Estructuras de Datos
Grado en Ingeniería Informática, 2º curso
Ejemplos resueltos

1. Escribir en SQL las sentencias necesarias para almacenar la información sobre vuelos comerciales incluyendo los siguientes datos:
    - Aeropuertos con código (tres letras, único) y ciudad (una ciudad puede tener varios aeropuertos).
    - Vuelos con número (único), origen y destino (códigos de aeropuerto), hora de salida, nº de plazas (suponiendo que siempre fuese el mismo modelo de avión), y línea aérea que los opera.
    - Líneas aéreas con nombre y abreviatura (única).
    - Pasajeros con DNI y nombre.
    - Reservas de vuelos por pasajeros en una fecha dada, con su precio.

```
create table aeropuerto (
    codigo varchar(3) primary key,
    ciudad text
);

create table compania (
    abreviatura varchar(3) primary key,
    nombre text
);

create table pasajero (
    dni varchar(12) primary key,
    nombre text
);

create table reserva (
    dni varchar(12) references pasajero,
    numero numeric(4) references vuelo,
    fecha date,
    precio numeric,
    primary key (dni, numero, fecha)
);

create table vuelo (
    numero numeric(4) primary key,
    origen varchar(3) references aeropuerto,
    destino varchar(3) references aeropuerto,
    linea varchar(3) references compania,
    hora time,
    plazas smallint
);
```

2. Escribir en SQL las consultas que permitan obtener la siguiente información en la base de datos del ejercicio anterior:
    a. Todos los vuelos entre Madrid y París con salida antes de las 12:00.
    b. Nombre de pasajeros que vuelan de Londres a París, indicando la fecha.
    c. Nombre de pasajeros que vuelan entre Londres y París, en cualquier sentido.
    d. Pasajeros que hacen un trayecto de ida y vuelta en el mismo día.

a.
```
select numero from vuelo, aeropuerto as salida, aeropuerto as llegada
where origen = salida.codigo and destino = llegada.codigo
and salida.ciudad = 'Madrid' and llegada.ciudad = 'Paris' and hora < '12:00'
```

b.
```
select nombre, fecha from pasajero natural join reserva natural join vuelo
    join aeropuerto as salida on origen = salida.codigo
    join aeropuerto as llegada on destino = llegada.codigo
where salida.ciudad = 'Londres' and llegada.ciudad = 'Paris'
```

c.
```
select nombre, fecha from pasajero natural join reserva natural join vuelo
    join aeropuerto as salida on origen = salida.codigo
    join aeropuerto as llegada on destino = llegada.codigo
where (salida.ciudad = 'Londres' and llegada.ciudad = 'Paris')
        or (salida.ciudad = 'Paris' and llegada.ciudad = 'Londres')
```

d.
```
select nombre, ida.fecha
from (pasajero natural join reserva as ida natural join vuelo as vuelo_ida
            join aeropuerto as salida_ida on vuelo_ida.origen = salida_ida.codigo
            join aeropuerto as llegada_ida on vuelo_ida.destino = llegada_ida.codigo),
     (reserva as vuelta natural join vuelo as vuelo_vuelta
            join aeropuerto as salida_vuelta on vuelo_vuelta.origen = salida_vuelta.codigo
            join aeropuerto as llegada_vuelta on vuelo_vuelta.destino = llegada_vuelta.codigo)
where salida_ida.ciudad = llegada_vuelta.ciudad and salida_vuelta.ciudad = llegada_ida.ciudad
and ida.dni = vuelta.dni and ida.fecha = vuelta.fecha
```

3. Escribir en SQL las consultas que permitan obtener la siguiente información en la base de datos anterior:
    a. Líneas que no tienen vuelos con salida desde Londres.
    b. Vuelos completos, indicando la fecha.
    c. Vuelos vacíos (sin reservas) en 2011.
    d. Líneas que sólo operan vuelos con salida o llegada a Madrid.

