PROYECTO FINAL
ESPECIFICACIONES.

1.1. INSTRUCCIONES GENERALES.

- El proyecto se puede realizar en equipos hasta de 2 personas.
- De forma opcional se puede desarrollar una aplicación web o standalone empleando el lenguaje y herramientas de su preferencia.
- No se requiere realizar reporte alguno. Solo se deberá presentar:
  - Hoja de asignación de proyecto con rúbrica. Deberá ser llenada con los datos de los integrantes del equipo.
  - Diagrama ER empleando Notación Chen (Opcional)
  - Modelo Relacional empleando notación Crow's Foot o IDEF1X.
- Durante la entrega del proyecto se deberá realizar una explicación y justificación de las actividades realizadas en cada uno de los scripts.

Importante:
Se hará una selección aleatoria para designar al integrante que realizará la explicación para cada script SQL desarrollado.

1.2. ACTIVIDADES A REALIZAR PARA EL CASO DE ESTUDIO ASIGNADO.

1.2.1. Construcción del modelo ER (Opcional)

El diagrama deberá contener todos los elementos vistos en las prácticas de modelado: cardinalidades, tipos de relación.

1.2.2. Construcción del modelo relacional.

El modelo deberá contener todos los elementos vistos en prácticas de modelado: cardinalidades, tipos de datos, restricción NULL/NOT NULL. La notación a emplear es libre: Crow's Foot o IDEF1X.

Tip: El caso de estudio asignado esta diseñado de tal forma que el modelo deberá contener los siguientes elementos:

- Atributos derivados
- Atributos multivalorados
- Dependencias de identificación
- Al menos una relación 1:1 y una relación M:N con atributos en la relación.
- Relaciones recursivas.
- Jerarquía de supertipo y subtipos.
- Manejo de datos con histórico.

1.2.3. Script s-01-usuarios.sql

Todos los Scripts SQL deberán contener el encabezado empleado en las prácticas:

```
--@Autor(es):       <Nombre de los integrantes>
--@Fecha creación:  dd/mm/aaaa
--@Descripción:     <breve descripción del contenido y propósito del archivo>
```

- Este script contendrá la definición de 2 usuarios: `<iniciales>_proy_invitado` y `<iniciales>_proy_admin`, donde `<iniciales>` corresponden a la primera letra de cada apellido paterno de cada integrante. El usuario `admin` será el dueño de todos los objetos del caso de estudio.
- Incluir en el script la definición de 2 roles:
  - `rol_admin` Contendrá todos los roles necesarios para poder implementar el caso de estudio asignado.
  - `rol_invitado`. Solo deberá tener permisos para crear sesiones.
- Asignar a los usuarios admin e invitado los roles `rol_admin` y `rol_invitado` respectivamente.

1.2.4. Script s-02-entidades.sql

- Este archivo contendrá el código DDL empleado para crear las tablas del caso de estudio. El código ***no deberá*** ser creado con ER-Studio. Todas las restricciones deberán tener un nombre asignado.
- Se deberá hacer uso de todos los tipos de restricciones vistas en clase: `unique`, `check`, `primary key`, `foreign key`.
- La definición de algunas tablas deberá incluir el uso de `default`, seleccionar al menos un caso donde pueda aplicarse.
- Seleccionar al menos un caso donde sea factible emplear una columna virtual. De ser posible se pueden crear nuevas tablas que hagan uso de este tipo de columnas.
- Sugerencias para su implementación:
    - Verificar la existencia de atributos derivados. De no existir se puede agregar algún campo para reproducir el escenario. Los atributos derivados son excelentes candidatos para crear tablas virtuales.

1.2.5. Script s-03-tablas-temporales.sql

- Diseñar al menos un escenario donde se haga uso de una o más tablas temporales. Se pueden crear nuevas tablas en caso de ser necesario.
- Sugerencias para su implementación:
    - Suponer que se desea consultar datos de un conjunto de tablas, pero por el nivel de normalización que presentan, se decide crear una tabla temporal para aplicar un proceso de desnormalización e insertar los datos de forma temporal. Por ejemplo, suponer que se tiene una Jerarquía Super tipo – Subtipos. Se decide crear una tabla temporal que contendrá todos los datos de la Jerarquía.
    - Simular un carrito de compras. El contenido del carrito de compras se puede guardar en una tabla temporal y en el momento que el cliente decida comprar, los datos se consultan de la tabla temporal y se insertan en las tablas permanentes.
    - Suponer que se tiene que realizar ciertos cálculos que provienen de la consulta de varias tablas permanentes como son: promedios, costos totales, descuentos, ventas, etc. El resultado de estos cálculos se puede guardar en una tabla temporal para ser consultados por algún cliente.

