PROPUESTA DE SOLUCIÓN

PRÁCTICA 2 - DISEÑO DE BBDD

Presentación

Esta práctica sigue la temática de la primera práctica de la asignatura y toma como punto de partida una aproximación a la solución del diseño propuesto en dicha práctica.

A partir de la carga del modelo físico y de un conjunto de datos reales y sintéticos se pedirá hacer manipulaciones de datos y adaptar el diseño al SGBD específico para que las consultas, además de devolver los resultados esperados, optimicen los recursos utilizados. En particular, se tiene que saber analizar las relaciones entre las diferentes tablas, los volúmenes de datos y reformular las consultas o proponer, en caso de ser necesario, la creación de índices u otros tipos de mejoras.

La práctica se divide en 6 bloques. En cada uno de ellos se trabaja principalmente uno de los conocimientos que se consideran necesarios para superar la práctica:

BLOQUE 1: carga de datos.

BLOQUE 2: lenguaje SQL.

BLOQUE 3: revisión implementación modelo.

BLOQUE 4: gestión de usuarios y permisos.

BLOQUE 5: índices.

BLOQUE 6: optimización.

BLOQUE 1: carga de datos

En este bloque presentamos una guía para realizar una carga masiva de datos sobre la que trabajaremos en el resto de bloques. Seguid el procedimiento paso a paso antes de continuar con el resto de la práctica.

Es muy importante ejecutar correctamente este bloque para poder realizar satisfactoriamente el resto de bloques de la práctica.

Ejecutad la siguiente instrucción en SQL Developer.

```
SELECT * FROM v$version;
```

EJERCICIO 1A: mostrad una captura de pantalla con el resultado de la ejecución:

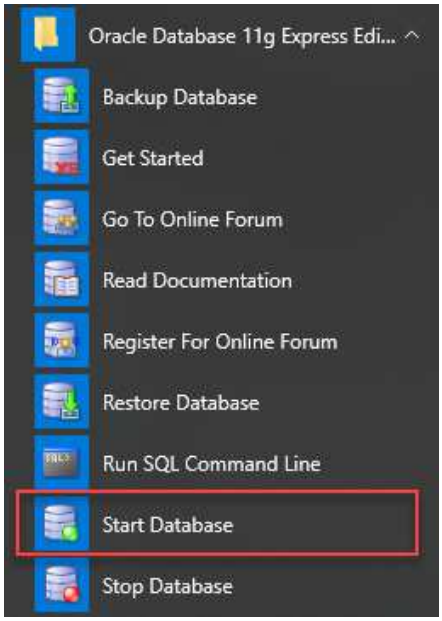
	BANNER
1	Oracle Database 11g Express Edition Release 11.2.0.2.0 - Production
2	PL/SQL Release 11.2.0.2.0 - Production
3	CORE11.2.0.2.0Production
4	TNS for 32-bit Windows: Version 11.2.0.2.0 - Production
5	NLSRTL Version 11.2.0.2.0 - Production

En el supuesto de que el resultado de la línea donde consta la versión de 'Database' no sea **"Oracle Database 11g Express Edition Release 11.2.0.2.0 – Production"** y donde consta la versión de "PL/SQL" no aparezca **"PL/SQL Release 11.2.0.2.0 – Production",** tendréis que revisar el fichero origen utilizado para hacer la instalación, desinstalar la versión actual y volver a hacer la instalación con la versión correcta.

No continuéis con la práctica hasta que no tengáis la versión XE (eXpress Edition) 11.2 **versión 32 bits** instalada, puesto que los resultados del optimizador pueden diferir de los esperados y provocar que las consultas no se ejecuten como se espera.

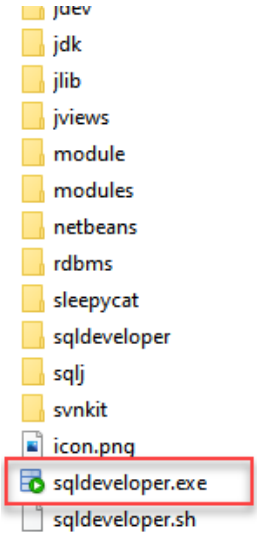
Seguiremos con la creación de un usuario desde la interfaz de Oracle SQL Developer. Tened en cuenta que, en las imágenes de esta práctica, la interfaz está en idioma inglés y, en vuestro caso, podría estar en algún otro idioma en función de cómo hayáis realizado la instalación.

Primero deberemos iniciar el servidor de bases de datos Oracle (en caso de que no esté ya iniciado). Para ello pulsaremos en "Start Database" en el menú de inicio de Windows, carpeta Oracle Database 11g Express Edition:



Esto abrirá una ventana de la línea de comandos indicando que se está iniciando el servicio. Si Windows os pide algún permiso para ejecutar, pulsad que sí. Cuando acabe (serán unos segundos), ya tendremos el servidor iniciado y podremos cerrar la ventana de línea de comandos.

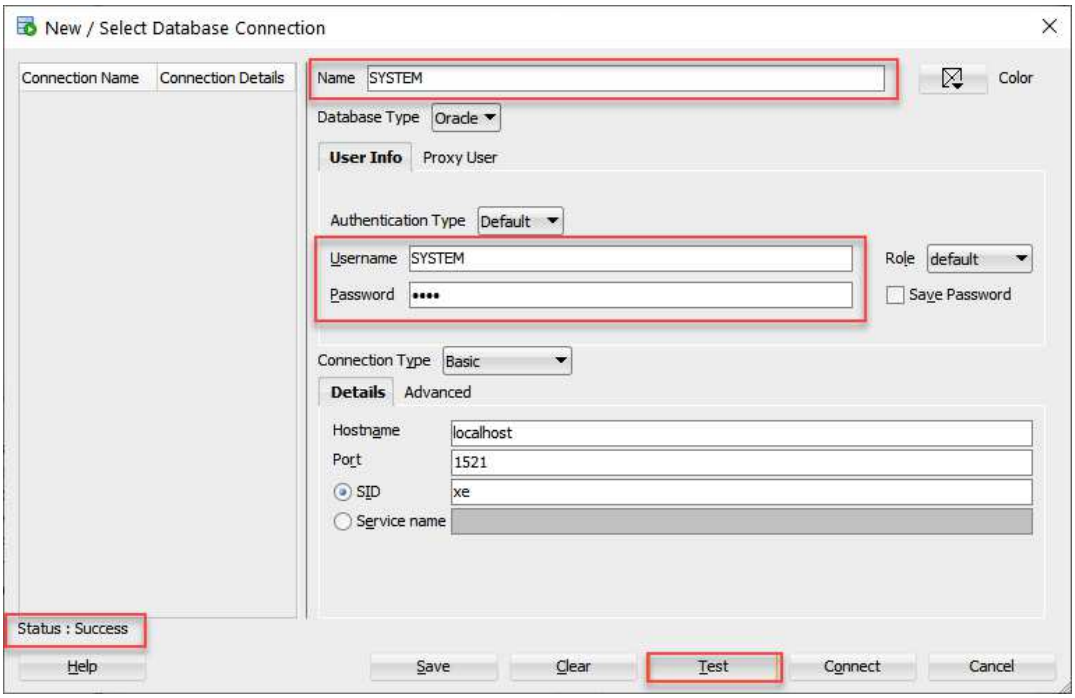
Ahora procederemos a crear el nuevo usuario con el que trabajaremos durante la práctica. Para ello necesitaremos abrir Oracle SQL Developer. Tendréis el ejecutable en la carpeta donde lo hayáis instalado:



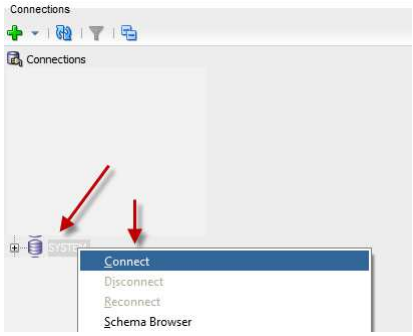
Crearemos primero una conexión con el usuario SYSTEM (si aún no la tenéis), que tiene rol de administrador de base de datos y, por tanto, nos permitirá crear nuestro usuario para la práctica. Pulsad el botón derecho encima de “Connections” y seleccionad “New Connection…”:

PROPUESTA DE SOLUCIÓN

En la pantalla que os aparecerá cread una conexión llamada SYSTEM, con Username SYSTEM y el *Password* que asignasteis durante la instalación de Oracle. Seleccionad la casilla para guardar el *password* y pulsad el botón *Test* para aseguraros de que la conexión es correcta:



Si la conexión es correcta, os aparecerá “Status: Success” como se muestra en la imagen anterior. En caso contrario, os aparecerá el mensaje de error correspondiente. Una vez creada os aparecerá en la lista de conexiones de la izquierda, donde ya podremos pulsar encima de la conexión con el botón derecho y seleccionar “Connect” para conectarnos a la base de datos:



Se nos abrirá una pestaña donde podremos empezar a ejecutar nuestros comandos SQL. En este caso, ejecutad la siguiente instrucción para crear el usuario que utilizaremos, T21:

```
CREATE USER T21
IDENTIFIED BY abc1234
DEFAULT TABLESPACE users
QUOTA UNLIMITED ON users
TEMPORARY TABLESPACE temp;
```

Podéis ejecutar la instrucción de varias maneras, aunque la más sencilla es pulsando la tecla F9, la cual ejecuta el script que tengáis en el editor en ese momento. Lo que acabamos de hacer es crear un usuario llamado T21, que tiene como *password* "abc1234", su *tablespace* por defecto es "users", tiene cuota ilimitada y su *tablespace* temporal es "temp".