a.
```
select nombre from compania
where not exists (select * from vuelo join aeropuerto on origen = codigo
                    where linea = abreviatura and ciudad = 'Londres')
```

```
-- o bien, otra idea:

select nombre from compania natural join
                        ((select abreviatura from compania)
                            except (select linea from vuelo join aeropuerto on origen = codigo
                                    where ciudad = 'Londres')) as x
```

b.
```
select numero, fecha from reserva natural join vuelo
group by numero, fecha, plazas /* plazas se incluye sólo para poderlo usar en la condición having */
having count(*) = plazas
```

c.
```
select * from vuelo where not exists (select * from reserva where reserva.numero = vuelo.numero)
```

d.
```
select nombre from compania where not exists
    (select * from vuelo, aeropuerto as salida, aeropuerto as llegada
     where linea = abreviatura and origen = salida.codigo and destino = llegada.codigo
     and salida.ciudad <> 'Madrid' and llegada.ciudad <> 'Madrid')
```

4. Escribir en SQL las consultas que permitan obtener la siguiente información en la base de datos anterior:
    a. Aeropuerto con más tráfico (contando salidas y llegadas).
    b. Líneas aéreas ordenadas por el nº total de reservas en los vuelos que operan.
    c. Nombre de la ciudad desde la que sale el primer vuelo del día.
    d. Gasto total promedio por pasajero.
    e. Facturación total por línea aérea y aeropuerto de salida.

a.
```
select origen from
    (select origen, count(*) as nsalidas from vuelo group by origen) as salidas,
    (select destino, count(*) as nllegadas from vuelo group by destino) as llegadas
where origen = destino
order by nsalidas + nllegadas desc limit 1
```

b.
```
select nombre from compania
order by (select count (*) as n from reserva natural join vuelo where linea = abreviatura) desc

-- o bien, para sacar el nº:

select nombre, count (*) as n from reserva natural join vuelo join compania on linea = abreviatura
group by nombre
order by n desc
```

c.
```
select ciudad from aeropuerto, vuelo
where origen = codigo and hora <= all (select hora from vuelo)
```

d.
```
select avg (gasto) from
    (select sum (precio) as gasto from reserva natural join pasajero group by dni) as gastos
```

e. select abreviatura, codigo, sum (precio) from reserva natural join vuelo
join compania on linea = abreviatura
join aeropuerto on origen = codigo
group by abreviatura, codigo

5. Dadas las siguientes tablas:

Artista

aid	nombre
0	W. Allen
1	D. Keaton
2	M. Farrow
3	G. Garbo
4	B. Lugosi
5	E. Lubitsch

Pelicula

pid	titulo	fecha
0	Interiors	1978
1	Broadway Danny Rose	1984
2	Ninotchka	1939
3	Crimes and misdemeanors	1989
4	The Godfather	1972
5	Flesh and the Devil	1926

Cine

cid	nombre
0	Renoir
1	Berlanga
2	Golem

Sala

cid	sid
0	1
0	2
1	1
2	1
2	2
2	3

Reparto

pid	aid	papel
0	1	Renata
1	0	Danny Rose
1	2	Tina Vitale
2	3	Ninotchka
2	4	Razinin
3	0	Cliff Stern
3	1	Halley Reed
4	1	Kay Adams
5	3	Felicitas

Direccion

pid	aid
0	0
1	0
2	5
3	0

Cartelera

cid	sid	pid
0	1	2
0	2	1
1	1	0
2	1	0
2	2	2
2	3	3

Escribir en SQL las siguientes consultas:

a. Nombre de actores que alguna vez han interpretado más de un papel en la misma película.
b. Nombre de directores que nunca han trabajado como actores.
c. Nombre del director con más salas proyectando alguna de sus películas, **utilizando sólo tres tablas**.
d. Títulos de películas que se proyectan en una única sala **sin utilizar operadores de agregación**.

a. select nombre from reparto as r1, reparto as r2 natural join artista
where r1.pid = r2.pid and r1.aid = r2.aid and r1.papel <> r2.papel

b. select nombre from artista natural join direccion
where aid not in (select aid from reparto)

c. select nombre, count(*) as n
from artista natural join direccion natural join cartelera
group by aid, nombre order by n desc limit 1

d. select titulo from pelicula
where not exists (select * from cartelera as p1, cartelera as p2
where p1.pid = pelicula.pid and p2.pid = p1.pid
and (p2.cid <> p1.cid or p2.sid <> p1.sid))

