國外到貨單(FORARR)效能根治:移動/翻頁/修改欄位

模組:HBSTK 主檔:FORARR.PRG、DOMARR.PRG 日期:2026-08-03

一、訴求與現象

客戶站(網路 DBF)國外到貨單表身:[上下筆][翻頁]移動很慢、[修改]時欄位間移動卡頓(尤其「採購單號→零件號碼」「到貨量→外幣單價」兩處)。同樣邏輯的 900 Clipper 版(HyperDOS)卻很順,使用者合理質疑 GTWVT。

二、為何 Clipper 順、Harbour 頓(核心觀念)

層面	Clipper(HyperDOS/DOSBox)	Harbour(GTWVT+DBFCDX)
螢幕	寫模擬 VRAM(零成本),監控視窗固定頻率取樣=天然批次重繪	每個 @SAY 立即同步 WM_PAINT+CJK 字型掃描=同步重繪稅
資料	瀏覽期間不加鎖→SMB oplock 保住→讀取命中本地快取(記憶體速度)	每次移動做 CDX 讀鎖/header 驗證→byte-range lock 打破 oplock→每讀=真網路來回

兩者皆無法在多站共享 DBF 架構下複刻:索引不鎖會讀到分裂中的頁(錯位/GPF);強留 oplock 正是 Clipper 時代 CDX 壞檔的經典元兇(90 年代業界標準=關 oplock 保平安)。正解=等效補償:批次重繪+程式層快取+消滅重複 IO。

三、五輪修正

輪	修正	對象
R1	GetASAllInfo 同零件 2 秒快取(STATIC 檔首、Abs 防午夜);ShowBody/ShowOrderBody 整頁去除逐行 Parts→dbSeek(原 seek 只為決定顏色);ShowOrderLine 僅游標行才 seek(保留定位副作用+警示);預開 DomArrS/DomSchS(原每鍵開檔→掃→關檔)	上下筆 → 恢復正常
R2	ShowBody/ShowOrderBody 包 DispBegin/DispEnd(13 行一次 flush);預開 ForPurM	翻頁重繪、修改

輪	修正	對象
R3	KeyPageUpDown 的 GoSerialNo(單頭/尾折半 skip,大單=百次索引 skip)改:ShowBody 畫完 dbGoTo 回頁首+GoSerialBy 相對 skip(12);預開 ForPurS(GetDFPurData 每次開關檔=真元兇)	翻頁 hilite 慢半拍、採購單號→零件號碼
R4	CountArrQty/ReGetArrQty 由「order1 seek 單號+順序掃整張單」改 order4(cPartsId+cOrgOrdNo,三表同構)seek 直達只掃幾筆;VS06 兩趟連掃是「到貨量→外幣單價」最卡主因	修改兩個關鍵欄位跳轉 → 好了
R5	進單時 REPLACE 表頭 nSubCount WITH 0 洗掉行數快取→CountRecord 掃整張單;改寫回原值保留快取(空值自癒機制原樣)。移除進單時重複的 PartsMessage(ChkBotMsg 已含,連跑兩次)	進單顯示第一頁 → 通過

四、診斷心法

・使用者的精準觀察是最好的 profiler:「頁面已翻好,hilite 慢半拍」→ 直指 ShowBody 之後的 GoSerialNo;「A 欄到 B 欄卡」→ 直指該欄 VALID/下一欄 WHEN。

・客戶站 Debug log 的 BLOCK 時間戳可量測每鍵節奏(~240ms/鍵)並揪出隱藏角色(ShowFPurArrHelp 說明框每鍵重畫靜態文字 ~77ms,列入待辦)。

・「上下筆修好但翻頁沒好」證明各場景成本結構不同,要分開治,不能一招打天下。

五、踩坑

坑	說明
錨字串尾隨空格	RETURN cState 與 cSeconds 行尾各有一個空格,bytes 錨對不到;先 repr() 檢查再定錨
bash 單行多 export	含空格值的多變數單行 export 會壞(bcc 找不到);多行 export 才穩
grep 括號誤報	grep "FORARR.PRG(" 未轉義括號報 unmatched () 誤判 BUILD FAILED;以 exit code+exe 時間戳為準