Si la instrucción anterior se ejecuta correctamente, nos aparecerá el mensaje "user T21 created" en la ventana de salida de la ejecución. En caso contrario, solucionad el error.

Una vez obtenido el mensaje "user T21 created", tendremos que asignar los permisos necesarios para seguir con la práctica: crear sesión, crear usuarios, crear tablas, crear vistas, crear vistas materializadas, crear disparadores y crear secuencias. Para ello, ejecutad la siguiente instrucción SQL:

```
GRANT
 CREATE SESSION,
 CREATE USER,
 CREATE TABLE,
 CREATE VIEW,
 CREATE MATERIALIZED VIEW,
 CREATE TRIGGER,
 CREATE SEQUENCE
TO T21;
```

De nuevo, si la instrucción se ejecuta correctamente, os aparecerá el mensaje "GRANT succeeded".

Ahora ya debemos crear la conexión para el nuevo usuario T21. Seguid los mismos pasos que para crear la conexión SYSTEM, pero esta vez cambiando el nombre de la conexión y del usuario por T21, y utilizando la contraseña abc1234 que le hemos asignado.

PROPUESTA DE SOLUCIÓN

En este momento ya tenemos la conexión que utilizaremos durante la práctica. Cerrad la conexión del usuario SYSTEM, ya que es peligroso tener una sesión abierta con un usuario con tantos privilegios. Para ello, pulsamos encima de la conexión con el botón derecho y seleccionamos “Disconnect”. En su lugar, abrimos la nueva conexión T21 (botón derecho, “Connect”).

Por último, procederemos con una carga de datos a partir del fichero de *backup* suministrado con el enunciado (T21_backup.dmp). Para hacer la carga del fichero .dmp, abrid la ventana del CMD o ‘Símbolo del Sistema’ y moveos con el comando CD al directorio donde tengáis guardado el fichero .dmp y escribid la siguiente instrucción desde la línea de comandos:

```
imp T21/abc1234 LOG=T21_log.log FILE=T21_backup.dmp
```

EJERCICIO 1B: anotad el número de registros que contiene cada una de las tablas indicadas en el cuadro siguiente:

	Nombre tabla	Registros
1	Person	186316
2	Company	151309
3	Stand	334

EJERCICIO 1C: queremos cargar los datos de las salas de reuniones que existen en los diferentes pabellones del recinto ferial. Nos vienen dados en un fichero tipo TSV. Con este fin, crearemos una nueva tabla llamada "MeetingArea", donde cargaremos los datos del fichero de texto "MeetingArea.TSV".

La definición del fichero "MeetingArea.TSV" es la siguiente:

Archivo: MeetingArea.TSV

Formato: Variable, separado por tabuladores

Codificación: UTF8

Contenido: Características de las distintas áreas de reunión de los pabellones

Campo	Significado	Tipo original	Anchura	¿Puede ser nulo?
code	Código	Carácter variable	20	
codePavilion	Código del pabellón donde se encuentra el área	Carácter	10	NO
zone	Zona	Carácter	20	NO
facilitie	Tipo característica	Carácter variable	20	NO
quantity	Cantidad del servicio (máx. personas, m2, altura, …)	Numérico con decimales	2 decimales	NO

Comentarios:

La clave primaria es code.

Para un mismo codePavilion y zone, no se puede repetir el valor de 'facilitie'.

El campo codePavilion hace referencia a code de la tabla Pavilion.

Mostrad el código SQL utilizado para crear la tabla "MeetingArea". Para crear las restricciones que sean necesarias, tened en cuenta todos los detalles indicados en la

definición del fichero. Dad nombre a todas las restricciones.

```
CREATE TABLE MeetingArea (
    code VARCHAR2(20 CHAR) CONSTRAINT PK_MeetingArea PRIMARY KEY,
    codePavilion CHAR(10 CHAR) CONSTRAINT NN_Meetingarea_codePavilion NOT NULL,
    zone CHAR(20 CHAR) CONSTRAINT NN_Meetingarea_zone NOT NULL,
    facilitie VARCHAR2(20 CHAR) CONSTRAINT NN_Meetingarea_facilitie NOT NULL,
    quantity NUMBER(*,2) CONSTRAINT NN_Meetingarea_quantity NOT NULL,
    CONSTRAINT AK_MeetingArea UNIQUE (codePavilion, zone, facilitie),
    CONSTRAINT FK_MeetingAreaPavilion FOREIGN KEY (codePavilion) REFERENCES Pavilion (code)
);
```

Ahora ya podemos cargar la tabla MeetingArea y para ello tomaremos los datos facilitados en el fichero “MeetingArea.TSV”.

EJERCICIO 1D: Mostrad el contenido del fichero .ctl utilizado para cargar los datos del fichero “MeetingArea.TSV”. en la tabla MeetingArea con SQLLoader. Evitad cargar el contenido de la primera fila (cabecera con el nombre de las columnas). ¡No editéis el fichero “MeetingArea.TSV” para modificar el contenido!

NOTA: según configuración de la base de datos (CHARACTER SET), puede ser necesario indicar que el indicador de decimal es una coma.

```
OPTIONS (SKIP=1)
LOAD DATA
CHARACTERSET 'UTF8'
INFILE 'MeetingArea.tsv'
INTO TABLE MeetingArea
FIELDS TERMINATED BY X'9'
TRAILING NULLCOLS
(
  code,
  codePavilion,
  zone,
  facilitie,
  quantity
)
```

EJERCICIO 1E: anotad aquí la instrucción sqlldr que habéis utilizado para cargar los datos del fichero .ctl:

```
sqlldr userid=T21/abc1234 control= MeetingArea.ctl
```

EJERCICIO 1F: indicad el número de registros cargados en la tabla MeetingArea:

	Nombre tabla	Registros
1	MeetingArea	105

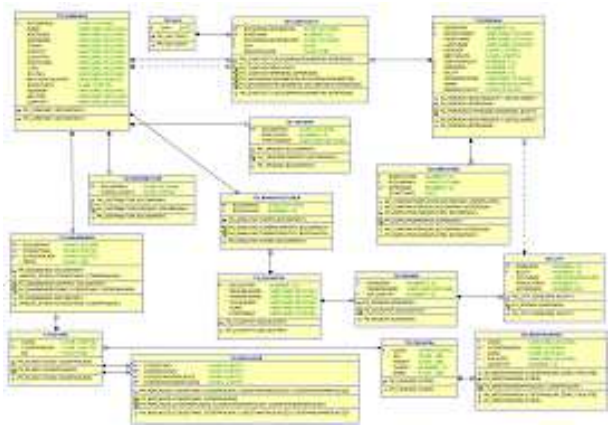
PROPUESTA DE SOLUCIÓN

EJERCICIO 1G: deseamos disponer de una versión gráfica ER del modelo que hemos importado. Para ello, utilizaremos ingeniería inversa mediante las opciones que incorpora SQLDeveloper. Ejecutad los siguientes pasos para obtener el diagrama:

1) Desde el SQL Developer, menú “File” -> “Data Modeler” -> “Import” -> “Data Dictionary” (o en el idioma en el que tengáis la instalación).
2) En la ventana del asistente, elegid el usuario T21. Pulsad el botón “Next”. Cuando se os pida elegir un esquema, marcad el T21. Pulsad sobre el botón “Next”.
3) Cuando aparezca una ventana donde elegir los objetos a importar, marcad todas las tablas. Pulsad sobre el botón “Next” y después sobre el botón “Finish”.