6. Determinar la forma normal de los siguientes esquemas, para los que se indican todas las dependencias funcionales existentes entre sus atributos.
   a. R(a, b, c, d)
      $a \rightarrow b$ $b \rightarrow c$
   b. R(a, b, c, d)
      $a \rightarrow b$ $b \rightarrow c$ $c \rightarrow a$
   c. R(a, b, c, d, e)
      $\{a, b\} \rightarrow \{c, d\}$ $c \rightarrow a$ $d \rightarrow e$

a. La única clave del esquema es {a, d}.
   La dependencia $a \rightarrow b$ incumple 2NF, por tanto el esquema es 1NF.

b. En este caso hay más claves: {a, d}, {b, d}, {c, d}.
   En las tres dependencias, la izquierda no es una (super)clave, por lo tanto no es BCNF. Pero en los tres casos la derecha es un atributo primo, por lo que sí se acepta por 3NF. Por tanto el esquema es 3NF.

c. Hay dos claves: {a, b}, {b, c}.
   La dependencia $\{a, b\} \rightarrow \{c, d\}$ es BCNF porque {a, b} es una clave.
   La dependencia $c \rightarrow a$ es 3NF porque c no es una clave pero a es primo.
   La dependencia $d \rightarrow e$ es 2NF porque d no es una clave y e no es primo, pero tampoco es d parte incompleta de una clave.
   Por tanto el esquema es 2NF.

7. Descomponer en esquemas 3NF las relaciones del ejercicio anterior que no lo sean. Indicar qué claves contienen los esquemas resultantes.

a. Las dependencias ya forman una cobertura mínima porque no tienen redundancia y la parte derecha es un solo atributo en todos los casos.
   No hay más de una dependencia con la misma parte izquierda por lo que no procede agrupar. Creamos un esquema por cada dependencia:
   R1(a, b), R2(b, c)
   Como ninguno de estos esquemas contiene a ninguna de las claves originales (sólo había una), creamos un esquema con alguna de las claves (la única que había):
   R3(a, d)
   No hay ningún esquema redundante y hemos terminado.

   Las únicas claves son a en R1, b en R2, y {a, d} en R3. Nótese que los esquemas resultan ser BCNF (aunque no tendrían por qué).

b. Las dependencias ya forman una cobertura mínima. Creamos un esquema por cada dependencia:
   R1(a, b), R2(b, c), R3(a, c)
   Como ninguno de estos esquemas contiene a ninguna de las tres claves originales, creamos un esquema con alguna de ellas, por ejemplo:
   R4(a, d)
   No hay ningún esquema redundante y hemos terminado.

Las únicas claves son a en R1, b en R2, c en R3, y {a, d} en R4. Nótese que los esquemas resultan ser BCNF (aunque no tendrían por qué).

c. Las dependencias en cobertura mínima son (sólo se necesita separar la primera en dos, separando los atributos de su parte derecha):

$\{a, b\} \rightarrow c$ $\{a, b\} \rightarrow d$ $c \rightarrow a$ $d \rightarrow e$

Agrupando las dos primeras que tienen la misma parte izquierda, creamos tres esquemas

R1(a, b, c, d), R2(a, c), R3(d, e)

R1 contiene ya alguna de las claves originales (de hecho todas, aunque bastaría con una), por tanto no es necesario añadir más esquemas.

Por otra parte R2 es redundante: está contenido en R1, así pues lo eliminamos, y el resultado final es:

R1(a, b, c, d), R3(d, e)

Las claves son {a, b}, {b, c} en R1, y d en R3.

Nótese que R3 resulta ser BCNF, pues su única dependencia es con la clave, d. Sin embargo en este caso R1 está en 3NF pero no BCNF, pues todas las dependencias con una clave, excepto $c \rightarrow a$. Esta dependencia no cumple por tanto no es BCNF, pero sí 3NF pues a es primo.

A cambio, todas las dependencias originales se han preservado: están todas presentes globalmente en los nuevos esquemas.

8. Mismo ejercicio para BCNF. Indicar además qué claves externas aparecen.

a. Tomamos una dependencia que incumpla BCNF. Ninguna de las dos dependencias es BCNF así que tomamos por ejemplo $a \rightarrow b$, y separamos R en dos esquemas:

R1(a, c, d), R2(a, b)

La única clave en R2 es a, y la única dependencia $a \rightarrow b$ en R2 ahora es BCNF, luego R2 ya es BCNF.

