APELLIDOS _____ NOMBRE _____

Instrucciones

- El examen consta de dos partes. Esta parte dura media hora. La segunda parte durará dos horas y media.
- Escribe **en esta hoja** tu nombre y la respuesta al test.
- Puedes utilizar lápiz.

PARTE 1: Test de cuestiones teórico-prácticas

1. *(2 puntos)* **SOBRE LA SIGUIENTE CUADRÍCULA, escribe para cada una de las siguientes preguntas la ÚNICA afirmación correcta (a, b, c ó d)**. Una pregunta mal contestada NO puntúa negativo.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A														
puntos según nº aciertos →			0.0	0.2	0.4	0.6	0.8	1,0	1,2	1,4	1,6	1.8	2.0	2.0

APELLIDOS _____ NOMBRE _____

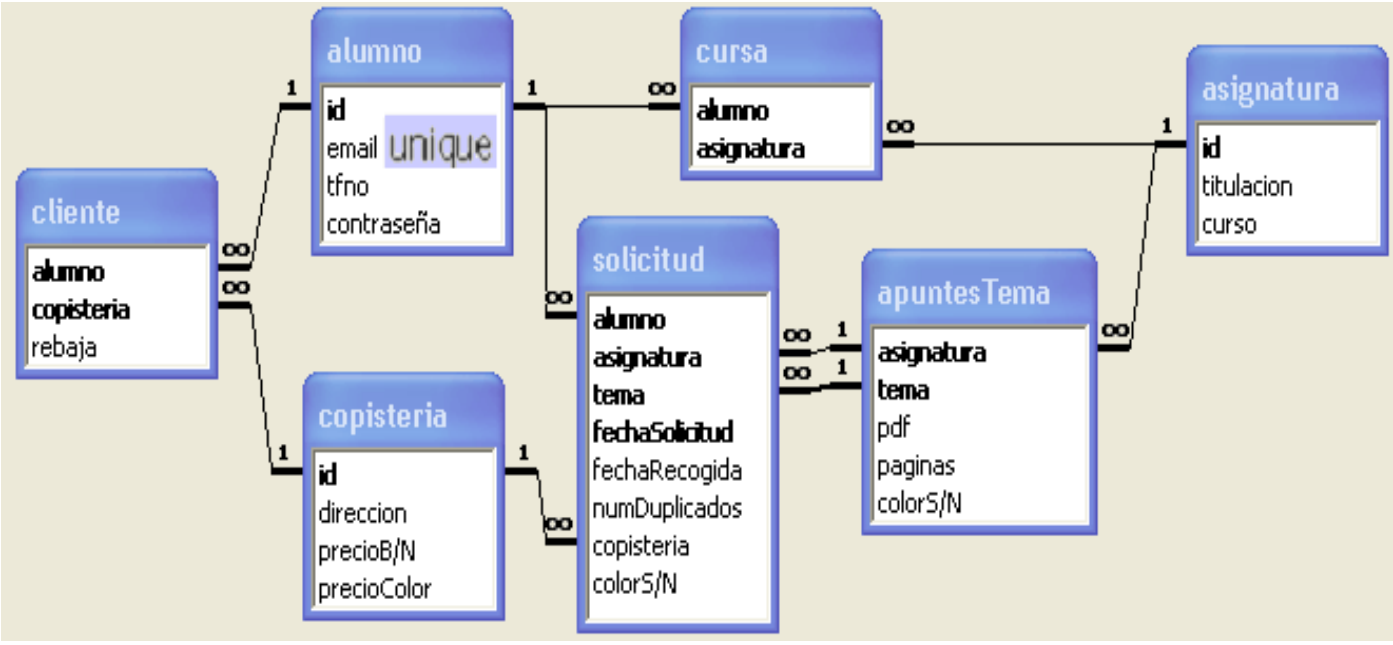
Instrucciones

- Tiempo de esta parte: 2 ½ horas.
- Escribe **en esta hoja tu nombre y la respuesta a la pregunta 2**.
- Escribe tu nombre **en todas las hojas** que uses.
- Puedes utilizar **lápiz**.
- Escribe las respuestas a los ejercicios **siguiendo el mismo orden** que en el enunciado.