1.2.6. Script s-04-tablas-externas.sql

- Diseñar un escenario donde se haga uso de una o más tablas externas.
- Sugerencias:
    - Se tiene un archivo de texto y se desea consultar sus datos sin tener que cargar los datos al interior de la BD. El archivo de texto deberá contener una lista de registros separados por algún carácter, por ejemplo: “#”, “*”,”,”, etc.

1.2.7. Script s-05-secuencias.sql

- Este archivo contendrá la definición de ***todas*** las secuencias necesarias para poder insertar registros en tablas que requieran la generación de valores secuenciales. Todas las secuencias deberán iniciar en un número seleccionado por el equipo.

1.2.8. Script s-06-índices.sql

- Realizar un análisis y seleccionar algunos casos donde el uso de índices pudiera ser adecuado:
    - Uso de índices Non Unique para mejorar el desempeño de las consultas, por ejemplo, identificar posibles campos que son empleados frecuentemente en búsquedas (claves, números de cuenta, rfc, curp, email, etc). Considerar de forma adicional, indexar llaves foráneas que participan en operaciones JOIN frecuentes.
    - Uso de índices Unique empleados para verificar duplicidad de valores.
    - Uso de un índice compuestos tipo Unique, empleados para validar duplicidad de combinaciones de valores de las columnas involucradas en el índice.
    - Uso de uno o más índices basados en el uso de funciones. Por ejemplo, definir un índice con la función `lower` o `upper` que permita agilizar las búsquedas sin importar si los datos se encuentran capturados en minúsculas o mayúsculas.

1.2.9. Script s-07-sinonimos.sql

- Generar 3 o más sinónimos públicos que le pertenezcan al usuario admin. Los sinónimos deberán estar disponibles para que otros usuarios puedan emplearlos, por ejemplo, el usuario invitado.
- Generar las instrucciones necesarias para que el usuario admin otorgue permisos de lectura al usuario invitado en al menos 3 entidades.
- Generar 3 o más sinónimos que le pertenezcan al usuario invitado. Dichos sinónimos deberán ser empleados para leer el contenido de las tablas a las que el usuario admin le otorgó permisos de lectura.
- Finalmente, suponer que un software necesita que todas las tablas del proyecto tengan un prefijo formado por 2 caracteres: `XX_<nombre_tabla>`. Para implementar este requerimiento se creará un sinónimo privado para cada tabla. Se recomienda realizar un programa anónimo PL/SQL empleando SQL dinámico para evitar escribir manualmente los sinónimos.

1.2.10. Script s-08-vistas-sql

- Realizar un análisis y crear 3 o más vistas. Analizar posibles escenarios donde una vista podría ser adecuada.
- Posibles escenarios:
    - Generar una vista para ocultar la complejidad de la consulta en la que se involucran varias tablas: joins, funciones de agregación, etc.
    - Generar una vista para proteger el acceso a columnas consideradas como 'delicadas': contraseñas, números de tarjeta, etc. En este escenario, el usuario admin puede otorgar permisos al usuario invitado para que pueda leer el contenido de una vista que no contiene columnas privadas.

1.2.11. Script s-09-carga-inicial.sql

- Este script contendrá algunos datos de prueba y carga inicial para poder ilustrar el correcto funcionamiento de la base de datos.
- No existe límite en cuanto a la cantidad de registros, con unos cuantos registros es suficiente.
- El script deberá hacer uso de las secuencias para realizar las inserciones.
- Tener en cuenta que un valor generado por una secuencia pudiera emplearse varias veces para insertar registros con llaves foráneas. En este caso se recomienda emplear un Script PL/SQL y el uso de variables para reutilizar sus valores.

1.2.12. Script s-10-consultas.sql

- Este archivo contendrá 5 o más consultas. El criterio es libre. Se debe emplear el uso de los siguientes elementos.
    - joins (inner join, natural join, outer join)
    - funciones de agregación (`group by` y `having`)
    - algebra relacional (operadores set: `union, intersect, minus`).
    - Subconsultas
    - Consulta que involucre el uso de un sinónimo
    - Consulta que involucre el uso de una vista
    - Consulta que involucre una tabla temporal
    - Consulta que involucre a una tabla externa.
- No es necesario crear una consulta por cada elemento. En una misma consulta pueden incluirse varios de los elementos anteriores.

1.2.13. Scripts s-11-tr-<nombre-trigger>.sql

Empleando la nomenclatura de nombrado de estos scripts, generar los siguientes triggers.

