



**TRABAJO PRACTICO I – MANTENIMIENTO DE CLAVE PRIMARIA Y CLAVE  
EXTRANJERA  
Version 3.0 (Julio 2025)**

Se pretende crear un manejador de archivos con las siguientes características:

- puede utilizar el lenguaje de programación que desee
- puede utilizar IA generativa para generar código, siempre y cuando comprenda el código generado y el diseño propuesto
- puede crear varios ejecutables, uno para cada comando o bien un único ejecutable tipo *shell* en donde el usuario tipea los comandos a ejecutar
- todo el contenido de los archivos es ASCII, no binario, son siempre registros de longitud fija
- <no de registro> es de base 0 a N-1
- puede hacer un borrado lógico de registros (o no, depende del diseño que Ud. implemente)
- en caso de error, emitir errores por consola y no realizar la operación solicitada
- si todo ok, no emitir ningún mensaje y realizar la operación solicitada
- una sola clave extranjera (FK) por archivo como máximo
- una sola clave primaria (PK) por archivo como máximo, obviamente
- no implementar claves candidatas
- PK, FK son alfanuméricas, y no coinciden con la ubicación física del registro
- la FK obviamente tiene el mismo largo que la PK del archivo que relaciona
- pueden mantener datos en memoria que luego se bajan a disco y viceversa
- todas las claves y datos son persistentes, es decir, si ingreso 1000 registros en N archivos, salgo de la aplicación, apago la pc, la prendo y vuelvo a entrar y los datos están disponibles nuevamente
- nombre de archivo sin espacios intermedios, ni comillas dobles, ni simples y residen todos en la carpeta actual (no indicar ruta o *path* del archivo, solo su nombre)
- <pos> es de base 0
- lo que está entre corchetes [...] (en los comandos), significa que es opcional, el resto es obligatorio

Los comandos a implementar son los siguientes (Ud. puede agregar más comandos si lo desea o imprimir más información):



```
$ crear <archivo> PK="<pos>,<largo>" LRECL="<largo
registro>" [FK="<pos>,<archivo>"]
```

- validar que <pos> y <largo> sean valores razonables dentro del registro
- validar que el archivo de la FK exista previamente (en caso de indicar FK)
- crea <archivo> desde cero, si existe previamente, lo destruye

```
$ insertar <archivo> REG="...."
```

- controlar que no se duplique PK
- controlar que el contenido de REG no supere LRECL
- controlar que el valor de FK del registro exista como valor de PK en el archivo que referencia la FK

```
$ borrar <archivo> PK="<valor pk>"
```

- no permitir borrar registro si hay FK que apuntan a este registro
- hacer borrado lógico, en lo posible, dependerá del diseño que Ud. proponga
- controlar que exista PK
- controlar que el registro ya no este borrado previamente

```
$ leer <archivo> PK="<valor pk>"
```

- lee registro a través del valor de su PK y muestra por pantalla:

```
#reg # registro
2 b aabbccc
```

La columna #reg muestra el número de registro físico dentro del archivo de datos, de base 0..(N-1)

La columna # no muestra nada cuando el registro es un registro activo, en caso de ser un registro borrado, indicará la letra "b" (borrado)

La columna registro mostrará el contenido del registro de datos (en este caso, como ejemplo, muestra el registro físico #2 borrado, cuyo registro contiene los caracteres "aabbccc")

- igual que comando mostrar pero solo muestra un registro
- si la PK no existe, mostrar PK no existe!
- obviamente, muestra registros borrados lógicos (si existieran) y no borrados



```
$ mostrar <archivo>
```

-muestra por pantalla:

```
#reg # registro
  0     ....
  1 b   ....
  2     ....
.....
```

-una línea de consola por registro, #reg indica nro de registro, # indica un "b" cuando el registro este borrado (borrado lógico, si es que Ud. implementó borrado lógico)

```
$ cambiar <archivo> PK="<valor pk anterior>" REG="..."
```

