resultado_consulta	NUMBER(12, 2)					N	Resultado de la ejecución de la consulta identificada por id_consulta. Será actualizado de forma automática por el sistema.

Tabla ESTADOACCION: Nos permitirá almacenar en el sistema los diferentes estados en los que se puede encontrar una acción de mitigación.

EstadoAccion							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_estado_acc	NUMBER(10)	S				N	Identificador único de estado

estado_accion	NVARCHAR(25)					N	Estado por el que puede pasar una acción de mitigación.

Según el enunciado, los estados posibles serán *Definida*, *En proceso*, *Acabada*, *En Revisión*. En principio se añadió esta restricción al campo en la definición de la tabla. Sin embargo, considerando que sería deseable poder añadir estados nuevos sin tocar la estructura de la tabla, con solo añadirlos a la tabla, se eliminó la restricción. Esto ha añadido una ligera complejidad al código del sistema, ya que debe comprobar siempre que se cumple la restricción, pero no habrá que cambiarlo tampoco en caso de ampliación de posibles estados. Al ser necesario que se inserten los estados siempre con el mismo id, la consideramos tabla maestra (se carga en el inicio del sistema y generalmente no se vuelve a cambiar).

Tabla ESTADOVULNERABILIDAD: Nos permitirá almacenar en el sistema los diferentes estados en los que se puede encontrar una vulnerabilidad.

EstadoVulnerabilidad							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_estado_vuln	NUMBER(10)	S				N	Identificador único de estado
estado_vuln	NVARCHAR(25)					N	Estado por el que puede pasar una vulnerabilidad.

Según el enunciado, los estados posibles serán *Identificada*, *No Mitigada*, *Parcialmente Mitigada*, *Totalmente Mitigada*. En principio se añadió esta restricción al campo en la definición de la tabla. Sin embargo, considerando que sería deseable poder añadir estados nuevos sin tocar la estructura de la tabla, con solo añadirlos a la tabla, se eliminó la restricción. Esto ha añadido una ligera complejidad al código del sistema, ya que debe comprobar siempre que se cumple la restricción, pero no habrá que cambiarlo tampoco en caso de ampliación de posibles estados. Al ser necesario que se inserten los estados siempre con el mismo id, la consideramos tabla maestra (se carga en el inicio del sistema y generalmente no se vuelve a cambiar).

Tabla ESTUDIOPROCESO: Nos permitirá almacenar en el sistema los registros de los diferentes análisis realizados sobre los procesos de la empresa, así como la fecha del estudio.

EstudioProceso							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_analisis	NUMBER(10)	S	S			N	Identificador único de análisis
id_proceso	NUMBER(10)	S	S			N	Identificador único de proceso
f_estudioProceso	DATE					N	Fecha del estudio

Tabla INCUMPLIMIENTO: Nos permitirá almacenar en el sistema los registros de los diferentes incumplimientos de políticas de seguridad detectados.

Incumplimiento

Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_incumplimiento	NUMBER(10)	S		S		N	Identificador único de incumplimiento.
f_deteccion_incumplimiento	DATE					N	Fecha en la que se ha detectado el incumplimiento.
id_auditoria	NUMBER(10)		S			N	Identificador único de la auditoría en la que se ha detectado el incumplimiento.
id_empleado	NUMBER(10)		S			N	Identificador único del empleado que ha cometido el incumplimiento.
id_politica	NUMBER(10)		S			N	Identificador único de la política que se ha incumplido.

Tabla INCUMPLIMIENTOSANYODEPTO: Tabla del módulo estadístico. Nos permitirá almacenar en el sistema los registros de los incumplimientos de cada departamento, agrupados de forma anual. Los registros se actualizarán de forma automática.

IncumplimientosAnyoDepto							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
anyo	NUMBER(4)	S				N	Año en el que se producen los incumplimientos
id_departamento	NUMBER(10)	S	S			N	Identificador único de departamento
num_incumplimientos	NUMBER(5)					N	Numero de incumplimientos anotados al departamento en el año indicado. Se actualiza automáticamente.

Tabla INCUMPLIMIENTOSANYOEMPLEADO: Tabla del módulo estadístico. Nos permitirá almacenar en el sistema los registros de los incumplimientos de cada empleado, agrupados de forma anual. Los registros se actualizarán de forma automática.

IncumplimientosAnyoEmpleado							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_empleado	NUMBER(10)	S		S		N	Identificador único de registro
anyo	INTEGER					N	Año en el que se producen los incumplimientos.
incumplimientos_propios	NUMBER(4)					N	Incumplimientos del empleado en el año en cuestión. Se actualiza automáticamente.
incumplimientos_departamento	NUMBER(4)					N	Incumplimientos del departamento (al que pertenece el empleado) en el año en cuestión. Se actualiza automáticamente.
incumplimientos_totales	NUMBER(5)					N	Suma de los dos anteriores, Se actualiza automáticamente.

Tabla LOGOPERACIONES: Nos permitirá almacenar en el sistema los registros de las ejecuciones de procedimientos almacenados del sistema.

LogOperaciones							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_log	NUMBER(10)	S		S		N	Identificador de los registros de log.

f_ejecucion	DATE					N	Fecha y hora de ejecución del procedimiento.
usuario	NVARCHAR2(15)					N	Usuario de BBDD que ha ejecutado el procedimiento.
nombre_procedimiento	NVARCHAR2(50)					N	Nombre del procedimiento ejecutado.
params_entrada	NVARCHAR2(300)					N	Parámetros de entrada del procedimiento.
resultado	NVARCHAR2(300)					N	Resultado del procedimiento.

Tabla MUESTREO: Nos permitirá almacenar en el sistema los registros de los muestreos realizados por las diferentes auditorías (que podrán ser internas o externas).

Muestreo							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_politica	NUMBER(10)	S	S			N	Identificador único de política.
id_auditoria	NUMBER(10)	S	S			N	Identificador único de auditoría.
id_empleado	NUMBER(10)	S	S			N	Identificador único de empleado.
f_muestreo	DATE					N	Fecha de ejecución de la prueba.

Tabla POLITICA: Nos permitirá almacenar en el sistema los registros de las diferentes políticas de seguridad definidas en la empresa.