PARTE 2: Ejercicios

Una universidad ha firmado un acuerdo con ***un conjunto de copisterías*** para que se encarguen de vender, a un precio ventajoso, copias de los temas desarrollados por los profesores a los alumnos matriculados de las correspondientes asignaturas. Para gestionarlo, se ha desarrollado un sistema informático que permite a los alumnos ver qué temas están disponibles para ser impresos. Los alumnos pueden solicitar mediante este sistema una o varias copias (duplicados) de cada tema de su interés a una copistería concreta. Las copisterías, por su parte, tienen acceso a los ficheros PDF de los temas a imprimir y a las solicitudes dirigidas por los alumnos a la copistería. Cada copistería además puede aplicar porcentajes de rebaja a los alumnos que sean buenos clientes.

Esquema de la BD (los atributos en negrita de cada tabla forman su clave primaria)



asignatura

- Contiene el identificador alfanumérico de la asignatura. Además guarda la titulación y el curso. No se admite que haya nulos en ninguno de los campos.

apuntesTema

- Contiene los apuntes de un tema de una asignatura. La **clave primaria** está **compuesta** por dos campos: asignatura y tema (numérico). **Asignatura es una clave extranjera** a una de las asignaturas de la tabla anterior.

- Ninguno de los campos deberá contener nulos.
- El campo "**pdf**" es de tipo multimedia (tipo de datos "BLOB" en Oracle, "objeto OLE" en Access) y contiene los apuntes del tema en formato pdf.
- El atributo "**paginas**" contiene el número de páginas del fichero pdf y "**colorS/N**" valdrá **true si es interesante imprimir el tema en color**.
- Los profesores de cada asignatura se encargan de incluir los datos a esta tabla.

alumno

- El atributo **id** es numérico y es la clave primaria. El **email** del alumno es clave candidata. Todos los atributos son obligatorios. La contraseña es una cadena alfanumérica cifrada.

cursa

- Contiene los alumnos de cada asignatura. Obsérvese que un mismo identificador de alumno puede aparecer en varias filas de la tabla. También puede repetirse un mismo identificador de asignatura en varias filas.
- Alumno es clave extranjera que referencia a la tabla alumno y asignatura es otra clave extranjera que referencia a la tabla asignatura.

copistería

- Cada fila tiene información de una copistería que se encarga de imprimir apuntes. **Id** es un identificador alfanumérico. PrecioB/N y precioColor contienen cuánto cobra la copistería por cada copia en blanco y negro o en color. Ningún atributo debería contener nulos.

cliente

- Contiene las copisterías de las que son clientes los alumnos. Un alumno puede ser cliente de varias copisterías y una copistería puede tener varios alumnos entre sus clientes.
- El atributo alumno es una clave extranjera que referencia a la tabla alumno y copistería otra clave extranjera que referencia a la tabla copistería.
- El atributo "**rebaja**" contendrá un número positivo o cero que indica el % de rebaja a aplicar al alumno en sus pedidos. No puede valer nulo. Los alumnos que no aparecen en esta tabla no se benefician de ninguna rebaja.

solicitud

- Cada fila recoge una solicitud de impresión de un tema, realizada por un determinado alumno, a una copistería concreta, en una fecha dada. Si colorS/N es true las copias se realizarán en color, si no en blanco y negro.
- El único atributo que admite nulos es "**fechaRecogida**". Si contiene **nulo significa** que aún **no se ha retirado el encargo de la copistería**.
- Un alumno puede encargar el mismo tema varias veces a la misma copistería de dos formas: **(1)** Asignando a **numDuplicados** (entero positivo) un valor mayor que 1. Por ejemplo, si contiene 3 la copistería imprimirá 3 copias del tema completo. **(2)** Haciendo varias solicitudes del mismo tema en fechas (fechaSolicitud) diferentes. Obsérvese que **fechaSolicitud** forma parte de la clave primaria.
- La tabla contiene **tres claves extranjeras**. **Alumno** es clave extranjera que referencia al alumno de la tabla alumno que hace la solicitud. La pareja **asignatura y tema** *conjuntamente forman la segunda clave extranjera* que referencia a un tema de la tabla apuntesTema. Ambas forman parte de la clave primaria junto al atributo fechaSolicitud (que no es clave extranjera).
- La tercera clave extranjera no forma parte de la clave primaria y contiene a qué copistería se han solicitado las copias del tema.

1. *(0,5 puntos)* Completa la siguiente **instrucción SQL** que crea la tabla **solicitud**. Incluye en la instrucción los tipos y restricciones de integridad que faltan según la descripción inicial de las tablas. Se pide una instrucción y no varias.

```
CREATE TABLE solicitud (
  alumno
  asignatura          varchar2(15) not null,
  tema
  fechaSolicitud
  fechaRecogida
  numDuplicados
  copistería
  colorS/N
);
```

2. *(0,5 puntos)* Escribe **las instrucciones sql** necesarias para modificar la tabla solicitud, la cual no tiene todavía filas, de manera que si la fecha de recogida no es nula, ésta deberá ser siempre posterior o igual a la fecha de solicitud y el número de duplicados puede ser únicamente 0, 1 o 2.