En R1 se ha perdido la dependencia $b \rightarrow c$ al separar b a R2. Pero queda una dependencia $a \rightarrow c$, que se infiere de las dos dependencias originales. Y esta dependencia no es BCNF pues la única clave de R1 sigue siendo {a, d}, por tanto descomponemos R1 en:

R3(a, d), R4(a, c)

La clave de R3 es {a, d}, y la única clave de R4 es a, de modo que estos dos esquemas ya son BCNF.

El resultado final es pues: R2(a, b), R3(a, d), R4(a, c).

Las claves son a, {a, d} y a en R2, R3 y R4 respectivamente. Las claves externas son:

R3.a references R4.a

R4.a references R2.a

Nótese que se ha perdido la dependencia $b \rightarrow c$. Obsérvese también que en este caso el resultado difiere de la normalización a 3NF en el ejercicio anterior (aunque podría pasar que los dos métodos den el mismo resultado).

b. Empezamos por ejemplo con $a \rightarrow b$, separando R en dos esquemas:

R1(a, c, d), R2(a, b)

Donde igual que en el apartado anterior, R2 ya es BCNF.
En R1 tenemos c → a que no es BCNF (podríamos igualmente razonar con la dependencia indirecta a → c igual que antes, lo cual daría lugar a otro resultado, esencialmente igual al apartado anterior), y descomponemos en:
R3(c, d), R4(a, c)
R4 tiene dos claves: a y c, con una dependencia mutua BCNF entre ellas. R4 es por tanto BCNF. R3 tiene una única clave: {c, d}, luego ya es BCNF.

El resultado final es pues: R2(a, b), R3(c, d), R4(a, c).

La clave de R2 es a, la de R3 es {c, d}, y las de R4 son a y c. Las claves externas son:
R3.c references R4.c
R4.a references R2.a

De nuevo, se ha perdido la dependencia b → c.

c. Empezamos por ejemplo con c → a, separando R en dos esquemas:
R1(b, c, d, e), R2(a, c)
Nótese que acabamos de perder la dependencia {a, b} → {c, d} al separar los atributos a y b.
La única clave de R2 es c, y la única dependencia c → a es pues BCNF, luego R2 también lo es.

En R1 la única clave es {b, c} y la única dependencia no BCNF es d → e. Descomponemos:
R3(b, c, d), R4(d, e)
La clave de R4 es d y el esquema es BCNF con su única dependencia d → e. La única clave de R3 es {b, c} y no hay más dependencia que {b, c} → d, por lo que R3 también es BCNF y hemos terminado.

El resultado final es pues: R2(a, c), R3(b, c, d), R4(d, e).

La claves son c, {b, c}, y d en R2, R3 y R4 respectivamente. Las claves externas son:
R3.d references R4.d
R4.c references R2.c

9. Tenemos una base de datos de una tienda online de libros con el siguiente estado y las claves primarias y externas que se indican.

Usuario

id	nombre
1	María
2	Carlos
3	Eva

Libro

isbn	titulo	idAutor	idioma	precio
500	Lengua de cal	3	castellano	25
300	La vie devant soi	2	francés	10
200	Edisto	1	inglés	5
400	Mansura	3	castellano	12
100	The interrogative mood	1	inglés	15

Autor

idAutor	nombre
1	Powell
2	Gary
3	Azúa

Compra

idCompra	id	isbn	fecha
1	3	200	2012-12-21
2	2	400	2012-12-22
3	2	400	2012-12-22
4	1	500	2012-12-27
5	3	100	2012-12-27
6	2	300	2013-01-09
7	1	400	2013-01-10
8	1	300	2013-01-13

Claves externas:
Libro.idAutor → Autor.idAutor
Compra.id → Usuario.id
Compra.isbn → Libro.isbn

a. Escribir las siguientes consultas en cálculo relacional, álgebra relacional, y SQL.