- Al menos 2 triggers tipo row level. Posibles escenarios:
    - Validar reglas de negocio que no pueden ser implementadas por restricciones básicas como son unique, check. Si la regla de negocio no se cumple, el trigger puede lanzar una excepción para evitar que la operación DML se ejecute.
    - Realizar alguna operación DML sobre alguna tabla como consecuencia de la ocurrencia de un evento en otra, por ejemplo, actualizar el número de existencias de un producto cada vez que se confirma una orden de compra, actualizar el número de vacantes cada vez que se ocupa un lugar, etc.
    - Realizar la replicación de datos en tiempo real. Por ejemplo, aplicar las operaciones que se ejecutan en una tabla en otra tabla llamada tabla de respaldo o tabla de replicación.
    - No es necesario que en cada trigger se programen los 3 eventos: `insert , update y delete`. Estos 3 eventos pueden ser programados en diferentes triggers. La entrega se considera incompleta si en ninguno de los trigger se implementa alguno de estos 3 escenarios.
- Diseñar un escenario donde sea útil el uso de un compund trigger.
    - Tomar como referencia las prácticas complementarias y los apuntes del tema 10. Los escenarios típicos son: empleado para resolver el problema de tablas mutantes, y escenario para mejorar el desempeño con consultas que pueden actualizar grandes cantidades de datos.

- Cada trigger deberá ser incluido en un archivo SQL. No programar escenarios triviales como mandar mensajes a pantalla cuando ocurre un evento, u operaciones simples. Se recomienda validar los escenarios con el profesor.

1.2.14. Scripts s-12-tr-<nombre-trigger>-prueba.sql

- Para cada uno de los triggers creados anteriormente se deberá crear un script SQL o un programa PL/SQL que ejecute un escenario para validar el correcto funcionamiento del trigger. El script deberá ser lo más automatizado posible y evitar la intervención manual. Emplear como referencia los archivos de prueba empleados en el tema 10. El script deberá imprimir los mensajes o consultas necesarias en consola para validar su funcionamiento de forma visual.

1.2.15. Scripts s-13-p-<nombre-procedimiento>.sql

- Realizar un análisis y diseñar escenarios donde sea adecuado el uso de procedimientos almacenados. Generar 2 o más procedimientos
- Posibles escenarios:
  - Procedimiento que puede ser invocado por un usuario para implementar una funcionalidad del caso de estudio. Por ejemplo, para crear una nueva orden de compra. El procedimiento aceptará los parámetros necesarios para realizar la creación de la orden de compra. El procedimiento realiza varias inserciones en tablas diferentes. A nivel general, el procedimiento oculta todas las operaciones que se requieren para ejecutar la funcionalidad del caso de estudio que implica la ejecución de varias sentencias SQL.
  - Procesamiento de Datos. El procedimiento realiza una consulta empleando un cursor y por cada registro encontrado realiza alguna acción, por ejemplo, exportar datos a un archivo de texto, realizar alguna operación DML en otra tabla por cada registro obtenido por la consulta.
- No realizar procedimientos triviales como inserciones simples a tablas, o consultas sencillas a tablas. Se recomienda validar los escenarios con el profesor.

1.2.16. Scripts s-14-p-<nombre-procedimiento>-prueba.sql

- Para cada uno de los procedimientos creados se deberá generar un script de prueba que permita validar su funcionamiento. De forma similar a los scripts de prueba para triggers, se deberá evitar al máximo la intervención manual para verificar su funcionamiento.
- Estos programas deberán hacer un uso adecuado de las transacciones. Es decir, si el programa realiza operaciones DML, se deberá tener una estructura similar a la siguiente:

```
declare
  begin
    ----instrucciones, invocación de procedimientos, etc.
    ---commit al final de las operaciones, todo se ejecutó correctamente.
    commit;

  exception
    when others then
    --algo salio mal, se aplica rollback
    rollback;

end;
/
```

1.2.17. Scripts s-15-fx-<nombre-funcion>.sql

- Crear al menos 3 scripts, cada uno con la definición de una función. Algunos escenarios:
  - Realizar el calculo del total de una factura con base a los parámetros de entrada.
  - Procesar cierto número de parámetros para formar cadenas, por ejemplo, obtener un numero de serie a partir del tipo de auto, marca, modelo, etc.
  - Desarrollar funciones para leer archivos binarios de disco y regresar un objeto BLOB/CLOB para ser insertado en una tabla.

1.2.18. Scripts s-16-fx-<nombre-funcion>-prueba.sql

- Para probar las funciones del punto anterior, se puede realizar lo siguiente:
  - Crear un script o programa PL/SQL que la invoque para verificar su funcionamiento.
  - Emplear la función en algún otro programa (trigger, procedimiento, bloque anónimo). En este caso no será necesario crear un script especial para validar su funcionamiento.