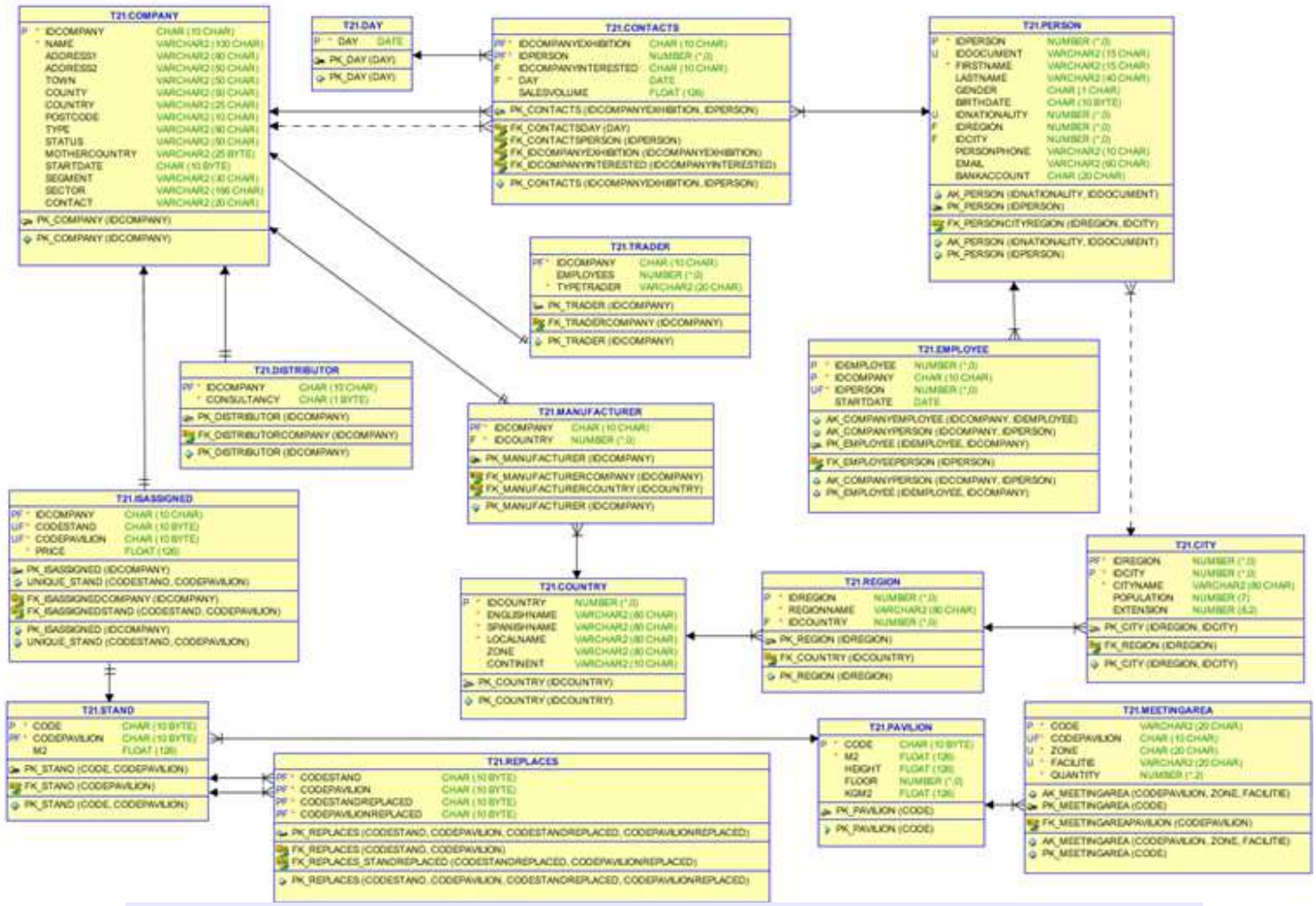
Generado el diagrama ER, ordenad mínimamente las tablas para evitar los excesivos cruces de las líneas de las relaciones.

NOTA: El diagrama ER que obtengáis debe ser similar al siguiente, ¡pero donde se puedan ver todos los detalles!



Pegad aquí una captura de pantalla con el diagrama obtenido:

PROPUESTA DE SOLUCIÓN



BLOQUE 2: LENGUAJE SQL

En este bloque trabajaremos los conocimientos en lenguaje SQL. Para ello, se solicita que realicéis una serie de consultas siguiendo las indicaciones proporcionadas. Leed con atención los ejercicios para asegurar que vuestras consultas cumplen con todos los requerimientos.

EJERCICIO 2A: los responsables del recinto ferial quieren conocer el total de contactos realizados durante la feria en los diferentes pabellones, entre las empresas que participan en la feria, a efectos determinar si exponer en un pabellón u otro puede tener incidencia en la cantidad de contactos.

Nos interesa visualizar los valores de estas columnas, así como los metros cuadrados de los pabellones, con dichos nombres, y en el orden indicado:

	Columna	Descripción
1	code	code de Pavilion
2	m2	m2 de Pavilion
3	totalContacts	Número de contactos que se han realizado en cada pabellón (calculado).

Además, los resultados se deben ordenar de manera que se muestre primero los pabellones donde han existido más contactos.

```
SELECT
  Pavilion.code,
  Pavilion.m2, count(*) totalContacts
FROM Contacts, Company, IsAssigned, Pavilion
WHERE Contacts.idCompanyExhibition = Company.idCompany
  AND Company.idCompany = IsAssigned.idCompany
  AND IsAssigned.codePavilion = Pavilion.code
  AND Contacts.idCompanyInterested IS NOT NULL
GROUP BY Pavilion.code, Pavilion.m2
ORDER BY COUNT(*) DESC;
```

Captura de pantalla con los resultados:

	CODE	M2	TOTALCONTACTS
1	GV4	20000	908
2	GV5	17000	778
3	GV3	43000	708
4	GV2	32000	419
5	MP2	24000	387
6	GV8	13800	365
7	MP4	10300	215
8	MP7	6330	170

EJERCICIO 2B: para que la ejecución de la consulta anterior resulte más cómoda, cread una vista llamada V_ContactsByPavilion que liste esos resultados. Dad nombre a todas las columnas.

Mostrad a continuación el código SQL necesario para crear la vista:

```
CREATE VIEW V_ContactsByPavilion AS
  SELECT
    Pavilion.code,
    Pavilion.m2, count(*) totalContacts
  FROM Contacts, Company, IsAssigned, Pavilion
  WHERE Contacts.idCompanyExhibition = Company.idCompany
    AND Company.idCompany = IsAssigned.idCompany
    AND IsAssigned.codePavilion = Pavilion.code
    AND Contacts.idCompanyInterested IS NOT NULL
  GROUP BY Pavilion.code, Pavilion.m2
  ORDER BY COUNT(*) DESC;
```

EJERCICIO 2C: ahora se desea mostrar sólo los resultados para los pabellones de la zona 'GV' sea cual sea su número. Utilizando la vista creada, realizad una consulta SQL que muestre la cantidad de contactos de dichos pabellones. Mostrad también una captura de pantalla con los resultados.

```
SELECT * FROM V_ContactsByPavilion WHERE CODE LIKE 'GV%';
```

	CODE	M2	TOTALCONTACTS
1	GV4	20000	908
2	GV5	17000	778
3	GV3	43000	708
4	GV2	32000	419
5	GV8	13800	365

EJERCICIO 2D: debido a que el nombre de los países se almacena en diversas columnas según idioma (inglés, castellano y en el idioma original), las consultas se pueden complicar cuando se realizan por nombre de país. Es por ello que se desea crear una vista con nombre V_AllCountries, que permita consultar dicha información utilizando cualquiera de los tres nombres de país existentes. Las columnas de que debe constar la vista son:

	Columna	Descripción
1	idCountry	idCountry de Country
2	name	Nombre del país consultado.

En el caso de realizarse consultas que devuelvan múltiples filas, estas deben quedar

ordenadas por los nombres de los países, ascendentemente.

Además y para facilitar las consultas, los nombres de los países a consultar se escribirán siempre en mayúsculas, por lo que la vista lo deberá tener en cuenta.

```
CREATE VIEW V_AllCountries AS
  SELECT UNIQUE(idCountry), name FROM (
    SELECT idCountry, UPPER(englishName) name FROM COUNTRY
    UNION ALL
    SELECT idCountry, UPPER(spanishName) name FROM COUNTRY
    UNION ALL
    SELECT idCountry, UPPER(localName) name FROM COUNTRY)
ORDER BY name;
```

EJERCICIO 2E: utilizando V_AllCountries, realiza tres consultas para recuperar respectivamente los datos del país 'LIETUVA', del país 'LITHUANIA', y del país 'LITUANIA'.

Mostrad los tres SQLs utilizados y una captura de pantalla con el resultado de ejecutar la consulta para 'LITUANIA':

```
SELECT * FROM V_AllCountries WHERE name = 'LIETUVA';

SELECT * FROM V_AllCountries WHERE name = 'LITHUANIA';

SELECT * FROM V_AllCountries WHERE name = 'LITUANIA';
```

	IDCOUNTRY	NAME
1	123	LITUANIA

EJERCICIO 2F: entre los cambios introducidos en el diseño del sistema informático para la feria de movilidad, se encuentra la posibilidad de almacenar los contactos establecidos tanto con empresas como con personas sin vinculación con las empresas participantes en la feria (pero si registradas). Debido a que pueden existir diferencias significativas de contactos entre ambos colectivos, se desea realizar una consulta que recupere dicho número de contactos para cada uno de ellos.