Politica							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_politica	NUMBER(10)	S		S		N	Identificador único de política.
nombre_politica	NVARCHAR2(50)					N	Nombre descriptivo asociado a la política.
url_politica	NVARCHAR2(100)						URL de la documentación relativa a la política de seguridad en el gestor documental de la empresa.
num_incumplimientos	NUMBER(4)					N	Numero de vulnerabilidades detectadas asociadas al proceso. Por defecto es 0. Se actualizará automáticamente al registrar una vulnerabilidad.

Tabla PROCESO: Nos permitirá almacenar en el sistema los registros de los diferentes procesos de la empresa.

Proceso							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_proceso	NUMBER(10)	S		S		N	Identificador único del proceso
nombre_proceso	NVARCHAR2(50)					N	Nombre descriptivo del proceso.
url_proceso	NVARCHAR2(100)						URL de la documentación relativa al proceso en el gestor documental de la empresa.
ha_sido_analizado_proceso	NUMBER(1)					N	Se registrará de forma automática si el proceso ha sido analizado en alguna ocasión por algún análisis o no.
num_vuln_detectadas	NUMBER(10)					N	Numero de vulnerabilidades detectadas asociadas al proceso. Por defecto es 0. Se actualizará automáticamente al registrar una vulnerabilidad.

Tabla SESION: Nos permitirá almacenar en el sistema los registros de las sesiones de formación impartidas.

Sesion							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_sesion	NUMBER(10)	S		S		N	Identificador único de la sesión de formación.
titulo_sesion	NVARCHAR2(100)					N	Título de la sesión de formación.
modalidad_sesion	NVARCHAR2(15)					N	Tipos de sesiones de formación, podrán ser presenciales o telemáticas.
f_inicio_sesion	DATE					N	Fecha de inicio de la sesión de formación.

Restricciones adicionales:

```
ALTER TABLE sesion ADD CHECK ( modalidad_sesion IN ( 'PRESENCIAL', 'TELEMATICA' ) );
```

Tabla VULNERABILIDAD: Nos permitirá almacenar en el sistema los registros de las diferentes vulnerabilidades detectadas en los sistemas de la empresa.

Vulnerabilidad							
Campo	Tipo	PK	FK	SEQ	UQ	NULL	Descripcion
id_vulnerabilidad	NUMBER(10)	S		S		N	Identificador único de la vulnerabilidad.
id_analisis	NUMBER(10)		S			N	Identificador único del análisis en el que se ha descubierto la vulnerabilidad.
id_proceso	NUMBER(10)		S			N	Identificador único del proceso al que afecta la vulnerabilidad que se ha descubierto.
f_deteccion_vuln	DATE					N	Fecha en la que se detecta la vulnerabilidad.
id_estado_vuln	NUMBER(10)		S			N	Identificador del estado en el que se encuentra la vulnerabilidad. Los estados posibles se han definido en el enunciado, pero pueden cambiar.
es_vuln_critica	NUMBER(1)					N	Indica si se ha considerado la vulnerabilidad como crítica o no. 1=SI , 0=NO
nombre_vuln	NVARCHAR2(50)					N	Nombre asignado a la vulnerabilidad.
url_vuln	NVARCHAR2(100)						URL de la documentación relativa a la vulnerabilidad en el gestor documental de la empresa.

Restricciones adicionales:

```
ALTER TABLE vulnerabilidad ADD CHECK ( es_vuln_critica IN ( 0, 1 ) );
```

4.3 Procedimientos.

Como ya se ha indicado en los requisitos no funcionales[5], el enunciado indica que es necesario que toda la gestión del sistema se lleve a cabo mediante procedimientos almacenados, para tratar de evitar comportamientos no deseados del sistema.

Todos los procedimientos incluyen, por petición expresa del cliente, tratamiento de excepciones, para controlar los posibles errores que se comentan. He creado un tipo personalizado de excepción, para recoger algunas posibilidades, y además he comprobado algunas de las excepciones más habituales.

Por otro lado, todos los procedimientos deberán generar como parámetro de salida una cadena de texto, denominada RSP con el resultado de su ejecución: ”OK” en caso correcto, y “ERROR + DESCRIPCIÓN DEL ERROR” en caso contrario. Dicha cadena se almacenará en el log de operaciones como resultado de la ejecución del procedimiento.

Los distintos procedimientos del sistema los hemos agrupado en paquetes, organizados por funcionalidades similares, para facilitar la asignación de permisos de ejecución a los roles creados.

4.3.1 PKG_FUNCIONAMIENTO

El esquema general de todos los procedimientos de este paquete incorpora el siguiente código (obviamos la cabecera y fin):

```
-- NULOS: Comprobamos que ningún campo necesario tiene valor
nulo.
…
-- Comprobamos que el id proporcionado identifica a un registro
en el sistema.
-- Al ser PK, lo comprobaría el SGBD, pero así personalizamos el
mensaje en el log de operaciones en caso de error.
…
-- Si NO existe, lo anotamos en el log de operaciones y salimos
(excepción).
-- Si SÍ existe
      -- Ejecutamos la consulta XX
--  Almacenamos el resultado en el registro XX de la tabla
estadística
-- Anotamos en el log de operaciones
```

Se han incorporado en este paquete dos tipos de procedimientos: los que se ocupan de recalcular el contenido de determinados campos calculados almacenados en las tablas del sistema, y los que se ocupan de actualizar los campos de la tabla ESTADISTICA, que permitirá obtener en tiempo constante 1 los resultados de las consultas que se han considerado de interés para el sistema.

Los primeros son llamados desde los procedimientos de ABM, de forma automática (y es por ello que debemos compilar previamente este

paquete, antes de incorporar el de ABM), y los segundos son lanzados por triggers, también de forma automática. Por lo tanto, los procedimientos de este paquete en general no serán ejecutados “manualmente” por ningún usuario. Aún así, al igual que los del paquete PKG_ABM, aparte del usuario propietario del sistema (TFG en mi caso), los usuarios con el rol de OPERADOR podrán ejecutar estos procedimientos.