3. *(0,5 puntos)* Escribe las **instrucciones sql** necesarias para eliminar de las tablas de la BD, todas aquellas filas correspondientes a la titulación ITIS, que va a ser eliminada del sistema. Escribe las instrucciones en el orden en el que se deberían ejecutar. Considera que no se ha utilizado la opción cascade en la definición de ninguna clave extranjera.

4. *(0,5 puntos)* Se ha creado la tabla **solicitudesPerdidas(<u>alumno</u>, <u>copistería</u>, numCopias)**. Solicitudes perdidas son las hechas hace un tiempo (30 días o más) y no recogidas. Para cada alumno con alguna solicitud perdida, se almacena su identificador, el de la copistería y el número total de páginas encargadas por el alumno a la copistería en solicitudes perdidas (considerando las páginas del tema y el número de duplicados pedidos para el tema). La clave primaria está compuesta por los atributos alumno y copistería. Escribe la **instrucción sql** necesaria para incluir en dicha tabla la información disponible en la BD de la figura sobre solicitudes realizadas hace 30 días o más y que aún no se hayan recogido (sin fechaRecogida).

5. *(0,5 puntos)* Escribe una **consulta sql** que obtenga para cada titulación y curso el número total de solicitudes, el número de copisterías implicadas y la media de días transcurridos entre la realización de la solicitud y la recogida de la misma, para aquellas solicitudes recogidas en el año actual (no en 2008, sino en el año que sea cuando se ejecute la consulta).
6. *(0,5 puntos)* Crea una **vista** llamada **TemasDespreciados** que devuelva el identificador de asignatura y tema de aquellos temas para los que **no** se ha solicitado ninguna copia en los últimos 365 días. Elimina explícitamente los repetidos sólo si es necesario.
7. *(0,5 puntos)* Escribe una **vista** llamada **AsigTodosTemas** que devuelva todos los datos de las asignaturas (id, titulacion y curso) que tengan solicitudes para todos sus temas en los últimos 365 días. Nos interesa únicamente una solución que haga uso de la vista **TemasDespreciados**.
8. *(1 punto)* Escribe una **consulta sql** que obtenga los identificadores de las copisterías que tienen solicitudes (recogidas ya o no) de **todas** las asignaturas de la titulación "ITIG".
9. *(1 punto)* Escribe una **consulta sql** que obtenga el email del alumno, la fecha de solicitud e identificador de asignatura de ***la solicitud más reciente*** (recogida o no) de cada alumno que disponga de una rebaja superior al 10% en alguna copistería. Los alumnos pueden hacer varias solicitudes el mismo día. Si coincide que un alumno hizo varias solicitudes el último día que encargó copias, deberán aparecer los datos de todas ellas en el resultado.
10. *(1 punto)* Escribe una **consulta sql** que obtenga *todos* los datos de los alumnos que **no** han solicitado copias en color de más de 10 asignaturas diferentes a la misma copistería.
11. *(1 punto)* Escribe una **consulta sql** que obtenga los email, sin repeticiones, de los alumnos que han pedido (recogidas o no) **varias** solicitudes (diferentes) en color de los mismos apuntes de un tema, cuando figura en la BD que ese tema no es interesante imprimirlo en color.
12. *(0,5 puntos)* Escribe una **consulta en álgebra relacional** que muestre el email de los alumnos que han solicitado copias (recogidas o no) de apuntes de **todas** las asignaturas que cursan (al menos de un tema de cada una).

ALGUNAS FUNCIONES (puede que no hagan falta todas)

Nombre(argumentos)	Significado	Ejemplos
sysdate	Devuelve la fecha actual	**sysdate** → '15-sept-2008'
to_char(<fecha>, <formato>)	Convierte la fecha a cadena según el formato especificado	**to_char(sysdate,'YYYY')** → '2008' **to_char(sysdate,'YYYY/MM/DD')** → '2008/09/15'
to_date(<cadena>,<formato>)	Convierte la cadena dada según el formato especificado a fecha	**to_date('15-09-08','DD-MM-YY')** → [fecha de hoy]
substr(<cadena>,<inicio>,<avance>)	Extrae una subcadena de <cadena> de longitud <avance> empezando en el carácter <inicio>.	**substr('mi cadena', 5, 3)** → 'ade'

