TAREA EVALUABLE 3

Bases de Datos
CFGS DAW

Pau Miñana Climent
2020/2021

INTRODUCCIÓN

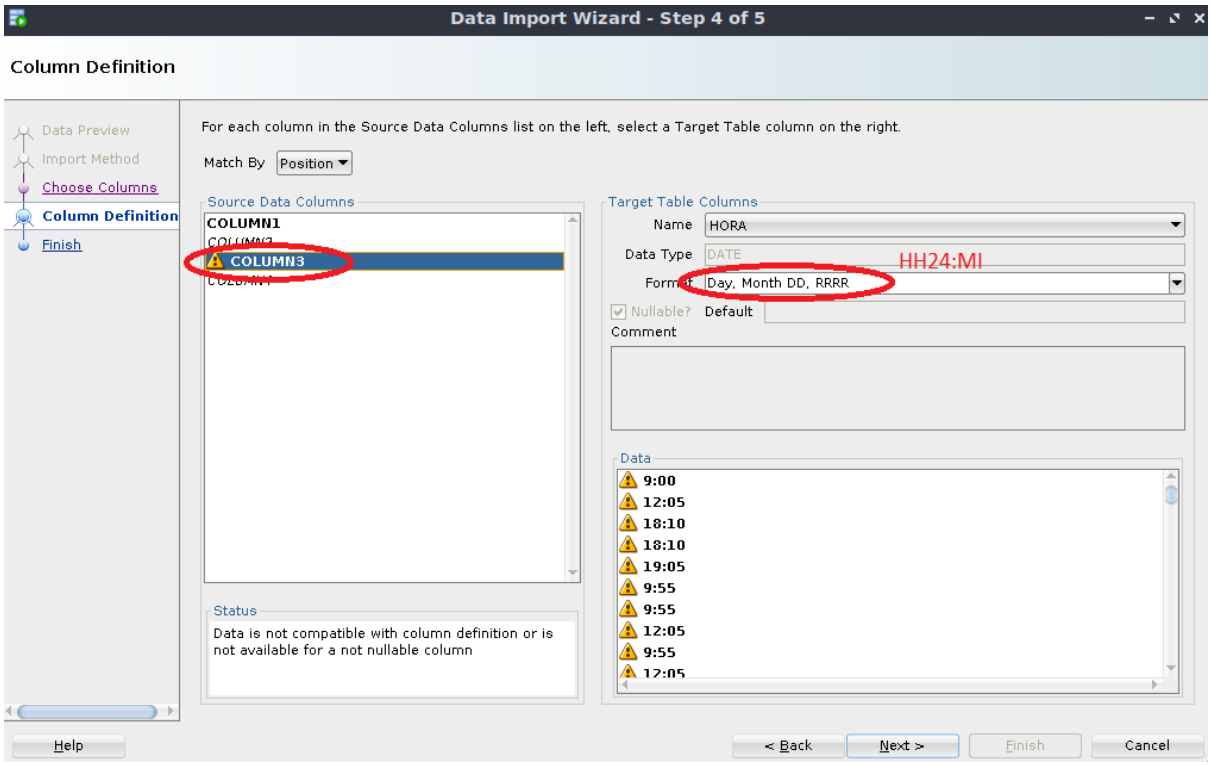
Leed detalladamente el documento y entregad en un documento de texto el código SQL o PL/SQL usado para responder a los distintos apartados. Estos vendrán marcados siempre con un numero para facilitar su localización y que no os dejéis nada. Podéis añadir los comentarios o reflexiones que creáis oportunos a vuestra respuesta.

Para la siguiente actividad evaluable se realizará la gestión de usuarios, permisos y funciones de la supuesta aplicación del CEEDCV que hemos estado estudiando. Es decir, como acceden los usuarios, que tipos hay, que permisos tienen, a qué información pueden acceder y, puesto que el usuario final no necesita entender nada de bases de datos, qué funciones y procedimientos habría que crear para que se pudiese acceder a las distintas opciones bien desde una supuesta aplicación web o de otro tipo. Obviamente el desarrollo de dicha app web o GUI se escapa de los contenidos de este módulo y no se va a desarrollar, pero puede resultar un proyecto interesante para que probéis como práctica en los módulos de 2º. Además, la conexión con la BD se podría hacer desde la aplicación directamente usando distintas opciones, pero como debemos hacerla con los conocimientos que hemos adquirido en el curso nos limitaremos a gestionar una Base de Datos de Oracle y realizar los procesos en PL/SQL.