Se han creado los siguientes procedimientos:

- Procedimientos de actualización de campos calculados
    - ACTUALIZAR_VULNERABILIDADES_DE_PROCESO
    - ACTUALIZAR_PORCENTAJE_DE_DEPARTAMENTO
    - ACTUALIZAR_ESTADO_VULNERABILIDAD
    - ACTUALIZAR_INCUMPLIMIENTOS_DE_POLITICA
- Procedimientos de actualización de la tabla ESTADISTICA
    - ACTUALIZAR_CONSULTA_1: es ejecutado por el disparador TRG_ACTUALIZAR_CONSULTA_1
    - ACTUALIZAR_CONSULTA_2: es ejecutado por el disparador TRG_ACTUALIZAR_CONSULTA_2
    - ACTUALIZAR_CONSULTA_3: es ejecutado por el disparador TRG_ACTUALIZAR_CONSULTA_3
    - ACTUALIZAR_CONSULTA_4: es ejecutado por el disparador TRG_ACTUALIZAR_CONSULTA_4
    - ACTUALIZAR_CONSULTA_6: es ejecutado por el disparador TRG_ACTUALIZAR_CONSULTA_6
    - ACTUALIZAR_CONSULTA_7: es ejecutado por el disparador TRG_ACTUALIZAR_CONSULTA_7
    - ACTUALIZAR_CONSULTA_8: es ejecutado por el disparador TRG_ACTUALIZAR_CONSULTA_8
    - ACTUALIZAR_CONSULTA_9: es ejecutado por el disparador TRG_ACTUALIZAR_CONSULTA_9
    - ACTUALIZAR_CONSULTA_10: es ejecutado por el disparador TRG_ACTUALIZAR_CONSULTA_10
    - ACTUALIZAR_CONSULTA_11: es ejecutado por el disparador TRG_ACTUALIZAR_CONSULTA_11

Podremos cargar este paquete ejecutando por orden los scripts 3_PKG_FUNCIONAMIENTO.sql y 4_PKG_FUNCIONAMIENTO_BODY.sql

4.3.2 PKG_ABM

En este paquete hemos agrupado todos los procedimientos de alta, baja y modificación (ABM) de las entidades del sistema. Como su propio nombre indica, los procedimientos de alta se encargan de insertar nuevos registros en el sistema, los de baja se encargan de eliminar registros existentes, y los de modificación actualizan registros ya existentes en el sistema.

El esquema general de todos los procedimientos de ABM sigue la siguiente estructura:

```
PROCEDURE ABM_TABLA (
      pParametroEntrada1 IN TABLA.campo1%TYPE,
      …,
      RSP OUT VARCHAR2) AS

      MI_EXC   EXCEPTION;
      PAR_IN      VARCHAR2(1000);
      PROC     VARCHAR2(100);
      vVariable1  ALGUN_TIPO_VALIDO;
      …
BEGIN
PAR_IN := 'Campo1: '||pParametroEntrada1||', Campo2: '||…;
PROC := 'PKG_ABM.ABM_TABLA';

-- NULOS: Comprobamos que ningun campo necesario tiene valor
nulo.
RSP := 'RESULTADO: ERROR: El campo1 no puede ser nulo.';
IF pParametroEntrada1 IS NULL THEN RAISE MI_EXC; END IF;
…
-- RESTRICCIONES: Si no hay valores nulos, comprobamos las
restricciones de integridad que afecten
…
--   ABM:   Comprobado   todo   lo   anterior,   tratamos   de
insertar/borrar/modificar el registro.
…
-- Registramos la actividad en el LOGOPERACIONES
INSERT      INTO      LOGOPERACIONES      (NOMBRE_PROCEDIMIENTO,
PARAMS_ENTRADA, RESULTADO) VALUES(PROC,PAR_IN,RSP);

-- ESTADISTICAS: Actualizamos los campos estadisticos afectados
por la ejecucion del procedimiento
…
-- ANOTAMOS LOS CAMBIOS
COMMIT;

-- Si ha saltado una excepcion, la procesamos
EXCEPTION
  WHEN MI_EXC THEN
   INSERT     INTO     LOGOPERACIONES     (NOMBRE_PROCEDIMIENTO,
PARAMS_ENTRADA, RESULTADO) VALUES(PROC,PAR_IN,RSP);
   COMMIT;
-- En determinados casos comprobamos si ha saltado alguna de las
excepciones predefinidas de Oracle, y finalmente
  WHEN OTHERS THEN
   RSP := 'RESULTADO: ERROR: ' || TO_CHAR(SQLERRM(-SQLCODE));
   INSERT     INTO     LOGOPERACIONES     (NOMBRE_PROCEDIMIENTO,
PARAMS_ENTRADA, RESULTADO) VALUES(PROC,PAR_IN,RSP);
```

```
   COMMIT;
END ABM_TABLA;
```

Se han creado los siguientes procedimientos:

- ALTAS
  - ALTA_ACCION
  - ALTA_ANALISIS
  - ALTA_ASIGNADASESION
  - ALTA_AUDITORIA
  - ALTA_DEPARTAMENTO
  - ALTA_EMPLEADO
  - ALTA_ESTADOACCION
  - ALTA_ESTADOVULN
  - ALTA_ESTUDIOPROCESO
  - ALTA_INCUMPLIMIENTO
  - ALTA_MUESTREO
  - ALTA_POLITICA
  - ALTA_PROCESO
  - ALTA_SESION
  - ALTA_VULNERABILIDAD
- BAJAS
  - BAJA_ACCION
  - BAJA_ANALISIS
  - BAJA_ASIGNADASESION
  - BAJA_AUDITORIA
  - BAJA_DEPARTAMENTO
  - BAJA_EMPLEADO
  - BAJA_ESTADOACCION
  - BAJA_ESTADOVULN
  - BAJA_ESTUDIOPROCESO
  - BAJA_INCUMPLIMIENTO
  - BAJA_MUESTREO
  - BAJA_POLITICA
  - BAJA_PROCESO
  - BAJA_SESION
  - BAJA_VULNERABILIDAD
- MODIFICACIONES
  - MODIFICA_ACCION
  - MODIFICA_ANALISIS
  - MODIFICA_ASIGNADASESION
  - MODIFICA_AUDITORIA
  - MODIFICA_DEPARTAMENTO
  - MODIFICA_EMPLEADO
  - MODIFICA_ESTADOACCION
  - MODIFICA_ESTADOVULN
  - MODIFICA_ESTUDIOPROCESO
  - MODIFICA_INCUMPLIMIENTO
  - MODIFICA_MUESTREO
  - MODIFICA_POLITICA
  - MODIFICA_PROCESO

- MODIFICA_SESION
- MODIFICA_VULNERABILIDAD

Aparte del usuario propietario del sistema (TFG en mi caso), tan solo los usuarios con el rol de OPERADOR podrán ejecutar estos procedimientos. Al no disponer de otros permisos sobre el sistema, nos aseguramos de que el sistema mantenga la integridad exigida.