2. *(0,5 puntos)* En esta solución se han usado tipos de datos del estándar ANSI:

```
CREATE TABLE solicitud (
alumno          int not null,
asignatura      varchar(15) not null,
tema            int not null,
fechaSolicitud  date not null,
fechaRecogida   date,
numDuplicados   int not null,
copistería      varchar(15) not null,
colorS/N        bit default false not null,

primary key (alumno, asignatura, tema, fechaSolicitud),
foreign key (alumno)
            references alumno(id),
foreign key (asignatura, tema)
            references apuntesTema(asignatura, tema),
foreign key (copisteria)
            references copisteria(id),
constraint numDuplicados_positivo check(numDuplicados>0)
);
```

3. *(0,5 puntos)* Escribe **las instrucciones sql** necesarias para modificar la tabla solicitud, la cual no tiene todavía filas, de manera que si la fecha de recogida no es nula, ésta deberá ser siempre posterior o igual a la fecha de solicitud y el número de duplicados puede ser únicamente 0, 1 o 2.

```
alter table solicitud
   add constraint     orden_fechas check (fechaRecogida is null or
                                          fechaSolicitud<=fechaRecogida);
alter table solicitud
   drop constraint    numDuplicados_positivo;
alter table solicitud
   add constraint     rango_duplicados check (numDuplicados between 0 and 2);
```

4. *(0,5 puntos)* Escribe las **instrucciones sql** necesarias para eliminar de las tablas de la BD, todas aquellas filas correspondientes a la titulación ITIS, que va a ser eliminada del sistema. Escribe las instrucciones en el orden en el que se deberían ejecutar. Considera que no se ha utilizado la opción cascade en la definición de ninguna clave extranjera.

```
delete from cursa
where asignatura in (select id from asignatura where titulacion='ITIS');

delete from solicitud
where asignatura in (select id from asignatura where titulacion='ITIS');

delete from apuntesTema
where asignatura in (select id from asignatura where titulacion='ITIS');

delete from asignatura
where titulacion='ITIS';
```

5. *(0,5 puntos)* Se ha creado la tabla **solicitudesPerdidas(<u>alumno</u>, <u>copistería</u>, numCopias)**. Solicitudes perdidas son las hechas hace un tiempo y no recogidas. Para cada alumno con alguna solicitud perdida, se almacena su identificador, el de la copistería y el número total de páginas encargadas por el alumno a la copistería en solicitudes perdidas (considerando las páginas del tema y el número de duplicados pedidos para el tema). La clave primaria está compuesta por los atributos alumno y copistería. Escribe la **instrucción sql** necesaria para incluir en dicha tabla la información sobre solicitudes realizadas hace 30 días o más y que aún no se hayan recogido (sin fechaRecogida).

```
insert into solicitudesPerdidas
   select S.alumno, S.copisteria, sum(S.numDuplicados*AT.paginas)
   from   solicitud S join
          apuntesTema AT on (S.asignatura, S.tema) = (AT.asignatura, AT.tema)
   where S.fechaRecogida is null and S.fechaSolicitud<=sysdate-30
   group by S.alumno, S.copisteria
```

6. *(0,5 puntos)* Para cada titulación y curso obtener el número total de solicitudes, el número de copisterías implicadas y la media de días transcurridos entre la realización de la solicitud y la recogida de la misma, para aquellas solicitudes recogidas en el año actual (en el año que sea cuando se ejecute la consulta).

```
select    A.titulacion, A.curso, count(*), count(distinct S.copisteria),
          avg(S.fechaRecogida-S.fechaSolicitud)
from      asignatura A join
          solicitud  S on A.id=S.asignatura
where     to_char(S.fechaRecogida,'yyyy') = to_char(sysdate,'yyyy')
group by A.titulacion, A.curso
```

7. *(0,5 puntos)* Crea una **vista** llamada **TemasDespreciados** que devuelva el identificador de asignatura y tema de aquellos temas para los que no se ha solicitado copia en los últimos 365 días. Elimina explícitamente los repetidos sólo si es necesario.

```
create view TemasDespreciados as
   select AT.asignatura, AT.tema
   from    apuntesTema AT
   minus
   select S.asignatura, S.tema
   from    solicitud S
   where  S.fechaSolicitud>=sysdate-365
```