Realizad una consulta que muestre exactamente las columnas que se indican a continuación, con el mismo nombre que se indica, y en el mismo orden.

	Columna	Descripción
1	dateContact	Día en que se ha establecido el contacto en formato DD/MM/YYYY
2	weekDay	Día de la semana en texto (lunes, martes, …)
3	totalContactsProf	Total de contactos entre empresas expositoras para el día.
4	totalContactsProfPercent	Porcentaje de contactos, del total existente entre

		empresas expositoras de todos los días, con dos decimales.
5	totalContactsPriv	Total de contactos entre empresas y personas que no son de empresas participantes en la feria, para el día
6	totalContactsPrivPercent	Porcentaje de contactos entre empresas y personas que no son de empresas participantes en la feria, con dos decimales, sobre el total de contactos de todos los días

NOTA: Se puede obtener el nombre del día de la semana con un TO_CHAR con DAY.

Ejemplo: SELECT TO_CHAR(SYSDATE, 'DAY', 'NLS_DATE_LANGUAGE=SPANISH') FROM Dual;

Además, los resultados deben estar ordenados ascendentemente por fecha. Mostrad la consulta SQL que cumpla con dichos requerimientos:

```
SELECT
  DayCongress.day  dateContact,  TO_CHAR(DayCongress.day,  'DAY',  'NLS_DATE_LANGUAGE=SPANISH')
weekDay,
  -- Empresas
  (
    SELECT count(*) totaContacts FROM Contacts, Company, IsAssigned
    WHERE Contacts.idCompanyExhibition = Company.idCompany
      AND Company.idCompany = IsAssigned.idCompany
      AND Contacts.day = DayCongress.day
      AND Contacts.idCompanyInterested IS NOT NULL) totalContactsProf,
  TRUNC(
    (
    SELECT count(*) totaContacts FROM Contacts, Company, IsAssigned
    WHERE Contacts.idCompanyExhibition = Company.idCompany
      AND Company.idCompany = IsAssigned.idCompany
      AND Contacts.day = DayCongress.day
      AND Contacts.idCompanyInterested IS NOT NULL)*100 /
    (
    SELECT count(*) totaContacts FROM Contacts, Company, IsAssigned
    WHERE Contacts.idCompanyExhibition = Company.idCompany
      AND Company.idCompany = IsAssigned.idCompany
      AND Contacts.idCompanyInterested IS NOT NULL),2)||'%' totalContactsProfPercent,
   -- Privados
        (
    SELECT count(*) totaContacts FROM Contacts, Company, IsAssigned
    WHERE Contacts.idCompanyExhibition = Company.idCompany
      AND Company.idCompany = IsAssigned.idCompany
      AND Contacts.day = DayCongress.day
      AND Contacts.idCompanyInterested IS NULL) totalContactsPriv,
  TRUNC(
    (
    SELECT count(*) totaContacts FROM Contacts, Company, IsAssigned
    WHERE Contacts.idCompanyExhibition = Company.idCompany
      AND Company.idCompany = IsAssigned.idCompany
      AND Contacts.day = DayCongress.day
      AND Contacts.idCompanyInterested IS NULL)*100 /
    (
    SELECT count(*) totaContacts FROM Contacts, Company, IsAssigned
    WHERE Contacts.idCompanyExhibition = Company.idCompany
      AND Company.idCompany = IsAssigned.idCompany
      AND Contacts.idCompanyInterested IS NULL),2)||'%' totalContactsPrivPercent
FROM Day DayCongress
ORDER BY DayCongress.day;
```

PROPUESTA DE SOLUCIÓN

Mostrad una captura de pantalla con los resultados de ejecutar dicha consulta.

	DATECONTACT	WEEKDAY	TOTALCONTACTSPROF	TOTALCONTACTSPROFPERCENT	TOTALCONTACTSPRIV	TOTALCONTACTSPRIVPERCENT
1	08/09/21	MIÉRCOLES	757	19,16%	3387	15,36%
2	09/09/21	JUEVES	774	19,59%	3383	15,34%
3	10/09/21	VIERNES	822	20,81%	5102	23,14%
4	11/09/21	SÁBADO	1178	29,82%	5941	26,95%
5	12/09/21	DOMINGO	419	10,6%	4227	19,17%

BLOQUE 3: REVISIÓN IMPLEMENTACIÓN MODELO

EJERCICIO 3A: nos indican que existen 9298 registros en la tabla Employee, pero que solo 9275 estarían relacionados con posibles empresas.

¿A que es debido que puedan existir empleados sin empresa asignada?

Ello es debido a que no hay una relación entre ambas tablas. En concreto faltaría la relación, clave foránea, entre el campo idCompany de Employee y de idCompany de Company. Adicionalmente, faltaría que el campo idCompany de Employee tuviera una restricción de NOT NULL.

EJERCICIO 3B: mostrad la consulta SQL que recupera las columnas idEmployee, idPerson, idCompany y startDate, de los empleados que están asignados a empresas inexistentes, y mostrad también la captura de pantalla con el resultado de dicha consulta.

Puesto que se espera que la consulta sea óptima, evitad utilizar subconsultas y la cláusula 'NOT IN' en el SQL propuesto.

```
SELECT Employee.idEmployee, Employee.idPerson, Employee.idCompany, Employee.startDate
FROM Employee LEFT JOIN Company ON (Employee.idCompany = Company.idCompany)
WHERE Company.idCompany IS NULL;
```

	IDEMPLOYEE	IDCOMPANY	IDPERSON	STARTDATE
1	4	4971299	411959	12/03/04
2	5	SC287292	415875	19/03/05
3	9	SC294497	407602	29/10/81
4	12	SC330807	415199	26/02/10
5	12	SC321046	413483	01/01/85
6	13	SC311646	410998	29/03/06
7	14	SC288886	407577	14/04/82
8	15	SC329227	407031	08/03/11
9	16	SC326493	408128	25/04/84
10	18	SC288351	408588	02/08/99
11	19	SC317089	414851	17/02/11
12	19	SC281414	410080	06/10/19
13	20	SC301983	415108	30/01/94
14	20	SC290219	408657	04/06/85
15	21	SC312388	407767	27/07/78
16	21	SC297189	416791	08/04/84
17	22	SC289462	408618	10/02/08
18	22	SC287102	409928	04/07/95
19	24	SC329982	406657	18/03/92
20	25	SC322846	413868	06/07/94
21	28	SC277455	406694	24/01/83
22	67	SC286163	412177	29/07/17
23	69	SC285007	409268	25/07/99

PROPUESTA DE SOLUCIÓN

EJERCICIO 3C: Mostrad la instrucción/instrucciones para modificar la tabla, que aseguraría que no se pudieran insertar empleados que no tuvieran empresa asignada, y que no se podrían asignar estos a empresas inexistentes.

```
ALTER TABLE Employee ADD CONSTRAINT FK_EmplyeeCompany
  FOREIGN KEY (idCompany) REFERENCES Company(idCompany);

ALTER TABLE Employee ADD CONSTRAINT NN_idCompany CHECK (idCompany IS NOT NULL);
```

¿La instrucción o instrucciones mostradas se pueden ejecutar? En caso afirmativo indicad si es lo esperado y en caso negativo anotad los posibles motivos.

```
La creación de la clave foránea no se puede ejecutar, puesto existen empleados que tienen
asignadas empresas inexistentes; 'error ORA-02298: parent keys not found'.

La instrucción para la creación de la restricción NN_idCompany sí se podría ejecutar, puesto
al crearse la restricción solo comprueba que el campo no contenga valores tipo nulo.

Antes de ejecutar las instrucciones se deberían eliminar los registros que no cumplen las
condiciones o corregir dichos datos.
```

EJERCICIO 3D: revisando el modelo físico de datos, vemos que en la tabla Replaces pueden existir datos erróneos. En concreto se detecta que un stand aparece como sustituto de él mismo.

Realizad la consulta para verificar dicha aseveración. Muestra el SQL y el resultado de la consulta:

```
SELECT * FROM Replaces
WHERE
codeStand = codeStandReplaced
AND codePavilion = codePavilionReplaced;
```

	CODESTAND	CODEPAVILION	CODESTANDREPLACED	CODEPAVILIONREPLACED
1	B1	GV8	B1	GV8

EJERCICIO 3E: mostrad seguidamente el SQL que se debería ejecutar para evitar que en un futuro se pudieran insertar stands como sustitutos de ellos mismos.

```
ALTER TABLE Replaces ADD CONSTRAINT CH_codeStand
  CHECK ((codeStand)||(codePavilion) <> (codeStandReplaced)||(codePavilionReplaced));
```

EJERCICIO 3F: ¿Se ha podido ejecutar la instrucción propuesta? ¿Es correcto haber podido ejecutar dicha instrucción según los datos existentes en la tabla? En caso negativo argumentad los motivos y anotad y mostrad las acciones/instrucciones que se deberían tomar para poder crear dicha restricción.