Podremos cargarlos en el sistema ejecutando (por este orden) los scripts 5_PKG_ABM.sql y 6_PKG_ABM_BODY.sql (después de haber ejecutado los scripts 1, 2, 3 y 4, evidentemente), desde una conexión del usuario propietario del sistema.

4.3.3 PKG_ESTADISTICAS

El esquema general de todos los procedimientos de este paquete es similar (con un par de excepciones que señalaremos más adelante):

1. Obtenemos de la tabla ESTADISTICA los campos descripcion_consulta y resultado_consulta correspondientes al registro de la consulta que nos interesa,

2. Y lo mostramos por pantalla mediante DBMS_OUTPUT.PUT_LINE (tendremos que habilitar la salida de DBMS en SQL Developer[2] para ver el resultado; en otros clientes como SQL Plus lo veremos sin cambiar nada).

3. Seguimos controlando, como en todos los procedimientos, si salta alguna excepción. En este caso hemos añadido un mensaje personalizado para el caso de que el valor obtenido de la tabla ESTADISTICA no corresponda a un indicador registrado en el sistema.

Se han creado los siguientes procedimientos:

- VER_RESULTADO_CONSULTA_1
- VER_RESULTADO_CONSULTA_2
- VER_RESULTADO_CONSULTA_3
- VER_RESULTADO_CONSULTA_4
- VER_RESULTADO_CONSULTA_5
- VER_RESULTADO_CONSULTA_6
- VER_RESULTADO_CONSULTA_7
- VER_RESULTADO_CONSULTA_8
- VER_RESULTADO_CONSULTA_9
- VER_RESULTADO_CONSULTA_10
- VER_RESULTADO_CONSULTA_11

Las excepciones son:

- El procedimiento que muestra el resultado de la consulta 5, en lugar de consultar la tabla estadística, obtiene el dato registrado (y actualizado automáticamente) en la tabla de departamentos.
- El procedimiento que muestra el resultado de la consulta 8, obtiene de la tabla ESTADISTICA el identificador de la sesión, y a partir de él muestra el título de la sesión. La consulta se sigue ejecutando en tiempo constante, ya que se consulta primero un campo de la tabla ESTADISTICA, y luego se recupera en base al anterior (indexado, porque es clave primaria, por lo que la consulta es también inmediata) otro campo de la tabla SESION.
- Lo mismo ocurre con el procedimiento que muestra la consulta 11, ya que en ESTADISTICA almacenamos el identificador del empleado, que se utiliza para obtener el nombre completo desde la tabla EMPLEADO.

4.4 Secuencias y disparadores

Para realizar inserciones en las tablas evitando problemas de duplicidad en los campos identificativos (en nuestro caso solo se trataría de las claves primarias de algunas de las tablas), utilizaremos el método que nos permite Oracle[7]:

1. Creamos una secuencia, indicado el primer valor y el incremento, asignándole un nombre lo más identificativo posible.
2. Posteriormente crearemos un disparador que, en el caso de inserción de un nuevo registro, asigne de forma automática un valor nuevo al id a partir del último valor obtenido de la secuencia antes de la inserción, evitando que tengamos que controlar ese campo para no repetirlo.

El esquema utilizado para todos los que han sido necesarios es el siguiente:

```
-- TABLA
CREATE SEQUENCE NOMBRE_DE_SECUENCIA START WITH 1 INCREMENT BY 1
NOCACHE ORDER;

CREATE OR REPLACE TRIGGER TRG_INSERTAR_TABLA
BEFORE INSERT ON TFG.TABLA
FOR EACH ROW
WHEN (NEW.id_de_tabla IS NULL)
BEGIN
:NEW.id_de_tabla := NOMBRE_DE_SECUENCIA.NEXTVAL;
END;
/
```

Un caso ligeramente diferente es el de la tabla del log de ejecución de procedimientos. En este caso, no solo insertaremos el identificador único del registro; también recogeremos de la tabla dual[4] el nombre del usuario de base de datos con el que estamos ejecutando la inserción, y la fecha completa:

```
-- TABLA LOGOPERACIONES
CREATE SEQUENCE SEQ_ID_LOGOPERACIONES START WITH 1 INCREMENT BY
1 NOCACHE ORDER ;

CREATE OR REPLACE TRIGGER TRG_INSERTAR_LOGOPERACIONES
BEFORE INSERT ON TFG.LOGOPERACIONES
FOR EACH ROW
WHEN (NEW.id_log IS NULL)
BEGIN
:NEW.id_log := SEQ_ID_LOGOPERACIONES.NEXTVAL;
SELECT    TO_CHAR(SYSDATE,    'DD-MM-YYYY    HH24:MI:SS')    INTO
:NEW.f_ejecucion FROM DUAL;
SELECT USER INTO :NEW.usuario FROM DUAL;
END;
```

[4] https://en.wikipedia.org/wiki/DUAL_table

Se trata de los que he denominado triggers "autoincrementales", y que podremos cargar en el sistema ejecutando con el usuario TFG el script 6_SECUENCIAS_Y_DISPARADORES.sql

Se han definido con este objetivo las siguientes secuencias y triggers:

