11078 Base de Datos II, Ejercicio BDB

Ud. tiene las siguientes relaciones de un viejo negocio de video club:

PELICULA(IDP,TITULO) PK(IDP)
SOCIO(DNI,APENOM) PK(DNI)
ALQUILA(IDAL,FH,DNI,IDP,PRECIO,FHDEV NULL,IDE) PK(IDAL)

IDP,IDAL son datos enteros largos, contadores; FH fecha-hora de alquiler, FHDEV fecha-hora de devolución de la película que se carga en el momento en que se devuelve la película; IDE es un numero entero entre 1 y 10 que identifica al empleado que recibió la película.
Se debe poder registrar todos los alquileres y devoluciones, saber qué películas están alquiladas en este momento, saber el monto total mensual en alquileres (sum(alquila.precio) para un mes determinado), monto total mensual de alquiler recibido por cada empleado (se pagan comisiones a partir de este dato); consultas tales como: total mensual en alquileres de un cliente cualquiera, total mensual por película, etc. ALQUILA solo guarda alquileres del año actual, no hay alquileres de un año para el otro, cuando termina el año se respalda el contenido de ALQUILA y luego se borra y IDAL comienza desde 1 nuevamente. Las cajas de las películas (o DVD,CD,etc) tienen pegado el valor de su IDP. Se pretende migrar este modelo relacional a una Base de Datos de tipo Clave-Valor BDB, solo contamos con la API básica BDB (Ej: no podemos hacer cursor para recorrer las claves ya cargadas). Indique:

a) Las estructuras de datos a usar, si agrega algún dato indique que guardaría allí y en qué momento, cómo funciona, etc.
b) Las claves a utilizar, como guardaría esas claves y como podría leerlas
c) Como Analista, ¿es necesario modificar algún procedimiento dentro del negocio?

Existen varias posibilidades… aquí una posible:

a) Las estructuras de datos a usar, si agrega algún dato indique que guardaría allí y en qué momento, cómo funciona, etc.

```
struct pelicula { // clave "P%d"
   char titulo[100];
   long idal;  // 0L -> no esta alquilada, sino indica id alquiler
}
struct socio { // clave "S<dni>"
   char apenom[100];
   int nalq;   // contador de alquileres pendientes: 0..N, se incrementa
cuando alquila y se decrementa cuando devuelve
}
struct alquila { // clave "A%d"
   long idal; // 1..N
   long fh; // AAAAMMDDHHMNSS
   long dni;  // clave S<dni>
   long idp; // clave P%d
   double precio;
   long fhdev; // 0L o bien AAAAMMDDHHMNSS
   int ide; // 1..10
}
struct mes { // 0..11
   long d_idal; // idal desde (A%d) para este mes o bien 0L
   long h_idal; // idal hasta (A%d) para este mes o bien 0L
}
struct meta { // clave "M0"
   long u_idp;  // ultimo numero de pelicula usado 1..N
   long u_idal; // ultimo numero de alquiler usado 1..N
   long u_dni;  // ultimo numero de claves dni     1..N
   struct mes m[12]; // arreglo que guarda id de alquileres de cada mes
}
// guardo de a 1000 dni's con clave D1, si necesito
// guardar mas dni's genero clave D2 y asi sucesivamente
// aqui guardaria todos los dni's de todos los clientes
// que alquilaron este año o años anteriores
struct dni { // clave "D1".."DN"
   long dni[1000]; // cada elemento tiene 0L o bien un DNI valido
}
```

b) Las claves a utilizar, como guardaría esas claves y como podría leerlas

Clave	Descripción
P1..N	ABM películas, a medida que doy de alta, me aseguro que sea correlativo y voy actualizando clave M0 u_idp último número de película usado
S<dni>	ABM socios, ídem anterior, pero además, en clave D1..N guardo el nuevo dni dado de alta o también puedo buscar allí un dni existente
A1..N	ABM alquileres, ídem anterior, actualizo en M0 u_idal último número de alquiler. A medida que doy de alta en alquileres, también actualizo en M0 mes[<mes de la fecha-hora-aquiler>].d_idal y h_idal si estos campos están en cero, se actualiza con el id alquiler actual y sino, se verifica el número y se va actualizando para guardar el primer y último idal de dicho mes. Se suma/resta contador nalq en socio. Se actualiza idal en película.

c) Si como Analista, modifica algún procedimiento dentro del negocio

No sería necesario. Cuando se devuelve un video, se cuenta con idp, con dicha clave puedo leer estructura película y de allí obtener idal para acceder al alquiler correspondiente, actualizar precio y fecha-hora-devolucion, id empleado que recibió la devolución y luego poner en cero pelicula.idal.
Si el número de la película se pierde, leo M0 y puedo hacer loop de 1 a M0.u_idp búsqueda secuencial por nombre de película.
No es necesario saber dni a nombre de quien estaba el alquiler que se esta devolviendo.
Si la estructura mes[] se actualiza correctamente, se puede acceder por clave a los alquileres de cada mes y hacer todas las consultas/procesos que se necesitan.
Habrá un proceso de inicio de año para hacer backup de alquileres, borrar alquileres, actualizar M0 u_idal, mes para volver a comenzar.
Para guardar históricamente los alquileres agregar el año en 4 dígitos + idal para armar la clave primaria de alquileres históricos. También guardar históricamente (por año) la información de metadata.