1.2.19. Scripts s-17-lob-<nombre-programa>.sql

- El proyecto deberá incluir al menos un escenario donde se haga uso de datos LOB (columnas CLOB o BLOB). Se deberá generar un programa PL/SQL (procedimiento, trigger o programa anónimo) que maneje este tipo de dato. Posibles escenarios:
  - Suponer que en un directorio existen varias fotos de los empleados cuyo nombre de la imagen corresponde al id del empleado. A través de un programa PL/SQL se leerá cada foto y será actualizada en la base de datos.
  - Suponer que se tienen archivos de texto que corresponden a los reportes de los estudiantes, se desea que ese texto se almacene en la base de datos en una columna CLOB.
  - Crear una función que reciba como parámetro el nombre de un archivo binario. La función leerá el archivo binario y regresará un objeto tipo BLOB/CLOB. Esta función será empleada para realizar inserciones de datos BLOB/CLOB. Por ejemplo, suponer que se ha creado una función llamada `lee_foto`, una sentencia `insert` podría hacer uso de ella de la siguiente manera:

```
insert into empleado(empleado_id,foto)  values (1,lee_foto('mifoto.png'));
```

  - Crear un trigger en el que cada vez que se haga la inserción o actualización de un dato BLOB/CLOB, su contenido sea copiado o exportado a un archivo en el servidor. El trigger emplearía el identificador o valor de la PK para ser asignado como nombre del archivo.
- No olvidar que este tipo de objetos requieren la creación de un objeto tipo `directory`.
- En caso que el modelo relacional asignado no contenga columnas BLOB/CLOB, agregar al menos una columna en algún objeto que pueda requerirlo, por ejemplo, la foto de un auto, el texto de un artículo, el pdf de un libro, el pdf que representa el comprobante de pago, etc.

1.2.20. Scripts s-18-lob-<nombre-programa>-prueba.sql

- Similar a los triggers y procedimientos, deberá existir un script de prueba que verifique el correcto manejo de objetos LOB. Para validar la correcta inserción, actualización o eliminación de estos objetos bastará con imprimir la longitud del objeto. Por ejemplo:

```
select empleado_id, dbms_length(foto) as longitud_foto
from empleado
where empleado_id = 1;
```

1.2.21. Scripts s-19-cur-<nombre-programa>.sql

- El proyecto deberá incluir el uso de uno o más cursores. El cursor puede aparecer en alguno de los procedimientos o triggers mencionados en los puntos anteriores, o si se desea, se puede crear un nuevo programa PL/SQL.

Notar que en un solo escenario se pueden implementar varios requerimientos: Por ejemplo, en un trigger se puede incluir el uso de una función creada por el alumno, el uso de cursores, o el manejo de datos BLOB. Lo mismo puede ocurrir para un procedimiento. Si esto ocurre ya no será necesario crear cada uno de los scripts antes mencionados.

1.3. PRESENTACIÓN Y EJECUCIÓN DEL PROYECTO.

Para realizar la demostración del correcto funcionamiento del proyecto, crear los siguientes archivos:

1.3.1. Script s-00-main.sql

- Este script deberá invocar a todos los scripts necesarios para realizar las siguientes acciones:
  - Eliminar al usuario en caso que exista
  - Crear todos los objetos necesarios.
  - Los triggers pueden ser omitidos o se pueden ejecutar hasta el final en caso de requerirse. Esto dependerá de los escenarios de prueba a verificar (Los triggers no deberán ser probados a través de la carga inicial, se deberá emplear los scripts de prueba).

- Antes de ejecutar el script principal, se deberá mostrar el contenido de cada script. Contestar las preguntas realizadas por el profesor.
- Posterior a la serie de preguntas, ejecutar el script.
- Se tomará en cuenta el correcto uso de usuarios, es decir, utilizar a los usuarios Oracle del sistema operativo y SYS de la base de datos solo para cuestiones administrativas.
- Ejecutar los scripts de prueba para validar, triggers, procedimientos, objetos LOBs, funciones y cursores.
- Finalmente ejecutar el script `resultados-proyecto-final.sql` que se encuentra en la carpeta compartida `BD/teoria/proyecto-final`

1.4. APLICACIÓN WEB/STANDALONE

- En caso de haber desarrollado aplicación, se deberá realizar un pequeño demo de su uso. No se requiere programar todo el caso de estudio. Basta con programar un caso de uso pequeño. Por ejemplo: Registrar artículo, registrar viaje, consultar reservación, etc.