- Secuencias
  - SEQ_ID_ACCION
  - SEQ_ID_ANALISIS
  - SEQ_ID_AUDITORIA
  - SEQ_ID_DEPARTAMENTO
  - SEQ_ID_EMPLEADO
  - SEQ_ID_INCUMPLIMIENTO
  - SEQ_ID_POLITICA
  - SEQ_ID_PROCESO
  - SEQ_ID_SESION
  - SEQ_ID_VULNERABILIDAD
  - SEQ_ID_LOGOPERACIONES
- Disparadores
  - TRG_INSERTAR_ACCION
  - TRG_INSERTAR_ANALISIS
  - TRG_INSERTAR_AUDITORIA
  - TRG_INSERTAR_DEPARTAMENTO
  - TRG_INSERTAR_EMPLEADO
  - TRG_INSERTAR_INCUMPLIMIENTO
  - TRG_INSERTAR_POLITICA
  - TRG_INSERTAR_PROCESO
  - TRG_INSERTAR_SESION
  - TRG_INSERTAR_VULNERABILIDAD
  - TRG_INSERTAR_LOGOPERACIONES

Por otro lado, para la actualización de las tablas estadísticas ha sido necesario crear otros disparadores, que se ejecutarán (según los distintos casos) después de cada alta, baja y/o modificación de registros en las tablas del sistema, para recalcular los valores estadísticos por los que se quiere poder consultar en tiempo 1.

Se trata de los que he denominado TRIGGERS ESTADISTICOS, y son los siguientes:

- TRG_ACTUALIZAR_CONSULTA_1: Tras cada borrado o inserción en la tabla PROCESO, ejecuta el procedimiento que recalcula los datos estadísticos que pudieran haber cambiado por la actividad. Actualiza el resultado en la tabla ESTADISTICA, en el campo resultado_consulta del registro 1.

- TRG_ACTUALIZAR_CONSULTA_2: Tras cada borrado o inserción en la tabla VULNERABILIDAD, ejecuta el procedimiento que recalcula los datos estadísticos que pudieran haber cambiado por la

actividad. Actualiza el resultado en la tabla ESTADISTICA, en el campo resultado_consulta del registro 2.

- TRG_ACTUALIZAR_CONSULTA_3: Tras cada borrado, actualización o inserción en la tabla ACCION, ejecuta el procedimiento que recalcula los datos estadísticos que pudieran haber cambiado por la actividad. Actualiza el resultado en la tabla ESTADISTICA, en el campo resultado_consulta del registro 3.

- TRG_ACTUALIZAR_CONSULTA_4: Tras cada borrado, actualización o inserción en la tabla INCUMPLIMIENTO, ejecuta el procedimiento que recalcula los datos estadísticos que pudieran haber cambiado por la actividad. Actualiza el resultado en la tabla ESTADISTICA, en el campo resultado_consulta del registro 4.

- TRG_ACTUALIZAR_CONSULTA_6: Tras cada borrado o inserción en la tabla INCUMPLIMIENTO, ejecuta el procedimiento que recalcula los datos estadísticos que pudieran haber cambiado por la actividad. Actualiza el resultado en la tabla ESTADISTICA, en el campo resultado_consulta del registro 6.

- TRG_ACTUALIZAR_CONSULTA_7: Tras cada borrado, actualización o inserción en la tabla ACCION, ejecuta el procedimiento que recalcula los datos estadísticos que pudieran haber cambiado por la actividad. Actualiza el resultado en la tabla ESTADISTICA, en el campo resultado_consulta del registro 7.

- TRG_ACTUALIZAR_CONSULTA_8: Tras cada borrado o inserción en la tabla ASIGNADASESION, ejecuta el procedimiento que recalcula los datos estadísticos que pudieran haber cambiado por la actividad. Actualiza el resultado en la tabla ESTADISTICA, en el campo resultado_consulta del registro 8.

- TRG_ACTUALIZAR_CONSULTA_9: Tras cada borrado, modificación o inserción en la tabla VULNERABILIDAD, ejecuta el procedimiento que recalcula los datos estadísticos que pudieran haber cambiado por la actividad. Actualiza el resultado en la tabla ESTADISTICA, en el campo resultado_consulta del registro 9.

- TRG_ACTUALIZAR_CONSULTA_10: Tras cada borrado o inserción en la tabla ACCION, ejecuta el procedimiento que recalcula los datos estadísticos que pudieran haber cambiado por la actividad. Actualiza el resultado en la tabla ESTADISTICA, en el campo resultado_consulta del registro 10.

- TRG_ACTUALIZAR_CONSULTA_11: Tras cada borrado, actualización o inserción en la tabla ACCION, ejecuta el procedimiento que recalcula los datos estadísticos que pudieran haber cambiado por la actividad. Actualiza el resultado en la tabla ESTADISTICA, en el campo resultado_consulta del registro 11.

4.5. Módulo Estadístico

Tal y como se refleja en el enunciado, se demanda por parte del cliente un listado de consultas que desea que se ejecuten en tiempo constante 1. Lo que esto nos indica es que, al realizar la consulta, quieren que el sistema ya tengo el valor deseado calculado (en lugar de ejecutar una consulta que puede ser pesada y tardar un tiempo indefinido).

Esto nos ha obligado a añadir al sistema campos y tablas que, en principio, no eran necesarias, para almacenar los datos calculados que den respuesta a dichas consultas.

Los campos añadidos son:

- PROCESO.num_vuln_detectadas: Este campo, en el que se registra el número de vulnerabilidades detectadas que están asociadas al proceso en cuestión, es actualizado de forma automática tras cada alta o baja de un registro de vulnerabilidad mediante el procedimiento almacenado PKG_FUNCIONAMIENTO.ACTUALIZAR_VULNERABILIDADES_DE_PROCESO. No se ejecutará tras una modificación, porque no hemos permitido que, una vez dada de alta una vulnerabilidad, se modifique el proceso al que afecta.

- POLITICA.num_incumplimientos: este campo, que contiene el número de incumplimientos registrados en el sistema asociados a la política en cuestión, es actualizado automáticamente tras cada alta o baja de un incumplimiento mediante el procedimiento almacenado PKG_FUNCIONAMIENTO.ACTUALIZAR_INCUMPLIMIENTOS_DE_POLITICA. Al igual que en el caso anterior, no se admite la modificación de la auditoria, el empleado o la política asociados al incumplimiento, por lo tanto no es necesario recalcular el campo tras modificaciones.