Tened en cuenta que en un caso real, puede que no dispusierais de más información que la de las anteriores actividades evaluables y quedaría a vuestro criterio toda esta organización y la decisión de qué funciones y características podría tener y como aplicarlo. Obviamente, esto os requeriría un tiempo del que probablemente no dispongáis y resultaría más difícil de evaluar, así que os marcaré qué debéis hacer con incluso alguna guía extra en el proceso… pero resulta un ejercicio útil que antes de seguir leyendo este documento dediquéis un rato a plantear como resolveríais el problema y qué decisiones tomaríais. Así podéis comparar con lo que os planteo y ver los pros y los contras de las elecciones.

Para crear la base de datos disponéis de un script que crea un usuario evaluable3 con un tablespace propio y los archivos csv. Son prácticamente los mismos que en la tarea anterior, se ha cambiado alguna tupla pero el único cambio relevante es en el campo día de TI que ahora va de 2 a 6 ya que este es el formato en el que Oracle reconoce los días de lunes a viernes por defecto.

Podéis usar la guía de la actividad anterior para montarla. Sólo un aviso, al importar el csv de la tabla TI, en el campo hora el SQL Developer podría no reconocer el formato automáticamente como en los otros casos, así que después de dar en siguiente 3 veces veréis una señal de advertencia y debéis situaros en la columna 3 y cambiar el formato a “HH24:MI”



Gestión de Usuarios (2,5 puntos)

Para empezar con la gestión de usuarios la opción más viable es crear 2 roles de usuario: Alumno y Profesor y asignar estos roles a los usuarios, pero simplemente no conceder permisos sobre las tablas a NINGÚN usuario ni a estos roles. El único permiso del que deben disponer todos los usuarios es para poder crear sesión.

Aunque a priori esto puede parecer extraño tiene una explicación muy sencilla, ni alumnos ni profesores tienen acceso real a toda la información de las tablas, sólo a la que les corresponde… un alumno puede ver a sus profesores, pero no a todos los profesores del centro, un profesor sólo a sus alumnos, etc… Además como se ha comentado, no pueden ni deben realizar sus consultas en SQL libremente, sólo pueden realizar ciertas consultas preestablecidas por nosotros.

Así pues, los permisos que tendrán los roles Alumno y Profesor serán sólo para lanzar ciertos procedimientos que nosotros diseñamos con el usuario administrador de la BD. De este modo aunque los usuarios no tienen permiso para consultar nada la función que lanzan sí lo tiene, y así pueden acceder sólo a la información que nosotros hemos establecido.

Primer problema con que nos encontramos, los usuarios de Oracle no pueden empezar con un numero y nuestra CP de usuario es DNI. Para simplificar el proceso vamos a:

1. Añadir el campo usuario a la tabla usuarios, de tipo VARCHAR(6), con la restricción de que el nombre de usuario no se puede repetir
2. Rellenar mediante PL/SQL todos los usuarios con las 3 primeras letras del nombre y las 3 primeras letras del apellido1 (es una mala decisión de diseño pues es fácil que se repitan, pero en este caso nos vale). Para ello podéis usar la función SUBSTR(cadena,num_inicio,num_fin) que extrae "substrings" de una cadena.
3. Crear los roles profesor y alumno y un usuario alumno de ejemplo al que asignaremos sólo el permiso para crear una sesión y el rol correspondiente. Obviamente el nombre del usuario será el contenido en "usuario" y su contraseña la establecida en "password". Este alumno nos servirá para probar los procedimientos posteriormente, puesto
4. Dar al rol alumno permiso para ejecutar las funciones procedimientos que se correspondan (esto no podrá hacerse hasta el final cuando ya se hayan creado los mismos, así que de momento podríais dar acceso para ejecutar cualquier procedimiento y función bien al alumno creado o temporalmente al rol y así poder probarlos).

Gestión de la BD (3 puntos)

Para la gestión de la base de datos simplemente vamos a crear un par de Triggers y funciones para superar las limitaciones hasta ahora de nuestra base de datos.

1. El día de la semana en la tabla TI se muestra en forma numérica. Se ha adaptado la tabla a Oracle de modo que 2=lunes...6=viernes, pues así es como Oracle nos devuelve los días numéricamente si los extraemos de una fecha. Para poder mostrar el horario semanal de forma fácilmente comprensible, crea la siguiente FUNCIÓN que se podrá usar posteriormente en los procedimientos:
   evaluable.dia_sem: Se le pasa como parámetro el número del día (de 2 a 6) y devuelve, el día de la semana escrito en Valenciano o Castellano.
2. Un trigger para la actualización en cascada del departamento en profesores cuando se cambie en la tabla de departamentos;
3. Un trigger para la actualización en cascada del cod_mod en imparten y en matriculados cuando se cambie en modulos.
4. Un trigger para evitar que un profesor pueda estar matriculado como alumno en un módulo que lo imparta alguien de su mismo departamento (tanto al añadir datos como al cambiarlos). Recordad que ya existe un profesor que incumple esta restricción. Analizad los pros y los contras de vuestra respuesta.