Nombre de usuarios que han comprado “Mansura”.

$$\{ u.nombre \mid Usuario(u) \; and \; \exists l, c(Libro(l) \; and \; Compra(c) \; and \; l.titulo = 'Mansura' \; and \; c.id = u.id \; and \; c.isbn = l.isbn) \}$$

$$\pi_{nombre}(Usuario \bowtie Compra \bowtie \sigma_{titulo='Mansura'}(Libro))$$

```
select nombre from Usuario natural join Compra natural join Libro
where titulo = 'Mansura'
```

Nombre de usuarios que no han comprado nada.

$$\{ u.nombre \mid Usuario(u) \; and \; not \; \exists c(Compra(c) \; and \; c.id = u.id) \}$$

$$\pi_{nombre}(Usuario) - \pi_{nombre}(Usuario \bowtie Compra)$$

```
select nombre, sum(precio)
from Autor natural join Compra natural join Libro
group by idAutor, nombre
```

Títulos de libros comprados por Eva.

$$\{ l.titulo \mid Libro(l) \; and \; \exists u, c(Usuario(u) \; and \; Compra(c) \; and \; u.nombre = 'Eva' \; and \; c.id = u.id \; and \; c.isbn = l.isbn) \}$$

$$\pi_{titulo}(\sigma_{nombre='Eva'}(Usuario) \bowtie Compra \bowtie Libro)$$

```
select titulo from Usuario natural join Compra natural join Libro
where nombre = 'Eva'
```

Títulos de libros que no se han vendido (sólo en cálculo y álgebra).

$$\{ l.título \mid Libro(l) \; and \; not \; \exists c(Compra(c) \; and \; c.isbn = l.isbn) \}$$

$$\pi_{título}(Libro) - \pi_{título}(Libro \bowtie Compra)$$

Gasto total de cada usuario, junto con el nombre de éste (sólo en SQL).

```
select nombre, sum(precio)
from Usuario natural join Compra natural join Libro
group by id, nombre
```

b. Mostrar el resultado de las siguientes consultas.

$$\{ c_1 \mid Compra(c_1) \; and \; not \; \exists c_2 (Compra(c_2) \; and \; c_2.fecha > c_1.fecha) \}$$

La compra más reciente:
{ (8, 1, 300, '2013-01-13') }

$$\{ a_1, a_2 \mid Autor(a_1) \; and \; Autor(a_2) \; and \; a_1.idAutor \neq a_2.idAutor \; and \; not \; \exists l_1, l_2, c_1, c_2 (Libro(l_1) \; and \; Libro(l_2) \; and \; Compra(c_1) \; and \; Compra(c_2) \; and \; l_1.idAutor = a_1.idAutor \; and \; l_2.idAutor = a_2.idAutor \; and \; c_1.isbn = l_1.isbn \; and \; c_2.isbn = l_2.isbn \; and \; c_1.id = c_2.id)) \}$$

Pares de autores que no han vendido libros a ningún usuario en común:
{ (1, 'Powell', 2, 'Gary'), (2, 'Gary', 1, 'Powell') }

$$\{ l_1 \mid Libro(l_1) \; and \; not \; \exists l_2 (Libro(l_2) \; and \; l_2.precio > l_1.precio) \}$$

El libro más caro:
{ (500, 'Lengua de cal', 3, 'castellano', 25) }

$$\{ u_1, u_2 \mid Usuario(u_1) \; and \; Usuario(u_2) \; and \; u_1.id \neq u_2.id \; and \; not \; \exists l, c_1, c_2 (Libro(l) \; and \; Compra(c_1) \; and \; Compra(c_2) \; and \; c_1.isbn = l.isbn \; and \; c_2.isbn = l.isbn \; and \; c_1.id = u_1.id \; and \; c_2.id = u_2.id)) \}$$

Pares de usuarios que no han comprado ningún libro en común:
{ (1, 'María', 3, 'Eva'),
(2, 'Carlos', 3, 'Eva'),
(3, 'Eva', 1, 'María')
(3, 'Eva', 2, 'Carlos') }

$$Autor - \pi_{idAutor,nombre} \left(Autor \bowtie \sigma_{idioma \neq 'inglés'} (Libro) \right)$$

Autores que sólo han escrito libros en inglés (o no han escrito ningún libro):
{ (1, 'Powell') }

$$\pi_{nombre} \left(Autor \bowtie Libro \bowtie \left(\sigma_{idCompra \neq idc} \left(Compra \bowtie_{isbn} \rho_{idCompra/idc} (Compra) \right) \right) \right)$$

Nombre de autores que han vendido más de un ejemplar del mismo libro:
{ ('Azúa'), ('Gary') }

$$Usuario - \pi_{id,nombre}\left(Usuario \bowtie Compra \bowtie \sigma_{idioma \neq 'inglés'}(Libro)\right)$$