- DEPARTAMENTO.porc_empls_con_ses_pend: en este campo anotaremos el resultado calculado que indica el porcentaje de empleados que pertenecen al departamento en cuestión y que tienen sesiones de formación asignadas sin terminar. Es actualizado tras cada alta, baja o modificación de una asignación de sesión formativa mediante el procedimiento almacenado PKG_FUNCIONAMIENTO.ACTUALIZAR_PORCENTAJE_DE_DEPARTAMENTO.

- VULNERABILIDAD.id_estado_vuln: en este caso, no se trata de un campo añadido al sistema. Pero sí es un campo que se actualiza de forma automática tras cada alta, baja o modificación de ACCION, para que el estado de la vulnerabilidad sea el correcto (se comprueba si quedan

acciones sin acabar asociadas a la vulnerabilidad). Se esta actualización se ocupa el procedimiento almacenado PKG_FUNCIONAMIENTO.ACTUALIZAR_ESTADO_VULNERABILIDAD.

En cuanto a las tablas añadidas, son 3:

- ESTADISTICA: en esta tabla se almacenan los resultados de las consultas estadísticas que devuelven un valor numérico. Nos ocuparemos de que siempre estén en el mismo orden, y de esta forma los procedimientos de visualización podrán siempre consultarlas sin necesidad de comprobar que se trata del resultado que nos interesa. Hay 3 excepciones a esta regla:
    - La consulta 5, cuyo resultado se obtiene consultando directamente el campo porc_empls_con_ses_pend de la tabla DEPARTAMENTO. El registro número 5 de la tabla ESTADISTICA no se actualizará, pero se inicializará a 0 por mantener la coherencia del modelo que he elegido.
    - La consulta 8, en la que se solicita un título de una sesión. Para poder utilizar la misma tabla y no añadir otra al sistema, el campo almacenado en el registro 8 de ESTADISTICA corresponde al identificador de la sesión, y el procedimiento que muestra el resultado de la consulta obtiene dicho identificador, que a su vez usa para obtener el título de la sesión. En este caso el tiempo de la query es 2 (en lugar de 1) pero sigue siendo tiempo constante ya que recuperamos el dato de la SESION a partir de su clave primaria (indexada por defecto).
    - La consulta 11, en la que se solicita un empleado. En el registro correspondiente de ESTADISTICA almacenamos el id del empleado (lo cual no es incorrecto), pero he considerado, como en el caso anterior, que es aceptable un tiempo de respuesta fijo 2, y se obtiene el nombre del empleado a partir del identificador.
- INCUMPLIMIENTOANYOEMPLEADO: En esta tabla, actualizada de forma automática tras cada alta o baja de un incumplimiento, almacenaremos, agrupados por empleado y año, los incumplimientos cometidos por empleado, y los asociados a su departamento. De esta manera, podremos dar respuesta a la consulta 12. Haciendo uso de los datos de la misma tabla, tras cada actualización de la misma se ejecutará la consulta 13, y los identificadores de los 5 usuarios buscados se almacenarán en la tabla ESTADITICA, en los registros 13, 14, 15, 16, 17.
- INCUMPLIMIENTOANYODEPTO: al igual que la tabla anterior, se actualiza de forma automática tras cada alta o baja de un incumplimiento, y almacena los incumplimientos asociados a cada

departamento, agrupados por año. El resultado de la consulta 14 se obtendrá directamente desde esta tabla.

Las consultas a las que hago referencia son:

1. Proceso de gestión interno que, teniendo en cuenta toda la información de que se dispone en la BD, ha tenido un mayor número de vulnerabilidades detectadas.
    1. Al dar de alta o de baja una vulnerabilidad (no se permite modificar el proceso ni el análisis asociados a la vulnerabilidad), se recalcula desde el mismo procedimiento ABM el número total de vulnerabilidades del proceso (num_vuln_detectadas): TFG.PKG_FUNCIONAMIENTO.ACTUALIZAR_VULNERABILIDADES_DE_PROCESO(pProceso, RSP);
    2. y se ejecuta mediante un disparador la query de la consulta 1, almacenando el resultado en el campo correspondiente de la tabla estadística: TFG.PKG_FUNCIONAMIENTO.ACTUALIZAR_CONSULTA_1(RSP); => TRG_ACTUALIZAR_CONSULTA_1
2. Porcentaje de vulnerabilidades que, en el momento de ejecutar la consulta, están totalmente mitigadas.
    1. Con cada vulnerabilidad que se registre, se dé de baja o se modifique, se actualiza la consulta_2 y se anota en la tabla estadística: Calculamos el total de vulnerabilidades, el total de las que están mitigadas, y hacemos el %: TFG.PKG_FUNCIONAMIENTO.ACTUALIZAR_CONSULTA_2(RSP); => TRG_ACTUALIZAR_CONSULTA_2
3. Número total de acciones de mitigación que, en el momento de ejecutar la consulta, no están totalmente acabadas.
    1. Con cada alta, baja o modificación de acción se recalcula consulta 3: TRG_ACTUALIZAR_CONSULTA_3
4. Política de seguridad que, en el momento de ejecutar la consulta, ha tenido más incumplimientos (teniendo en cuenta todos los departamentos de la empresa).
    1. Al dar de alta o de baja un incumplimiento (no se admite la modificación de la auditoria, el empleado o la política asociados al incumplimiento), actualizamos el campo num_incumplimientos de POLITICA: TFG.PKG_FUNCIONAMIENTO.ACTUALIZAR_INCUMPLIMIENTOS_DE_POLITICA(pPolitica, RSP);
    2. y recalculamos mediante un disparador sobre la tabla INCUMPLIMIENTO la query consulta 4, almacenando el resultado en el campo correspondiente de la tabla estadística: TFG.PKG_FUNCIONAMIENTO.ACTUALIZAR_CONSULTA_4(RSP); => TRG_ACTUALIZAR_CONSULTA_4