坑	說明
行為忠實	CountArrQty 對 ForArrS 累加分支不中(既有行為回0),優化只換掃描路徑不改語意

六、修改檔案與後續

・source\FORARR.PRG、source\DOMARR.PRG(備份.bak_perf_/.bak_asinfocache_20260803_1317)
・待辦:ShowFPurArrHelp 每鍵重畫(~77ms/鍵)、PartsMessage移動中時間節流、瀏覽快照(進單載入 array)構想、letodb(Harbour DB伺服器)長期方向
・可推廣:同樣手法適用 DOMARR/DOMPUR/FORPUR/DOMSCH 等同構模組(Shape-A家族)

七、專章:DBFCDX Driver 深度解析 — Clipper 為何快、Harbour為何慢

兩邊都叫 DBFCDX,但 driver內部的「一致性模型」完全不同,這是速度差的核心。

7.1 一次 dbSkip(1) 在兩邊實際發生的事

DOS Clipper(Comix/SIx,經HyperDOS)	Harbour DBFCDX
查程式內索引頁快取 → 命中 → 直接取下一筆 recno 讀 DBF 記錄 → DOS INT21h → SMB redirector → oplock 快取命中 → 本機記憶體 → 全程 0 次網路來回(常態)	hb_cdxIndexLockRead():對 CDX 加 shared byte-range lock(網路來回1) 讀 CDX header 版本號與快取比對(來回2);版本變了→丟棄全部索引頁快取重讀 解鎖(來回3) 讀 DBF 記錄——該檔發生過 byte-range lock,SMB client 已永久停用 read cache → 真網路讀(來回4) → 每次 skip 至少 3~4 次網路來回(常態)

7.2 一致性哲學對照

面向	Clipper/Comix	Harbour DBFCDX
索引頁快取失效時機	只在「加鎖時」(RLock/FLock=同步點)才重讀驗證	每次索引移動都鎖住驗證版本
一致性模型	髒讀容忍:別站剛改的索引可能晚幾步才看到	嚴格:任何一步都保證看到最新索引
讀取期間鎖活動	零	每 skip/seek 一對 lock/unlock
SMB 快取(oplock)	無鎖活動 → 快取活著	鎖活動 → 快取被殺,連 DBF 讀取也變網路
風險	理論上有讀到 stale 索引的窗口(當年 CDX 壞檔傳說的溫床之一)	不會讀到 stale,但每步付網路稅

Clipper 不是「做得比較好」,是「賭得比較大」:它假設兩次加鎖之間索引舊一點沒關係。Harbour 作者選擇不賭——stale 索引頁在極端時序下會 seek 錯位甚至 GPF,Harbour 要在所有環境都正確。

7.3 對策可行性

選項	可行性與說明
叫 Harbour「別驗證」	無官方開關;hack 掉=引入 Clipper 的 stale 風險,卻沒有 Clipper 多年運轉的容錯運氣
獨占開啟(EXCLUSIVE)	Harbour 知道沒別人會改 → 完全跳過鎖與驗證、頁快取全開=本地級速度。共用主檔多站不能獨占——這正是暫存表/報表檔(如 DomOrdR1 用 EXCLUSIVE)飛快的原因
減少「移動次數」(本報告 R1~R5)	dbGoTo 取代長 skip、order4 直達、預開表、程式層快取——每省一次移動=省 3~4 個網路來回
瀏覽快照(B 路)	進單一次載入 array=之後零移動零網路稅;等效 Clipper 的快取,但一致性邊界明確(進單那一刻)
letodb(C 路,終極解)	同樣的 DBFCDX 驗證邏輯照跑,但跑在 S001 本地磁碟(微秒級),網路只剩一來一回 TCP 指令——「每步驗證」從毒藥變免費

7.4 一句話總結

Clipper 快在「不問就信快取」,Harbour
慢在「每步都去問一次」——而問一次在 SMB 上是 3~4 個來回。優化方向=
「少問」(減移動+程式層快取),長期解=「把被問的人搬到本地」(letodb)。