La creación de la restricción no se ha podido ejecutar puesto existen datos que incumplirían dicha restricción, y es correcto que no se haya podido ejecutar.

Para poder ejecutar la instrucción de creación de la restricción se debería borrar en primer lugar el registro que incumple la condición:

```
DELETE FROM Replaces
WHERE
codeStand = 'B1'
AND codePavilion = 'GV8'
AND codeStandReplaced = 'B1'
AND codePavilionReplaced = 'GV8';
```

o de forma más genérica:

```
DELETE FROM Replaces
WHERE codeStand = codeStandReplaced
AND codePavilion = codePavilionReplaced;
```

BLOQUE 4: GESTIÓN DE USUARIOS Y PERMISOS

Este bloque se centra en la administración de usuarios y sus permisos, punto primordial para asegurar que una base de datos es robusta en lo que respecta a la seguridad.

EJERCICIO 4A: dentro del paquete de facilidades que da la feria a los expositores, consta el acceso a la aplicación de registro de contactos que se han realizado durante la misma. Es por ello que se establecerá una método de acceso a datos por el cual, cada empresa solo podrá acceder a los datos propios de la tabla Contacts.

Cread la siguiente vista:

```
CREATE VIEW V_Contacts AS
  SELECT
    Contacts.day,
    Person.idPerson,
    Person.firstName,
    Person.lastName,
    CompanyIn.idCompany CompanyInt,
    CompanyIn.name CompanyName,
    CompanyIn.sector
  FROM Contacts, Company CompanyEx, Company CompanyIn, Person
 WHERE
    Contacts.idCompanyExhibition = CompanyEx.idCompany
    AND Contacts.idCompanyInterested = CompanyIn.idCompany
    AND Contacts.idPerson = Person.idPerson
    AND TRIM(Contacts.idCompanyExhibition) = TRIM(UPPER(SUBSTR(USER,6,10)))
  UNION
  SELECT
    Contacts.day,
    Person.idPerson,
    Person.firstName,
    Person.lastName,
    '' CompanyInt,
    '' CompanyName,
    ''
  FROM Contacts, Company CompanyEx, Person
  WHERE
    Contacts.idCompanyExhibition = CompanyEx.idCompany
    AND Contacts.idCompanyInterested IS NULL
    AND Contacts.idPerson = Person.idPerson
    AND TRIM(Contacts.idCompanyExhibition) = TRIM(UPPER(SUBSTR(USER,6,10)))
ORDER BY CompanyInt, idPerson;
```

EJERCICIO 4B: cread un usuario llamado “user_6847537”, con password “abc4321”, que tenga permisos para iniciar sesión, consultar la vista V_Contacts y para realizar inserciones sobre Contacts.

Mostrad y ejecutad el código SQL para crear dicho usuario, y otorgar al mismo, única y exclusivamente, los permisos mencionados. Cread dicho usuario desde el usuario ’System’.

```
CREATE USER user_6847537 IDENTIFIED BY abc4321;
```

PROPUESTA DE SOLUCIÓN

```
GRANT CREATE SESSION TO user_6847537;
GRANT SELECT ON T21.v_Contacts TO user_6847537;
GRANT INSERT ON T21.Contacts TO user_6847537;
```

Mostrad una captura de pantalla con las primeras quince filas de resultados de la ejecución de la vista V_Contacts sin restricciones. Ejecutadla desde una sesión iniciada con el usuario 'user_6847537'.

NOTA: recordad que siempre es necesario anteponer 'T21.' al nombre de la vista o tabla, a efectos las operaciones realizadas desde el usuario 'user_6847537' localice los elementos del usuario T21.

```
SELECT * FROM T21.V_Contacts WHERE ROWNUM <= 15;
```

	DAY	IDPERSON	FIRSTNAME	LASTNAME	COMPANYINT	COMPANYNAME	SECTOR
1	11/09/21	409640	Mina	Aiden Grier	10934314	THE FUNDING ORGANISATION LTD.	82912 - Activities of credit bureaus
2	12/09/21	411580	Arin	Aglaia Yanick	10934314	THE FUNDING ORGANISATION LTD.	82912 - Activities of credit bureaus
3	11/09/21	411247	Grover	Willoughby Liana	11010271	TURTLE BAY PADDLEBOARDS LTD	47640 - Retail sale of sports goods, fishing gear, camping goods, boats and bicycles
4	08/09/21	408943	Imogene	Nowell Tristin	11557191	WHITSTABLE METALS AUTO-CORE LTD	03410 - Manufacture of motor vehicles
5	11/09/21	409949	Avaline	Frieda Merv	11557191	WHITSTABLE METALS AUTO-CORE LTD	03410 - Manufacture of motor vehicles
6	08/09/21	409504	Myron	Xavier Pamila	11955576	WOODLAND CENTRE LIMITED	49320 - Taxi operation
7	09/09/21	409596	Shanon	Janna Bysshe	12494335	THE SAILING BOAT COMPANY LIMITED	77341 - Renting and leasing of passenger water transport equipment
8	11/09/21	410541	Eryn	Cori Derren	12494335	THE SAILING BOAT COMPANY LIMITED	77341 - Renting and leasing of passenger water transport equipment
9	11/09/21	410524	Letha	Flannery	12794906	WYS CAMPERS LTD	77110 - Renting and leasing of cars and light motor vehicles
10	09/09/21	409666	Mayson	Marjorie Milly	12796443	VID GROUP LTD	96090 - Other service activities n.e.c.
11	12/09/21	409587	Colton	Kori Colton	13111077	VANITY LUXURY HIRE LIMITED	77110 - Renting and leasing of cars and light motor vehicles
12	12/09/21	410536	Sheldon	Cy Hettie	13111077	VANITY LUXURY HIRE LIMITED	77110 - Renting and leasing of cars and light motor vehicles
13	11/09/21	410593	Brandi	Branson Cari	13491013	TRAYEZ ELECTRICAL LTD	95210 - Repair of consumer electronics
14	09/09/21	411602	Eddy	Odelia Roydon	13491013	TRAYEZ ELECTRICAL LTD	95210 - Repair of consumer electronics
15	10/09/21	411662	Karen	Leopold Marinda	13654892	THE STYLE DESK LTD	18201 - Reproduction of sound recording

EJERCICIO 4C: ahora nos piden realizar una inserción en Contacts desde el usuario 'user_6847537'. En concreto, realizad una inserción con los siguientes datos:

```
idCompanyExhibition: 956213
idPerson: 98703
idCompanyinterested: 6847537
day: 12/09/2021
salesVolume: NULL
```

Mostrad la instrucción SQL y el resultado de su ejecución:

```
INSERT INTO T21.Contacts
  VALUES ('956213', 98703, '6847537', TO_DATE('12/09/2021', 'DD/MM/YYYY'), NULL);
```

Ejecutad la siguiente consulta desde T21:

```
SELECT * FROM Contacts
WHERE
  idCompanyExhibition = '956213'
  AND day = TO_DATE('12/09/2021', 'DD/MM/YYYY')
  AND idCompanyInterested = '6847537';
```

Mostrad el resultado:

PROPUESTA DE SOLUCIÓN

	IDCOMPANYEXHIBITION	IDPERSON	IDCOMPANYINTERESTED	DAY	SALESVOLUME
1	956213	98703	6847537	12/09/21	(null)

¿Se ha realizado la inserción correctamente? En caso afirmativo, ¿según permisos otorgados al usuario es correcta dicha inserción?, en caso negativo, ¿a qué es debido? Argumentad la respuesta sea cual sea el caso observado.

La inserción realizada desde user_6847537 sí se ha realizado, y según permisos otorgados es correcto que el SGBD haya permitido realizar dicha inserción. En concreto, en el momento de otorgar permisos se ha indicado con un GRANT, que se permitían realizar INSERTS sobre Contacts.

En el caso de no observarse los datos desde T21, significará que no se ha realizado el COMMIT después de ejecutar la instrucción INSERT desde user_6847537.

BLOQUE 5: ÍNDICES

Este bloque trabaja otro tema primordial en bases de datos, sobre todo cuando tratamos con grandes volúmenes de datos.

IMPORTANTE: Para asegurar que el SGBD tiene actualizadas las estadísticas y que, por tanto, se utilizan los recursos según el contenido de las tablas y sus índices, se deberá ejecutar la siguiente instrucción:

```
BEGIN
Dbms_Stats.Gather_Schema_Stats (
       ownname => 'T21',
       estimate_percent => 100);
END;
```

Es **obligatorio** ejecutar este proceso de actualización de las estadísticas cada vez que se hagan cambios de importancia en la BBDD, tal como por ejemplo, el poblado/borrado masivo de datos, creación/modificación de tablas, creación de índices, etcétera.