- dar de baja PK anterior, FK anterior
- no puede dar de baja PK anterior si ésta es referenciada por FK's
- dar de alta PK nueva, FK nueva
- controlar que exista previamente la nueva PK
- controlar que no se duplique PK
- controlar que exista FK
- controlar tamaño de registro y de PK
- controlar que exista <archivo>

```
$ salir
```

-permite salir del shell, en caso de implementar este tipo de aplicación

Ejemplo de uso (supongamos que implementamos una aplicación de consola):

```
$ crear cliente PK="0,3" LRECL="10"
```

Crea archivo cliente cuyos registros ocuparán 10 bytes y los 3 primeros bytes son la clave primaria

```
$ insertar cliente REG="123Ana Mar"
```

Crea un nuevo registro en el archivo cliente con PK=123 y el resto del registro "Ana Mar"

```
$ insertar cliente REG="456Pepe "
```



Crea un nuevo registro en el archivo cliente con PK=456 y el resto del registro "Pepe " (Pepe y una serie de espacios hasta completar 10 bytes)

```
$ insertar cliente REG="789Jose Ma"
```

Crea un nuevo registro en el archivo cliente con PK=789 y el resto del registro "Jose Ma"

```
$ crear factura PK="0,8" LRECL="32" FK="18,cliente"
```

Crea archivo factura cuyos registros ocuparán 32 bytes y los 8 primeros bytes son la clave primaria (número de factura), el registro se diseñó de la siguiente forma:

Número, 8 bytes, PK, formato 99999999

Fecha, 10 bytes, formato dd/mm/aaaa

Cliente, 3 bytes, FK contra archivo cliente, formato 999

Monto, 11 bytes, formato 99999999.99 (8 enteros, punto y dos decimales)

Total: 32 bytes

```
$ insertar factura REG="1234567815/07/202578920111222.15"
```

Crea un nuevo registro en el archivo factura, en donde:

Número: 12345678

Fecha: 15/07/2025

Cliente: 789

Monto: 20111222.15

Este registro de factura está "enlazado" con el registro de cliente, a través del cliente 789; la factura 12345678 es del cliente 789

```
$ insertar factura REG="1234567916/07/202544400531231.25"
```

Intenta crear un nuevo registro en el archivo factura, en donde:

Número: 12345679

Fecha: 16/07/2025

Cliente: 444

Monto: 531231.25

El sistema debería dar error, porque el cliente con código 444 no existe.

```
$ borrar cliente PK="789"
```



Pretende borrar el registro cliente cuya clave primaria es 789, el sistema debería dar error porque la factura 12345678 es de este cliente y por lo tanto, no puedo borrar al cliente, la factura quedaría sin los datos del cliente y eso no es posible.

```
$ borrar cliente PK="456"
```

El sistema no dará mensaje de error y borrará el registro correspondiente al cliente 456, pues no hay ninguna factura relacionada con este cliente.

```
$ leer factura PK="12345678"
```

El sistema debería responder:

```
#reg # registro  
0 1234567815/07/202578920111222.15
```

```
$ leer factura PK="22111333"
```

El sistema debería dar error, pues la factura 22111333 no existe!

```
$ cambiar factura PK="12345678"  
REG="1234567801/07/202512300500000.00"
```

La factura 12345678 tenía los siguientes datos:

Número: 12345678  
Fecha: 15/07/2025  
Cliente: 789  
Monto: 20111222.15

Ahora tiene los siguientes datos:

Número: 12345678  
Fecha: 01/07/2025  
Cliente: 123  
Monto: 500000.00

```
$ leer factura PK="12345678"
```

El sistema debería responder:



```
#reg # registro
0 1234567801/07/202512300500000.00
```

```
$ mostrar cliente
```

El sistema responderá:

```
#reg # registro
0 123Ana Mar
1 b 456Pepe
2 789Jose Ma
```

```
$ salir
```

El sistema permite salir y guarda en disco toda información que pudiese tener en memoria.