Usuarios que sólo han comprado libros en inglés (o no han comprado ningún libro):
{ (3, 'Eva') }

$$\pi_{nombre}\left(Usuario \bowtie \left(\sigma_{idCompra \neq idc}\left(Compra \bowtie_{id,isbn} \rho_{idCompra/idc}(Compra)\right)\right)\right)$$

Nombre de usuarios que han comprado más de un ejemplar del mismo libro:
{ ('Carlos') }

```
select nombre, sum(precio) as gasto from Compra natural join Libro
                                                  natural join Usuario
group by id, nombre
order by gasto desc limit 1
```

El usuario que más ha gastado, junto con su gasto total:
{ ('María', 35) }

```
select u1.nombre, u2.nombre, count(distinct idAutor)
from (Usuario natural join Compra) as u1
      join (Usuario natural join Compra natural join Libro) as u2
      on u1.isbn = u2.isbn
where u1.id < u2.id
group by u1.id, u1.nombre, u2.id, u2.nombre
```

Nº de compras de libros del mismo autor en común de cada par de usuarios:
{ ('María', 'Carlos', 1), ('Carlos', 'Eva', 1) }

```
select titulo, count(*) as ventas from Compra natural join Libro
group by isbn, titulo
order by ventas desc limit 1
```

El título más vendido, con su nº de ventas:
{ ('Mansura', 3) }

```
select a1.nombre, a2.nombre, count(distinct id)
from (Compra natural join Libro natural join Autor) as a1
      join (Compra natural join Libro natural join Autor) as a2
      on a1.id = a2.id
where a1.idAutor < a2.idAutor
group by a1.idAutor, a1.nombre, a2.idAutor, a2.nombre
```

Nº de compradores en común de cada par de autores:
{ ('Gary', 'Azúa', 2) }

10. Tenemos un fichero con un millón de registros de longitud fija de 1.000 bytes cada uno en un disco magnético con las siguientes características:
    - 100 sectores por pista.
    - 8 sectores por bloque.
    - 500 bytes por sector.
    - Un seek promedio de 9 ms.
    - Una velocidad de giro de 10.000 rpm.

    Estimar el tiempo necesario para leer el fichero completo en los supuestos que se describen a continuación.

    Respecto a la fragmentación del fichero:

    a. Mejor caso: los datos del fichero ocupan una región compacta del disco (es decir, un supuesto ideal en el que la fragmentación es nula).
    b. Peor caso: el fichero está fragmentado al máximo

    Respecto al programa que lee los datos:

    i. Peor caso: los registros se leen de uno en uno.
    ii. Mejor caso: se lee el fichero completo en una sola instrucción.
    iii. Término medio: se leen los datos utilizando un buffer de lectura de 60.000 bytes.

Los cálculos se pueden organizar por registro, o mejor, por fragmentos que se van a leer sin "saltos" entre medias, y finalmente multiplicar el coste por fragmento por el número de fragmentos. La cuestión es por tanto identificar en cada caso qué fragmentos son ésos. También es posible separar el cálculo de coste por operaciones seek + latencia por un lado, y transferencia por otro, ya que el coste por transferencia es independiente de los saltos que se realicen entre lecturas. Aquí vamos a seguir esta opción.

Hacemos algunos cálculos preliminares útiles para todos los casos.

La velocidad de rotación en revoluciones por milisegundo es 10.000rpm / 60 / 1000 = 1/6 rotaciones por ms. Por tanto, una rotación (completa) tarda 6ms, y una rotación promedio se estima como 3ms.
Cada operación de seek + latencia supone 9ms + 3ms = 12 ms.

Cada bloque tiene 8 · 500 bytes = 4.000 bytes. En cada bloque caben pues 4.000 bytes / 1.000 bytes = 4 registros. Los registros caben de forma exacta en los bloques (por ser el tamaño de bloque múltiplo del tamaño de registro), lo que simplifica los cálculos.

El fichero ocupa 10^6 registros / 4 registros por bloque = 250.000 bloques.