Ejecutad desde el usuario T21 dicha actualización de estadísticas.

EJERCICIO 5A: nos suministran la siguiente consulta SQL, la cual nos dicen que se ejecuta con bastante frecuencia, a efectos revisar los sectores de actividad de las empresas:

```
SELECT *  FROM Company WHERE SUBSTR(SECTOR,1,2) = '49';
```

Mostrad el plan de ejecución de la consulta. Podéis obtener el plan de ejecución colocando el cursor en la línea de la consulta SQL y pulsando F10 en SQL Developer, o bien pulsando directamente F10, si es la única consulta existente en la hoja de trabajo. Asegurad que se muestran y que son legibles las columnas OPERATION, OBJECT_NAME, CARDINALITY y COST:

En la columna CARDINALITY se observa el valor estadístico calculado, sobre la variabilidad de los valores que hay en la columna utilizada para la búsqueda.

Así se puede obtener el valor aproximado de CARDINALITY contando el número de tuplas de la tabla Company y dividiendo este valor entre el número de valores diferentes de 'sectores de actividad' que existen, según su categorización por los dos primeros dígitos de la izquierda.

PROPUESTA DE SOLUCIÓN

EJERCICIO 5B: mostrad la consulta SQL que realiza los cálculos comentados anteriormente, mostrando sólo dos cifras decimales.

```
SELECT TRUNC(COUNT(*) / COUNT(DISTINCT SUBSTR(sector,1,2)), 2) AVG_Total FROM Company;
```

Mostrad ahora la captura de pantalla con dicho resultado:

	AVG_TOTAL
1	1681,21

Según la cardinalidad estimada, el optimizador opta por utilizar un plan de ejecución u otro.

EJERCICIO 5C: puesto nos dicen que este tipo de consulta (5A) se ejecuta con bastante asiduidad, optimizaremos su ejecución mediante la creación de un nuevo índice. Cread el índice que consideréis óptimo para mejorar el coste de ejecución de la consulta. Mostrad el código SQL utilizado para crear el índice:

```
CREATE INDEX IDX_sector ON Company (SUBSTR(sector,1,2));
```

EJERCICIO 5D: mostrad el nuevo plan de ejecución de la consulta después de crear el índice y actualizar las estadísticas:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1681	996
TABLE ACCESS	COMPANY	BY INDEX ROWID	1681	996
INDEX	IDX_SECTOR	RANGE SCAN	1681	4
Access Predicates				
SUBSTR(SECTOR,1,2)='49'				

EJERCICIO 5E: ¿mejora el coste de ejecución con el nuevo índice? Justificad la respuesta.

El planificador de ejecución elige utilizar el índice para realizar el acceso a Company por el campo sector, utilizando el índice, debido a que se evita realizar un recorrido completo, y con ello realiza un menor número de operaciones de E/S de disco.

El tiempo de ejecución mejora notablemente, y esto es debido a que al realizar la búsqueda utilizando un índice en árbol con los valores ordenados ascendentemente, y a partir de un cierto valor, se localizan las ocurrencias utilizando unos pocos accesos al índice, hasta que el valor deja de ser el buscado. En el caso inicial se realizaba un recorrido de toda la tabla revisando todos los valores de sector, para comprobar si coincide con el buscado.

PROPUESTA DE SOLUCIÓN

EJERCICIO 5F: nos hacen llegar otra consulta que también dicen ejecutar con mucha frecuencia, cambiando los valores consultados:

```
SELECT * FROM Pavilion WHERE Pavilion.code = 'MPCB';
```

Mostrad el plan de ejecución de la consulta anterior:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1
TABLE ACCESS	PAVILION	BY INDEX ROWID	1	1
INDEX	PK_PAVILION	UNIQUE SCAN	1	0
Access Predicates				
PAVILION.CODE='MPCB'				

EJERCICIO 5G: ¿podemos crear algún índice para mejorar el coste de ejecución de la consulta anterior? ¿Cuál sería dicho índice? En el caso de que no existiera índice, ¿mejoraría un posible nuevo índice el plan de ejecución? Mostrad y ejecutad los SQLs/DDLs que creáis necesarios para justificar la respuesta y argumentadla.

Responded ordenadamente a las cuestiones:

A. No podemos crear ningún índice para mejorar la consulta, ya que existe un índice compatible con la búsqueda realizada, en concreto para la columna code. Como podemos ver en el plan de ejecución, el optimizador está haciendo un UNIQUE SCAN mediante el índice PK_Pavilion. Como sabemos, el SGBD crea automáticamente un índice para las columnas que se marcan como clave primaria, tal como es code. El motivo es que siempre que el SGBD tiene que realizar un INSERT o un UPDATE, debe consultar si existe en la tabla ese mismo valor, por lo que las búsquedas en dicha columna serán muy habituales.

B. Así, no debemos (ni podemos) crear índice para dicha columna, puesto que ya existe.

C. Se crea una tabla similar:

```
CREATE TABLE Pavilion2 (
    code CHAR(10 CHAR) ,
    m2 FLOAT ,
    height FLOAT,
    floor NUMBER(5),
    kgm2 FLOAT
);
```

Se llena con el contenido de Pavilion:

```
INSERT INTO PAVILION2 SELECT * FROM PAVILION;
```

Después de reejecutar el cálculo de estadísticas se vuelve a solicitar el plan de ejecución:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	3
TABLE ACCESS	PAVILION2	FULL	1	3
Filter Predicates				
PAVILION2.CODE='MPCB'				

Así, aún a pesar de la poca cantidad de filas de la tabla, realizar un recorrido es más costoso que realizar el acceso por índice.

EJERCICIO 5H: mostrad el plan de ejecución de la siguiente consulta:

```
SELECT * FROM V_AllCountries WHERE name = 'LIETUVA';
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	11
VIEW	V_ALLCOUNTRIES		3	11
SORT		ORDER BY	3	11
SORT		UNIQUE	3	10
VIEW			6	9
UNION-ALL				
TABLE ACCESS	COUNTRY	FULL	2	3
Filter Predicates				
UPPER(ENGLISHNAME)='LIETUVA'				
TABLE ACCESS	COUNTRY	FULL	2	3
Filter Predicates				
UPPER(SPANISHNAME)='LIETUVA'				
TABLE ACCESS	COUNTRY	FULL	2	3
Filter Predicates				
UPPER(LOCALNAME)='LIETUVA'				

EJERCICIO 5I: debido a que se ejecutarán muy a menudo consultas similares a la mostrada, se desea optimizarlas utilizando un índice.

Debido a que dicha consulta se realiza sobre una vista no es posible optimizar la consulta creando un índice.

Cread una vista materializada, con nombre MV_AllCountries, que sea la equivalente a V_AllCountries. No indiquéis opciones de refresco de los datos de la vista en la instrucción de creación. Mostrad la instrucción utilizada:

```
CREATE MATERIALIZED VIEW  MV_AllCountries AS
  SELECT UNIQUE(idCountry), NAME FROM (
    SELECT idCountry, UPPER(ENGLISHNAME) NAME FROM COUNTRY
    UNION ALL
    SELECT idCountry, UPPER(SPANISHNAME) NAME FROM COUNTRY
    UNION ALL
    SELECT idCountry, UPPER(LOCALNAME) NAME FROM COUNTRY)
ORDER BY NAME;
```

Mostrad el plan de ejecución resultante de ejecutar la consulta:

```
SELECT * FROM MV_AllCountries WHERE name = 'LIETUVA';
```

PROPUESTA DE SOLUCIÓN

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	3
MAT_VIEW ACCESS	MV_ALLCOUNTRIES	FULL	1	3
Filter Predicates				
NAME='LIETUVA'				

Argumentad la idoneidad de que la vista sea materializada, según la volatilidad del contenido de dicha tabla, anotando sí se debería considerar preferible la vista materializada sobre la vista normal, con sus pros y contras.

En el caso de realizarse de forma habitual consultas como la mostrada, sin tener que considerar el idioma en que está escrito el nombre del país, ni considerar las mayúsculas/minúsculas, sí se consideraría adecuado crear la vista materializada.

Adicionalmente se debe considerar que el rendimiento de dicha vista materializada es mejor que el de la vista no materializada, puesto se evitan los UNION, y la consulta con la que se crea la vista materializada (con los UNION) se ejecuta una sola vez, al crearse la misma. También se deberá actualizar la vista materializada en el caso de cambios en los datos de la tabla, caso excepcional, puesto los países no cambian de nombre a menudo.

La contrapartida negativa la encontraríamos en el apartado de espacio ocupado en la BD, puesto tendremos datos redundantes; los de la tabla Countries y prácticamente los mismos repetidos en la vista materializada MV_AllCountries.

EJERCICIO 5J: cread el índice que creáis oportuno para mejorar el rendimiento de la consulta sobre la vista anterior. Mostrad la instrucción de creación y nuevamente el plan de ejecución:

```
CREATE INDEX IDX_MV_AllC_Name ON MV_AllCountries(name);
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	2
MAT_VIEW ACCESS	MV_ALLCOUNTRIES	BY INDEX ROWID	1	2
INDEX	IDX_MV_ALLC_NAME	RANGE SCAN	1	1
Access Predicates				
NAME='LIETUVA'				

¿Qué diferencias se observan en el plan de acceso vs cuando no existía el índice? ¿Ha mejorado el rendimiento? ¿En qué casos recomendaríais crear dicho índice, considerando los contenidos de las tablas implicadas?