8. *(0,5 puntos)* Escribe una **vista** llamada **AsigTodosTemas** que devuelva todos los datos de las asignaturas (id, titulacion y curso) que tengan solicitudes para todos sus temas en los últimos 365 días. Nos interesa únicamente una solución que haga uso de la vista **TemasDespreciados**.

```
create view AsigTodosTemas as
   select  A.*
   from    asignatura A
   where  A.id not in
            (select TD.asignatura
             from TemasDespreciados TD)
```

9. *(1 punto)* Escribe una **consulta sql** que obtenga los identificadores de las copisterías que tienen solicitudes (recogidas ya o no) de todas las asignaturas de la titulación "ITIG".

```
select  C.id
from    copisteria C
where   not exists
        (select A.id
         from   asignatura A
         where  A.titulacion="ITIG"
         minus
         select S.asignatura
         from   solicitud S
         where  S.copisteria=C.id)
```

10. *(1 punto)* Escribe una **consulta sql** que obtenga el email del alumno, la fecha de solicitud e identificador de asignatura de ***la solicitud más reciente*** de cada alumno que disponga de una rebaja superior al 10% en alguna copistería. Los alumnos pueden hacer varias solicitudes el mismo día. Si coincide que un alumno hizo varias solicitudes el último día que encargó copias, deberán aparecer los datos de todas ellas en el resultado.

```
select  A.email, S.fechaSolicitud, S.asignatura
from    alumno A join
        solicitud S on A.id=S.alumno
where   exists (select * from cliente C where C.alumno=A.id and rebaja>10) and
        S.fechaSolicitud=
        (select max(S2.fechaSolicitud) from Solicitud S2 where A.id=S2.alumno)
```

11. *(1 punto)* Escribe una **consulta sql** que obtenga *todos* los datos de los alumnos que **no** hayan solicitado copias en color de más de 10 asignaturas a la misma copistería.

```
select  A.*
from    alumno A
where   not exists
        select S.copisteria
        from solicitud S
        where S.alumno=A.id and S.colorS/N
        group by S.copisteria
        having count(distinct S.asignatura) > 10
```

12. *(1 punto)* Escribe una **consulta sql** que obtenga los email, sin repeticiones, de los alumnos que hayan pedido **varias** solicitudes (diferentes) en color de los mismos apuntes de un tema, cuando figura en la BD que ese tema no es interesante imprimirlo en color.

```
select distinct   A.email
from    ((solicitud S1 join
          solicitud S2 on (S1.alumno, S1.asignatura, S1.tema) =
                          (S2.alumno, S2.asignatura, S2.tema) ) join
          apuntesTema AT on  (S1.asignatura, S1.tema) = (AT.asignatura, AT.tema) ) join
          alumno A on S1.alumno=A.alumno
where   S1.fechaSolicitud<>S2.fechaSolicitud → que las solicitudes sean distintas
        and S1.colorS/N and S2.colorS/N and not AT.colorS/N
```

Otra forma:

```
select distinct   A.email
from    ( solicitud S1 join
            apuntesTema AT on  (S1.asignatura, S1.tema) = (AT.asignatura, AT.tema) ) join
            alumno A on S1.alumno=A.alumno
where  S1.colorS/N and not AT.colorS/N
group by A.email, S1.asignatura, S1.tema → grupos de mismos apuntes de un tema de un alumno
having     count(*)>1
```

13. *(0,5 puntos)* Escribe una **consulta en álgebra relacional** que muestre el email de los alumnos que hayan solicitado copias de **todas** las asignaturas que cursan (al menos de un tema de cada una).

En sql se escribiría, por ejemplo, así:

```
select  A.email
from    alumno A
where   not exists
        (select   C.asignatura
         from     cursa C
         where    C.alumno=A.id --correlación
         minus
         select   S.asignatura
         from     solicitud S
         where    S.alumno=A.id --correlación)
```

Nos piden su escritura en álgebra relacional:

Esto no sirve, (obtendría los email que figuran junto a todas las parejas alumno-asignatura de cursa):

~~Cursa_email ← $\pi_{email, alumno, asignatura}$ (cursa |X|$_{alumno=id}$ alumno)~~
~~Solicitud_asignatura ← $\pi_{alumno, asignatura}$ (solicitud)~~
~~Resultado ← (Cursa_email ÷ Solicitud_asignatura)~~

Solic ← $\pi_{alumno, asignatura}$ (solicitud)
NoSolic ← cursa – Solic
AlumResul (alum) ← π_{id} (alumno) – π_{alumno}(NoSolic)
Resultado ← π_{email} (AlumResul |X|$_{alum=id}$ alumno)