El buffer del caso iii también es un múltiplo exacto del tamaño de registro y de bloque, y simplificará también los cálculos: en el buffer caben 60.000 bytes / 1.000 registros = 60 registros; caben los datos de 60.000 bytes / 4.000 bytes = 15 bloques.

a. Al no haber fragmentación, las únicas operaciones de seek + latencia se deberán a "interrupciones" en la lectura producidas en el programa.
    i. Cada registro da lugar a un seek + rotación. Además se lee el bloque completo en el que esté almacenado el registro (a pesar de que en un bloque caben 4 registros), pues es la mínima unidad de lectura/escritura en cualquier caso, lo que da lugar a una parte de lectura repetida y

redundante (no aprovechada) de registros.[1]
Se producen por tanto un millón de lecturas de bloques (una por registro). De aquí se derivan pues:

- Seek + latencia: $10^6 \cdot 12ms = 12.000s = 200$ min
- Transferencia: $10^6 \cdot$ bytes en cada bloque / bytes en cada pista $\cdot$ tiempo de 1 rotación = $= 10^6 \cdot 4.000$ bytes / (100 bytes $\cdot$ 500 sectores) $\cdot$ 6ms = 8 min

Total: **208 min**

ii. Se produce un único (y necesario) movimiento de seek + latencia (9ms) y el programa no produce ninguno adicional. El coste de transferencia es el estrictamente necesario para leer los datos (una sola vez, sin redundancia):

- Seek + latencia: 12ms
- Transferencia: 10^6 registros $\cdot$ 1.000 bytes / 50.000 bytes (por pista) $\cdot$ 6 ms = 2 min

Total: **2 min 12 ms**

iii. El buffer se llena sin interrupciones por parte del programa. Así pues se producirá únicamente un seek + latencia cada vez que se llene el buffer. Para leer todos los registros, el buffer se tiene que leer (10^6 registros / 60 registros por buffer) = $5/3 \cdot 10^4$ veces. La transferencia se produce sin repetición de lecturas, pues el tamaño del buffer excede el tamaño de bloque, y es un múltiplo exacto de éste. Tenemos pues:

- Seek + latencia: $5/3 \cdot 10^4 \cdot 12ms = 3$ min 20 s
- Transferencia: 2 min (igual que para el caso ii)

Total: **5 min 20 s**

b. Como mínimo se va a producir un movimiento seek + latencia por bloque.

i. Como cada registro cabe en un bloque, el hecho de que los bloques estén separados resulta indiferente.[2] El coste es el mismo: **208 min**.

ii. El coste de transferencia no cambia respecto al caso a, pero se producen seek + latencia adicionales: uno por bloque.

- Seek + latencia: 12ms $\cdot$ 250.000 bloques = 50 min
- Transferencia: 10^6 registros $\cdot$ 1.000 bytes / 50.000 bytes (por pista) $\cdot$ 6 ms = 2 min

Total: **52 min**

iii. El buffer no evita los seek + latencia por los cambios de bloque que tienen lugar dentro del llenado del buffer. Se evitan las lecturas redundantes del caso i, y por lo demás todo resulta igual que en el caso ii: **52 min**.

---

[1] Si los registros abarcasen varios bloques, la única redundancia sería el sobrante de datos del último bloque abarcado por el registro. Si la longitud de los registros fuese múltiplo exacto de la longitud de bloque, ese sobrante sería cero.
[2] Si los registros ocupasen varios bloques sí se producirían saltos adicionales en la lectura de cada registro (tantos como bloques abarcase cada registro).

11. En un fichero en el que se almacenan registros de longitud variable con indicador de longitud tenemos los siguientes datos (se trata obviamente de un ejemplo simplificado donde los registros contienen un solo campo):

```
04Roma04Oslo06Lisboa13Santo Domingo05París
```

Supongamos que borramos Oslo, insertamos Atenas, borramos Santo Domingo, borramos Lisboa, e insertamos Berlín, en este orden. Mostrar el estado del fichero, con la lista de espacios libres, antes y después de la inserción de Berlín, según las estrategias de a) first-fit y b) worst-fit.
Indicación: para simplificar, supóngase que tanto el indicador de longitud como los punteros de la lista de espacios libres ocupan 2 bytes.

a) Head: 12

```
00    06    12      20             35     42
04Roma04*-1 06*20   13*06          05París06Atenas
```

Head: 20

```
00    06    12      20             35     42
04Roma04*-1 06Berlín13*06          05París06Atenas
```

b) Head: 20

```
00    06    12      20             35     42
04Roma04*-1 06*06   13*12          05París06Atenas
```

Head: 12

```
00    06    12      20      28     35     42
04Roma04*-1 06*28   06Berlín05*06  05París06Atenas
```