Se observa que se pasa de realizar un recorrido completo de la tabla a un acceso por índice, en concreto el recién creado IDX_MV_AllC_name.

El rendimiento sí ha mejorado, el coste ha pasado de 3 a 2, o sea, tenemos una mejora del 33%.

Se recomendaría crear el índice, puesto no se prevé realizar inserciones ni modificaciones sobre la tabla Country. Así, siempre que se realizase una consulta el rendimiento sería mejor, y no existirían penalizaciones por posibles actualizaciones o inserciones.

BLOQUE 6: OPTIMIZACIÓN

Este último bloque se centra en uno de los puntos que deberíamos dominar: ¿cómo optimizar nuestra base de datos para que resulte lo más eficiente posible?

Para empezar, ejecutad la siguiente consulta para comprobar la información que devuelve sobre el tamaño de las tablas de usuario:

NOTA: recordad ejecutar el recálculo de estadísticas cada vez que se hayan creado tablas, índices, se hayan realizado variaciones de columnas, poblado masivo de datos, …

```
SELECT
  A.TABLE_NAME,
  A.NUM_ROWS,
  A.BLOCKS,
  B.BLOCK_SIZE,
  TRUNC(((A.BLOCKS*B.BLOCK_SIZE)/NULLIF(A.NUM_ROWS,0)),2) AVG_ROW_SIZE
FROM
  USER_TABLES A JOIN USER_TABLESPACES B
  ON A.TABLESPACE_NAME = B.TABLESPACE_NAME
ORDER BY TABLE_NAME;

Resultado esperado:
```

	TABLE_NAME	NUM_ROWS	BLOCKS	BLOCK_SIZE	AVG_ROW_SIZE
1	CITY	8157	35	8192	35,15
2	COMPANY	151309	5038	8192	272,76
3	CONTACTS	25991	124	8192	39,08
4	COUNTRY	233	5	8192	175,79
5	DAY	5	5	8192	8192
6	DISTRIBUTOR	71	5	8192	576,9
7	EMPLOYEE	9298	43	8192	37,88
8	ISASSIGNED	333	5	8192	123
9	MANUFACTURER	138	5	8192	296,81
10	MEETINGAREA	105	5	8192	390,09
11	MV_ALLCOUNTRIES	489	5	8192	83,76
12	PAVILION	17	5	8192	2409,41
13	PERSON	186316	2014	8192	88,55
14	REGION	143	5	8192	286,43
15	REPLACES	4263	35	8192	67,25
16	STAND	334	5	8192	122,63
17	TRADER	123	5	8192	333

EJERCICIO 6A: observamos que la tabla Company es la que más bloques ocupa. Analizando los campos que la componen, vemos que se pueden normalizar varias columnas. Elegiremos empezar por sector por la uniformidad de datos y puesto se utiliza habitualmente para consultas.

Cread una tabla llamada CompanySector, con las columnas idSector y description.

PROPUESTA DE SOLUCIÓN

Cread una consulta SQL que muestre cual es el tamaño del texto de más caracteres que hay almacenado en la columna sector de Company. Mostrad dicho SQL y una captura de pantalla con el resultado de dicha consulta:

```
SELECT MAX(LENGTH(sector)) MaximumSize FROM Company;
```

	MAXIMUMSIZE
1	166

Mostrad la instrucción DDL para la creación de la tabla CompanySector. Elegid los tipos de datos más adecuados y el tamaño mínimo según los datos que pueblan la columna sector de Company. El contenido de idSector serán los primeros cinco dígitos de sector. El contenido de description será el contenido de sector, sin los primeros caracteres identificativos.

```
CREATE TABLE CompanySector (
  idSector CHAR(5 CHAR) CONSTRAINT PK_CompanySector PRIMARY KEY,
  description VARCHAR2(159 CHAR) CONSTRAINT NN_CS_description NOT NULL
);

NOTA: se utiliza el tipo CHAR para idSector por desconocerse si los ceros son significativos.
```

Mostrad la instrucción SQL para poblar de datos la tabla Sector.

```
INSERT INTO CompanySector
  SELECT SUBSTR(sector,1,5), SUBSTR(sector,9,166)
  FROM Company
  GROUP BY sector;
```

Mostrad la instrucción SQL para crear un campo idSector en Company.

```
ALTER TABLE Company ADD idSector CHAR(5 CHAR);
```

Mostrad la instrucción SQL para poblar con datos la nueva columna idSector de Company. Dicha columna contendrá los cinco primeros caracteres de la columna actual sector de Company.

```
UPDATE Company SET idSector = SUBSTR(sector,1,5);
```

Mostrad la instrucción DDL para añadir una restricción a idSector para que no pueda contener valores tipo NULL.

```
ALTER TABLE Company ADD CONSTRAINT NN_idSector CHECK (idSector IS NOT NULL);
```

Mostrad la instrucción DDL para añadir la restricción de clave foránea de la columna idSector de Company a idSector de CompanySector. Dadle nombre FK_Company_CompanySector.

```
ALTER TABLE Company
  ADD CONSTRAINT FK_Company_CompanySector
  FOREIGN KEY (idSector)
  REFERENCES CompanySector(idSector);
```

Mostrad la instrucción DDL para eliminar la columna sector de Company.

```
ALTER TABLE Company DROP COLUMN sector;
```

EJERCICIO 6B: observamos que en la tabla Company también se puede normalizar la columna type.

Cread una tabla con nombre CompanyType, que contenga las siguientes columnas:

```
idType         numérico de 4 dígitos, clave primaria
description    tipo carácter de anchura mínima según contenido de type de Company, no admite
valores tipo NULL
```

Anotad la consulta para comprobar qué tamaño debe tener description:

```
SELECT MAX(LENGTH(type)) MaximumSize FROM Company;
```

Anotad la instrucción DDL para la creación de la tabla CompanyType

```
CREATE TABLE CompanyType (
       idType NUMBER(4) CONSTRAINT PK_CompanyType PRIMARY KEY,
       description VARCHAR2(89 CHAR) CONSTRAINT NN_CT_description NOT NULL
);

NOTA: se utiliza el tipo NUMBER para idType por ser clave primaria sintética.
```

En este caso el contenido del campo no incluye un posible valor que sirva de clave primaria en la nueva tabla, por lo que se deberá crear una secuencia y un disparador para crear una clave primaria sintética.

En concreto usaremos las siguientes instrucciones:

```
CREATE SEQUENCE SEQ_idType
START WITH 1
```

PROPUESTA DE SOLUCIÓN

```
INCREMENT BY 1;

CREATE TRIGGER TR_INSERT_idType
BEFORE INSERT ON CompanyType
FOR EACH ROW
BEGIN
SELECT SEQ_idType.NEXTVAL INTO :NEW.idType FROM DUAL;
END;
```

Mostrad a continuación, en orden, las instrucciones para:

A. Poblar la tabla CompanyType con el contenido de type de Company.
B. Crear una columna numérica con nombre idType en Company.
C. Poblar la nueva columna idType de Company con el valor coincidente de la inserción de las filas en CompanyType (columna idType).
D. Crear una restricción para qué idType no permita valores NULL.
E. Crear una clave foránea con nombre FK_Company_CompanyType, que relacione idType de Company con idType de CompanyType.
F. Eliminar la columna type de Company.

```
A.
INSERT INTO CompanyType (description)
  SELECT type
  FROM Company
  GROUP BY type;

B.
ALTER TABLE Company ADD idType NUMBER(4);

C.
UPDATE Company SET idType = (
       SELECT CompanyType.idType
       FROM CompanyType
       WHERE
         Company.type = CompanyType.description
       );

D.
ALTER TABLE Company ADD CONSTRAINT NN_idType CHECK (idType IS NOT NULL);

E.
ALTER TABLE Company
  ADD CONSTRAINT FK_Comany_CompanyType
  FOREIGN KEY (idType)
  REFERENCES CompanyType(idType);

F.
ALTER TABLE Company DROP COLUMN type;
```

EJERCICIO 6C: mostrad el resultado de ejecutar el SQL:

```
SELECT
  Company.idCompany,
  SUBSTR(Company.name,1,30) name,
```