Procedimientos de Alumnos (4,5 puntos)

Tened en cuenta que los procedimientos tanto de profesores como de alumnos deben mostrar un resultado personalizado para cada usuario concreto. Esto requiere el uso del parámetro "user", que nos devuelve el nombre del usuario. De este modo aunque el procedimiento ejecutado sea el mismo, podemos hacer que la consulta devuelva lo que queramos adaptado al usuario (no olvidéis que el usuario no es la CP para la base de datos, necesitáis obtener el dni correspondiente al usuario).

Además, para una aplicación, los resultados obtenidos por las consultas deberían almacenarse para que la aplicación pudiese usarlos. En nuestro caso como dicha aplicación no existe, nos limitaremos a mostrar por pantalla los resultados. Por otro lado, cabe tener en cuenta que por cuestiones de tiempo, en esta tarea sólo vamos a desarrollar los procedimientos para ciertas funcionalidades y sólo para los alumnos. Para un funcionamiento completo se requiere añadir alguna funcionalidad más. Los procedimientos que se van a implementar son los siguientes:

1. *evaluable.muestra_modulos*: Muestra los modulos en los que esta matriculado el usuario que lo ejecuta.
2. *evaluable.muestra_ti*: Se supone que al pinchar en los módulos devueltos por la función anterior se puede ver el horario de ti del profesor. Por tanto esta función muestra el horario SEMANAL del profesor(o profesores) del módulo que se le pasa como parámetro. Aquí podéis usar la función *evaluable.dia_sem* para mostrar el día de la semana como texto. Para mostrar la hora de la tutoría puesto que por defecto Oracle muestra en los campos DATE sólo la fecha, no se vería correctamente. Podéis usar la "TO_CHAR(hora, 'HH24:MI')" en vez de hora en la consulta y así nos devuelve una cadena que contiene la sólo la hora.
3. *evaluable.muestra_reservas*: Muestra las reservas de ti que ha hecho el usuario. Pueden ser todas o sólo las que aún no se han realizado. Para obtener la fecha actual se puede hacer con "TO_DATE(SYSDATE())". Se recomienda añadir to_date para asegurar que el resultado de sysdate sea una fecha que se comparare bien con fechas dentro de una consulta. En caso de mostrar las que aún no se han realizado debéis insertar vosotros alguna para poder probarlo, ya que todas las ti en la BD son anteriores a la fecha actual
4. *evaluable.anula_tutoría*: Borra una tutoría ya existente del usuario actual. Obviamente necesita la fecha y id_ti como parámetros.
5. *evaluable.muestra_ti_libres:* Muestra las ti libres en las próximas 2 semanas del módulo que se le pasa como parámetro. Es decir, todas las ti del módulo en ese periodo, excepto las que ya existen en reserva_ti. Como antes, para probar la función adecuadamente se deben añadir reservas, puesto que no hay ninguna para fechas actuales y por tanto siempre estarán todas libres.

Recomendaciones para el punto 5: Este procedimiento os puede resultar un poco más enrevesado que los anteriores, así que os doy unas recomendaciones sobre como lo resolvería yo. Podéis ignorarlas, puesto que la solución no es única, o intentar resolverlo primero por vuestra cuenta y en caso de no lograrlo, volver aquí y leerlo entonces:

Un bucle de 0 a 13 puede ayudarte a crear el horario con todas las ti del módulo de las próximas semanas. Es más fácil usando una tabla auxiliar donde se guardará la información en vez de guardarlas en un vector o cursor y posteriormente borrar de la tabla las tutorías que ya estén reservadas. La tabla no mejor creala anteriormente, no cada vez que se ejecute el proceso, y recuerda borrar el contenido al terminar (o al principio) para que no se solapen al usar el procedimiento varias veces.

De nuevo necesitarás "TO_DATE(SYSDATE())" o alguna variante para este procedimiento. Para extraer el día de la semana de una fecha se puede usar "TO_CHAR(fecha,'D')", esto extrae el día como un numero que cumple lo comentado anteriormente (2=lunes...6=viernes).

RECORDAD UNA VEZ ACABADO REGRESAR AL PUNTO 4 de Gestión de Usuarios y modificar los permisos apropiadamente