客戶／廠商電話查詢 MakeCSFTS 效能優化

專案：HBSTK　日期：2026-06-09　檔案：source/TOOLS.PRG

一、訴求與現象

「客戶基本資料」的電話查詢（以及備註查詢）由 MakeCSFTS() 負責，屬於客戶／廠商的全文檢索（contains 包含式搜尋）。零件版 MakeFTSArray() 早已透過 IndexOnFTS()（XINDEX + WHILE + FOR）把掃描下放到 RDD 層加速，但客戶／廠商版的 MakeCSFTS 仍停留在純 PRG 全表逐筆掃描，資料量一大就明顯變慢，因此本次針對它進行效能優化。

二、瓶頸分析（改善前）

瓶頸	說明	代價
cmxKeyVal() 重複計算	第 3175 行 cKeySource:=cmxKeyVal() 算完卻沒用，3177/3179 又各算一次。	每筆多算一次昂貴的 Comix 組合鍵重組
用組合索引鍵比對	cmxKeyVal() 需重新組出整個 compound 索引鍵字串再做 $ 包含比對。	每筆字串重組成本高
照索引序掃描 + cmxKeySkip	設 dbSetOrder(nNtxOrder) 後用 cmxKeySkip(1) 走索引序掃全表；但包含式搜尋無法靠索引提早結束，且結果最後一律 ASORT 照 cId 重排。	索引頁導航額外成本，且走索引序毫無意義

三、踩坑與陷阱

陷阱	內容
原註解錯誤（致命）	程式內 3176 行註解寫 “// cAlias => cTel1”，會讓人以為 order 3 只查 cTel1。實際從 900\set\SubCdx.DBF 讀出，Customer 第 3 索引鍵是三支電話：cTel1-'.'-cTel2-'.'-cActTel。若照舊註解只比 cTel1，會漏查 cTel2／cActTel。
索引鍵須先驗證	索引鍵定義不在原始碼，而在 MainCdx.DBF（找 cCdxName）＋ SubCdx.DBF（各 tag 的 cTagKey）。動手改比對式前先用 Python 讀 DBF 確認實際 KEY 值，才沒寫錯等價式。

陷阱	內容
BIG5 註解寫入	禁止用 Edit/Write 直接寫中文到 PRG（會混入 UTF-8 亂碼）。本次以 Python bytes-only、中文以 .encode("big5") 寫入 PRG。

四、解決方法

項目	改善前	改善後
掃描順序	dbSetOrder(nNtxOrder) 索引序	dbSetOrder(0) 自然記錄序（最快）
逐筆導航	cmxKeySkip(1) ×3	dbSkip(1) ×3
order 3 比對	cKey $ cmxKeyVal()	cKey $ (cTel1-'.'-cTel2-'.'-cActTel)
order 5 比對	cKey $ cmxKeyVal()	cKey $ (cNote-cNote2-cNote3)
key 計算次數	每筆 2 次	每筆 1 次（存入 cKeySource）

行為保證不變：找到的客戶／廠商集合相同、最終排序相同（照 cId）、4095 筆上限、刪除筆跳過、UI 完全不動。新比對式與 cmxKeyVal() 完全等價，因為索引鍵本身就是該欄位組合表達式。

五、使用技術與用法

技術	用法
DBF 索引鍵反查	Python 讀 MAINCDX.DBF 取得 Customer 的 cCdxName，再讀 SUBCDX.DBF 依序列出各 tag 的 cTagKey，第 N 個 tag 即 order N 的索引鍵。
Clipper 字串 - 運算子	A - B 會把 A 的尾端空白移到結果最後再接 B，等同索引鍵的 trim-join；故 cTel1-'.'-cTel2-'.'-cActTel 與 cmxKeyVal() 完全等價。
自然序掃描	因結果最後 ASORT 重排，掃描順序不影響結果，改 order 0 用實體記錄序最快。

技術	用法
Python bytes-only 改 BIG5 PRG	讀檔為 bytes、以函式區段切片定位、replace 取代，中文註解 .encode("big5")，不對既有內容做 decode，避免 UTF-8 污染。