PROPUESTA DE SOLUCIÓN

```
  CompanyType.idType,
  SUBSTR(CompanyType.description,1,30) SectorDescription,
  CompanySector.idSector,
  SUBSTR(CompanySector.description, 1,30) TypeDescription
FROM Company, CompanyType, CompanySector
WHERE
  Company.idType = CompanyType.idType
  AND Company.idSector = CompanySector.idSector
  AND ROWNUM <=10
ORDER BY idCompany;
```

	IDCOMPANY	NAME	IDTYPE	SECTORDESCRIPTION	IDSECTOR	TYPEDESCRIPTION
1	551473	W.HALL (BACUP) LIMITED	3	Private Limited Company	68201	Renting and operating of Housi
2	552214	W. & G. YATES (YIELDFIELDS) LI	3	Private Limited Company	10110	Processing and preserving of m
3	552255	THE RUGBY GROUP BENEVOLENT FUN	4	PRI/LTD BY GUAR/NSC (Private,	88990	Other social work activities w
4	552330	WILLIAM OLIVER (NORTH SHIELDS)	3	Private Limited Company	2811	Manufacture metal structures &
5	552713	SUNNINGFIELDS INVESTMENT CO. L	3	Private Limited Company	68100	Buying and selling of own real
6	553055	SUTHERLAND FLATS LIMITED	3	Private Limited Company	64999	Financial intermediation not e
7	553128	TOWN AND COUNTY INVESTMENT COM	3	Private Limited Company	41100	Development of building projec
8	553535	TOTAL UK LIMITED	3	Private Limited Company	19209	Other treatment of petroleum p
9	553914	W.H.M.JEWELLERS LIMITED	3	Private Limited Company	47770	Retail sale of watches and jew
10	554067	WB INDUSTRIAL LTD.	3	Private Limited Company	70100	Activities of head offices

EJERCICIO 6D: volved a ejecutar la consulta que recupera el tamaño ocupado por las tablas.

Considerando que se ha eliminado de Company las columnas sector y type, ¿ha disminuido el espacio ocupado por dicha tabla? ¿A qué es debido?

En la tabla se almacena menos información pero el espacio no se libera. Dicho espacio queda disponible para cuando se inserten nuevos datos.

Para comprobar el espacio real ocupado por Company, tal como si creáramos y pobláramos por primera vez la tabla (sin manipulaciones posteriores), crearemos una tabla CompanyTest y la poblaremos.

A efectos de simplificar al máximo el proceso, cread dicha tabla utilizando una instrucción DDL siguiendo el siguiente patrón:

```
CREATE TABLE NewTableName AS
  (SELECT * FROM TableFromCopyEstructureAndData);
```

Mostrad la instrucción DDL utilizada para crear y poblar la tabla:

```
CREATE TABLE CompanyTest AS
  (SELECT * FROM Company);
```

Mostrad el resultado de volver a realizar la consulta de espacio utilizado por las tablas:

PROPUESTA DE SOLUCIÓN

	TABLE_NAME	NUM_ROWS	BLOCKS	BLOCK_SIZE	AVG_ROW_SIZE
1	CITY	8157	35	8192	35,15
2	COMPANY	151309	5038	8192	272,76
3	COMPANYSECTOR	864	13	8192	123,25
4	COMPANYTEST	151309	3420	8192	185,16
5	COMPANYTYPE	22	5	8192	1861,81
6	CONTACTS	25991	124	8192	39,08
7	COUNTRY	233	5	8192	175,79
8	DAY	5	5	8192	8192
9	DISTRIBUTOR	71	5	8192	576,9
10	EMPLOYEE	9298	43	8192	37,88
11	ISASSIGNED	333	5	8192	123
12	MANUFACTURER	138	5	8192	296,81
13	MEETINGAREA	105	5	8192	390,09
14	MV_ALLCOUNTRIES	489	5	8192	83,76
15	PAVILION	17	5	8192	2409,41
16	PERSON	186316	2014	8192	88,55
17	REGION	143	5	8192	286,43
18	REPLACES	4263	35	8192	67,25
19	STAND	334	5	8192	122,63
20	TRADER	123	5	8192	333

EJERCICIO 6E: ¿Cuál es el resultado de sumar el número de bloques utilizado por CompanyTest, CompanyType y CompanySector? Valorad el resultado obtenido y argumentad cómo afectará la nueva estructura a las consultas y a la calidad de los datos.

El resultado de sumar el espacio que ocupan las tres tablas es de 3438 bloques, notablemente inferior al de Company (5038). Una diferencia de 1600 bloques, o sea, un ahorro de 12Mb (1600 bloques de 8Kbytes cada uno).

Ello es debido a que si bien en Company existen dos columnas adicionales para almacenar el id de sector y de type, dichas columnas son de tipo numérico (tipos muy compactos).

La información de company y de type que se ha llevado a las nuevas tablas, no se repite en ellas, así de los 151309 registros que existen en Company, cada uno de ellos con su correspondiente descripción textual en la columna sector y en la columna type, quedan en solo 864 y 22 registros en CompanySector y CompanyType.

El ahorro de espacio es muy importante, así como la calidad de datos, puesto en CompanySector y CompanyType solo podrán existir datos validados en las tablas correspondientes con las que se relacionan, evitándose inserciones de datos de sectores y tipos con problemas, por ejemplo, tipográficos.

FIN DE LA PRÁCTICA

Recursos

Para resolver esta práctica es necesario utilizar los contenidos de los módulos 1 a 4, y parte del 5, del material docente.

Criterios de valoración

El peso de los ejercicios en la nota total de la práctica es el siguiente:

- 5% - Bloque 1 - Carga de datos
- 20% - Bloque 2 - Lenguaje SQL
- 20% - Bloque 3 - Revisión implementación modelo
- 15% - Bloque 4 - Gestión de usuarios y permisos
- 20% - Bloque 5 - Índices
- 20% - Bloque 6 - Optimización

Esta práctica tiene un peso del 50% en la nota de prácticas de la asignatura.

Recordamos que es obligatorio realizar la práctica para aprobar la asignatura.

Formato y fecha de entrega

El formato del fichero debe ser Word u OpenOffice, y se debe entregar una versión del mismo fichero en PDF. Utilizad el formato original del documento y los espacios reservados para responder cada ejercicio. Todo el código SQL se debe poder seleccionar para copiar y poder probar (en lugar de aparecer como capturas de pantalla**).**

Indentad (tabulad) adecuadamente las consultas SQL para hacerlas lo más legibles posible. Utilizad grafía Pascal para los nombres de las tablas y grafía Camel para los nombres de los campos de las tablas. Cread las tablas con una única instrucción CREATE, no añadáis restricciones a posteriori (con ALTER TABLE) si no se solicita explícitamente.

No enviéis ningún fichero adicional a los solicitados (ni con los SQL, ni con capturas de pantallas, etcétera). Realizad envíos independientes para la versión doc/odt y el fichero PDF.

El nombre del fichero tendrá el siguiente formato:

DBD_PRA2_Apellido1_Apellido2_Nombre.extensión

Los apellidos se escribirán sin acentos. Por ejemplo, un estudiante que se llame Alfredo García Melgar pondría el siguiente nombre al archivo:

DBD_PRA2_Garcia_Melgar_Alfredo.doc (u .odt) y .pdf

IMPORTANTE: El nombre y apellidos del estudiante también deben aparecer en la portada del documento con la solución.

Es responsabilidad del estudiante asegurar que el documento se ha entregado correctamente y corresponde a la actividad que se debe presentar.

PROPUESTA DE SOLUCIÓN

La fecha máxima para entregar la práctica es el día 7 de diciembre de 2021.

Nota: **Propiedad intelectual**

A menudo es inevitable, al producir una obra, hacer uso de recursos creados por terceras personas. Es por lo tanto comprensible hacerlo en el marco de una práctica de los estudios del Grado de Informática, siempre que esto se documente claramente y no suponga plagio en la práctica.

Por lo tanto, al presentar una práctica que haga uso de recursos ajenos, se tiene que presentar junto con ella un documento en que se detallen todos ellos, especificando el nombre de cada recurso, su autor, el lugar donde se obtuvo y su estatus legal: si la obra está protegida por el copyright o se acoge a alguna otra licencia de uso (CreativeCommons, licencia GNU, GPL ...). El estudiante tendrá que asegurarse que la licencia que sea no impide específicamente su uso en el marco de la práctica. En caso de no encontrar la información correspondiente tendrá que asumir que la obra está protegida por el copyright.

Deberán, además, adjuntar los ficheros originales cuando las obras utilizadas sean digitales, y su código fuente si corresponde.

Otro punto a considerar es que cualquier práctica que haga uso de recursos protegidos por el copyright no podrá en ningún caso publicarse en medios de comunicación, a no ser que los propietarios de los derechos intelectuales den su autorización explícita.