5. Dado un determinado departamento de la empresa, y teniendo en cuenta el momento de ejecutar la consulta, porcentaje de usuarios del departamento que no han acabado todas las formaciones de seguridad asignadas.
    1. Con cada alta, baja o modificación de asignación de sesión, se actualiza el campo correspondiente en DEPARTAMENTO, y en este caso NO se anota en la tabla estadística: lanzamos SIEMPRE TFG.PKG_FUNCIONAMIENTO.ACTUALIZAR_PORCENTAJE_DE_DEPARTAMENTO(id_departamento,RSP);
    2. 1º Calculamos el número total de empleados del departamento
    3. 2º Obtenemos el número total de empleados del departamento con sesiones asignadas y que no las han finalizado, y hacemos el %:
    4. El resultado de la consulta 5 se obtiene consultando directamente la tabla DEPARTAMENTO, no la registramos en la tabla ESTADISTICA
6. Teniendo en cuenta todas las auditorías externas realizadas, año en el cual se han detectado más incumplimientos (teniendo en cuenta sólo los detectados durante la auditoría).
    1. Cada vez que se dé de alta o baja un incumplimiento, obtenemos el año con más incumplimientos en los que la auditoría fuera externa, y el incumplimiento se detectase en el periodo de duración de la auditoría. Anotamos en consulta_6 de estadística.
    2. Recorremos todos los incumplimientos, teniendo en cuenta que su fecha esté entre inicio y fin de la auditoria, agrupandolos por año, y ordenándolos por año, y cogemos la primera fila. TRG_ACTUALIZAR_CONSULTA_6 => TFG.PKG_FUNCIONAMIENTO.ACTUALIZAR_CONSULTA_6(RSP);
7. Porcentaje de vulnerabilidades críticas que, en el momento de ejecutar la consulta, tienen alguna acción de mitigación abierta (que no esté en estado “acabada”).
    1. Cada alta, baja o modificación de vulnerabilidad crítica, se recalcula la query y se anota en consulta_7 de estadística.
    2. Tras cada ABM de ACCION, se comprueba si la vulnerabilidad asociada tiene acciones que no esten acabadas, y se actualiza el campo VULNERABILIDAD.id_estado_vuln: TFG.PKG_FUNCIONAMIENTO.ACTUALIZAR_ESTADO_VULNERABILIDAD(RSP);
    3. Obtenemos el número total de vulnerabilidades críticas
    4. Obtenemos el número de vulnerabilidades críticas que no estén en estado 4,'Totalmente Mitigada'
    5. Calculamos el % y lo anotamos en consulta_7 de estadística: TRG_ACTUALIZAR_CONSULTA_7 => TFG.PKG_FUNCIONAMIENTO.ACTUALIZAR_CONSULTA_7(RSP);

8. Teniendo en cuenta el último año (el anterior al año en curso), título de la sesión formativa telemática que ha tenido un porcentaje menor de participantes en total.
    1. Cada vez que se asigna una sesión, se recalcula la query y se anota en consulta_8 de estadística
    2. Obtenemos el año anterior a SYSDATE
    3. Obtenemos las sesiones del año pasado y su número de participantes, ordenando ASC y cogiendo la primera fila.
    4. Almacenamos el id de sesión en consulta 8 de estadística, al ver resultado obtendremos el título de la sesión cuyo id hemos almacenado.TRG_ACTUALIZAR_CONSULTA_8 => TFG.PKG_FUNCIONAMIENTO.ACTUALIZAR_CONSULTA_8(RSP);
9. Número de vulnerabilidades críticas detectadas internamente teniendo en cuenta todos los datos de que se dispone. Se consideran detectadas internamente si se detectaron en posterioridad al análisis realizado al inicio del proyecto por la consultora externa.
    1. Con cada ABM de vulnerabilidad, si es crítica, se recalcula la query y se anota en consulta_9 de estadística.
    2. Contamos la vulnerabilidades que son críticas, y cuyo análisis asociado sea de tipo INTERNO, y registramos en consulta 9 de estadística. TRG_ACTUALIZAR_CONSULTA_9 => TFG.PKG_FUNCIONAMIENTO.ACTUALIZAR_CONSULTA_9(RSP);
10. En el momento de ejecutar la consulta, porcentaje de acciones de mitigación en el sistema que están en los estados “en proceso” o “en revisión”.
    1. Cada alta, baja o modificación de acción de mitigación, se recalcula la query y se anota en consulta_10 de estadística.
        i. Obtenemos el Nº total de acciones
        ii. Obtenemos el Nº total de acciones en estados 2 o 4
        iii. Calculamos el % y anotamos en consulta 10 de estadística
        iv. TRG_ACTUALIZAR_CONSULTA_10 => TFG.PKG_FUNCIONAMIENTO.ACTUALIZAR_CONSULTA_10(RSP);
11. Teniendo en cuenta todas las acciones de mitigación en estado “en proceso”, persona responsable con más acciones asignadas.
    1. Para cada alta, baja o modificación de acción, se ejecuta la query y se anota en consulta_11 de estadística.
        i. Obtenemos las acciones "en proceso", agrupando por responsable DESC, y cogemos el primero
        ii. Anotamos en consulta 11 el id de empleado, al ver consulta obtendremos el nombre.
        iii. TRG_ACTUALIZAR_CONSULTA_11 => TFG.PKG_FUNCIONAMIENTO.ACTUALIZAR_CONSULTA_11(RSP);

4.6. Pruebas de validación.

Para comprobar el correcto funcionamiento del sistema, se han introducido datos de prueba (mediante el script 11_ALTAS_TABLAS_CORRECTAS.sql) y se ha comprobado que el resultado es el esperado. Se puede comprobar que los datos son los esperados, que quedan reflejados en el fichero 11_RESULTADO.txt.

Por otro lado, para comprobar que todos los procedimientos desarrollados cumplen con todas las restricciones que se les han impuesto (control de valores nulos, de excepciones, de restricciones de integridad que no se hayan controlado mediante restricciones directas en las tablas) se han ejecutado 322 pruebas unitarias, todas las cuales han sido correctas.

Todos los procedimientos han sido probados tratando de “colar” valores tanto nulos como erróneos en cada parámetro admitido. Dado que, durante el desarrollo, se han ido probando todos los procesos, el resultado ha sido de un 100% de éxito.

Con el script 12_PRUEBAS_UNITARIAS.sql se puede comprobar, mirando la tabla de log, el resultado de todas las pruebas. Ninguno de los datos se añadirá al sistema.

5. Conclusiones

Aunque no era este el tipo de proyecto que pensaba realizar en el trabajo de fin de grado, ha acabado resultando interesante y formativo, aunque se trate de un área en la que no tenía experiencia alguna, y en la que probablemente no volveré a trabajar.

Como DBA suelo estar en el lado contrario, discutiendo con los programadores sobre si el culpable de un bloqueo o la lentitud de una instancia es el código o la instancia en si. Este proyecto me ha ayudado a ponerme un poco en su piel, aunque no ha cambiado mi opinión sobre quien suele ser el culpable de los problemas.

Algo que desde luego no es posible ignorar es la importancia tanto de una correcta planificación del trabajo, como de el cumplimiento de dicha planificación. Al principio del proyecto realicé una planificación sin saber realmente en qué fases iba necesitar más tiempo. Y aunque finalmente no la he cumplido, lo que tengo claro es que no era una planificación correcta, y que para hacerla correctamente hay que tener bastante más experiencia.

Igualmente importante en el correcto desarrollo del proyecto es una fase de análisis totalmente correcta. En mi caso, los errores en esa fase han ocasionado múltiples retrasos ocasionados por los cambios que ha habido que ir introduciendo en el modelo.

Otra conclusión importante que saco del desarrollo de este proyecto es que, a pesar de la edad y la experiencia previa (repito, nula en este campo), con constancia es posible obtener los conocimientos necesarios para enfrentase a cualquier proyecto nuevo.

6. Glosario

A

Atributo
Cada una de las propiedades que posee la entidad de la que se desea guardar información.

B

BBDD
Una base de datos o banco de datos es un conjunto de datos almacenados sistemáticamente para su posterior uso.

C

Cardinalidad
El número de filas relacionadas de cada uno de los objetos en la relación.

Clave primaria
Campo o combinación de campos que identifica de forma única un registro en una tabla

Clave ajena
Campo o combinación de campos de una tabla que hacen referencia a un registro de otra tabla.

D

Datafile
Los ficheros físicos en los que se almacenan los objetos que forman parte de una base de datos Oracle. Cada uno pertenece a un único tablespace.

Diagrama de Gantt
Herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

Disparador
Comportamiento que se ejecuta cuando se realiza una operación sobre una tabla.

L

Log:
Registro de las acciones realizadas sobre la base de datos.

M

Modelo E/R
Herramienta para el modelo de datos, la cual facilita la representación de entidades de una base de datos.1 Fue definido por Peter Chen en 1976.

Modelo lógico
Modelo de datos lógico, que describe los datos con el mayor detalle posible, independientemente de cómo se implementarán físicamente en la base de datos.

O

Oracle
Sistema de gestión de bases de datos de tipo objeto-relacional (ORDBMS, por el acrónimo en inglés de Object-Relational Data Base Management System), desarrollado por Oracle Corporation.

P

PL/SQL
(Procedural Language/Structured Query Language) Lenguaje de programación incrustado en Oracle.

Procedimiento
Conjunto de instrucciones en PL/SQL.

Prueba unitaria
Método para comprobar el correcto funcionamiento de una unidad básica de código, como un procedimiento almacenado.

Prueba integrada
Método para comprobar el correcto funcionamiento de procesos más complejos incluidos en el sistema.

R

Registro
Conjunto de toda la información relativa a un elemento que se almacena en una tabla de la base de datos.

S

SGBD
Conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos.

SQL Developer
Entorno de desarrollo integrado y gratuito que simplifica el desarrollo y la gestión de Oracle Database en implementaciones tradicionales y en la nube. Incluye el módulo Data Modeler.

Script
Sentencia o conjunto de sentencias en lenguaje SQL que realiza acciones sobre la base de datos

String
Tipo de datos que es una cadena de caracteres.

T

Tablespace
: Unidad lógica de almacenamiento donde pueden ser guardados los datos de una base de datos. Está compuesto por ficheros físicos (datafiles).

U

Uml
: Lenguaje para describir modelos de sistema.

7. Bibliografía

Se han realizado multitud de consultas a la documentación aportada por el profesorado para la realización de este TFG y se ha consultado de forma frecuente los siguientes enlaces:

[1] Documentación de referencia sobre Oracle 12.2 PL/SQL https://docs.oracle.com/en/database/oracle/oracle-database/12.2/lnpls/index.html [Último acceso: 2 Junio 2022]

[2] Documentación de referencia sobre Oracle SQL Developer https://www.oracle.com/es/database/technologies/appdev/sqldeveloper-landing.html [Último acceso: 26 Marzo 2022]

[3] Documentación de referencia sobre Oracle Sql Developer Data Modeler https://www.oracle.com/es/database/technologies/appdev/datamodeler.html [Último acceso: 10 Marzo 2022]

[4] «Requisito Funcional» https://en.wikipedia.org/wiki/Functional_requirement [Último acceso: 10 Abril 2022].

[5] «Requisito no funcional» https://es.wikipedia.org/wiki/Requisito_no_funcional [Último acceso: 10 Abril 2022].

[6] «PL/SQL» https://es.wikipedia.org/wiki/PL/SQL. [Último acceso: 20 Marzo 2022].

[7] «Oracle» https://www.oracle.com/technical-resources [Último acceso: 2 Junio 2022].

8. Anexo

La lista de scripts a cargar para levantar el sistema es Ejecutados en este orden):

1_TABLESPACE_Y_ESQUEMA_PRINCIPAL.sql (ejecutar como **sys**)
2_CREACION_TABLAS.sql (ejecutar como **tfg**)
3_PKG_FUNCIONAMIENTO.sql (ejecutar como **tfg**)
4_PKG_FUNCIONAMIENTO_BODY.sql (ejecutar como **tfg**)
5_PKG_ABM.sql (ejecutar como **tfg**)
6_PKG_ABM_BODY.sql (ejecutar como **tfg**)
7_PKG_ESTADISTICAS.sql (ejecutar como **tfg**)
8_PKG_ESTADISTICAS_BODY.sql (ejecutar como **tfg**)
9_SECUENCIAS_Y_DISPARADORES.sql (ejecutar como **tfg**)
10_ROLES_Y_USUARIOS.sql (ejecutar como **sys**)
11_ALTAS_TABLAS_CORRECTAS.sql (ejecutar como **tfg**)