dBASE IV Compiler 2.0

Produttore e distributore:
Borland Italia srl
Via Cassanese 224 - Palazzo Leonardo - 20090 Segrate (MI)
Prezzi *(IVA esclusa)*:
dBASE Compiler 2.0 Lit. 699.000
Upgrade dal compilatore, dal Run Time o da qualsiasi versione di dBASE Lit. 349.000
Nota: per i primi tre mesi chi acquista una nuova copia di dBASE IV riceverà il dBASE Compiler 2.0 in omaggio.

Borland
dBASE IV Compiler 2.0 per Dos

di Valter Di Dio

Non è passato nemmeno un anno dalla comparsa del compilatore «originale» per dBASE che ecco pronta la versione 2.0. Le novità sono sia sostanziali che estetiche. Non essendo nel frattempo cambiato il dBASE IV le novità del compilatore non si notano subito; anzi credo che, a parte la nuova interfaccia, pochi scoprirebbero da soli le differenze. Invece il fatto che il numero di versione sia passato direttamente da 1.0 a 2.0 dovrebbe immediatamente avvertire il lettore attento che non si tratta di un semplice ritocco; il compilatore è stato modificato completamente e il risultato si tocca con mano al primo uso.

L'installazione

La scatola contiene tre dischetti ad alta densità, un manuale d'uso (piccolissimo), una guida rapida di riferimento (comodissima) e una copia del Language Reference del dBASE IV (pesantissima). I tre dischetti contengono quasi cinque mega di materiale in forma compressa che diventano nove una volta terminata l'installazione. Durante la fase di installazione sul disco rigido ci devono essere almeno 14 mega di spazio libero. Due sole le domande cui si deve rispondere: in quale directory vogliamo che sia messo il compilatore (di default \BDCOMP) e quale tabella dei codici estesi vogliamo usare (di default quella del DOS installato).

Si deve fare attenzione solo a quest'ultima domanda; infatti i programmi scritti con la tabella 437 non funzionano su macchine con quella 850, e viceversa. Si può comunque istruire il compilatore affinché utilizzi una differente tabella al momento di generare il codice eseguibile. Il miglior sistema per ignorare questo problema resta comunque quello di non usare caratteri maggiori di 127 per i nomi delle variabili o delle procedure.

Terminata l'operazione di installazione conviene eseguire subito il programma DRIVERS che in pratica non è altro che un file ZIP autoestraente con i driver per 68 tra le principali stampanti.

Alla fine dell'installazione, tra gli altri, troverete i file DBC del compilatore, BDL del linker e BDS della nuova interfaccia oltre alle librerie per il modo a 16 bit e quelle per lavorare, sulle macchine dal 386sx in poi, a 32 bit.

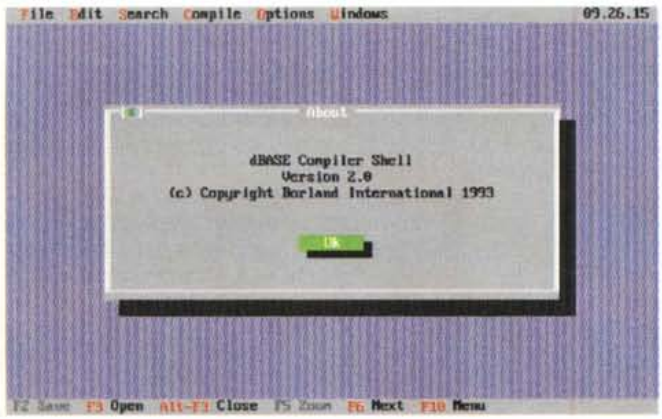
Principali novità

Sicuramente la novità principale, almeno dal punto di vista dell'utente, è il modo a 32 bit che consente sulle macchine 386 e successive di incrementare notevolmente la velocità di esecuzione. La Borland dichiara che in pratica si può avere anche un raddoppio delle prestazioni dato che, gestendo l'indirizzamento esteso, non è più necessario l'accesso alla memoria a segmenti. Naturalmente si deve essere certi che poi l'utente abbia una macchina con almeno un 386sx; il codice a 32 bit non funziona infatti sui 286 né sugli 8088/86.

Altra novità sicuramente ben vista è la riduzione del 50% dello spazio occupato dal programma eseguibile (.EXE); oltre al minor spazio occupato sul disco, una riduzione del codice significa anche una maggior velocità di caricamento del programma e un minor carico del traffico sulle reti.

Per il programmatore le novità più interessanti (quelle che rendono la vita più facile, per intendersi) sono la nuova interfaccia a menu e l'opzione di compilazione "Automatica" che permette di fare a meno della MAKE. In pratica, se si attiva questa possibilità, sia il compilatore che il linker cercano, a partire dal programma "Main" tutti i moduli richiamati e, se sono stati modificati li ricompilano, altrimenti viene linkato il vecchio object. Il programma MAKE (che faceva esattamente la stessa cosa) viene lasciato per quei casi in cui si debbano ricompilare dei moduli che non vengono esplicitamente chiamati dal programma.

Ancora per il programmatore è stata implementata la possibilità di creare dei moduli di libreria (.BDO) runtime; prima questo era possibile solo parzialmente e li si dovevano comunque linkare. Con i nuovi moduli .BDO molte funzionalità possono essere preparate una volta per tutte e poi distribuite insieme agli eseguibili; ad esempio si possono predisporre librerie per la gestione delle stampanti, della rete o per la presentazione dei dati. File di formati, etichette,



Ecco come si presenta la nuova finestra di lavoro del compilatore in modalità Shell.

Le opzioni di compilazione si attivano con dei semplici check-box. ▶

rapporti, schede e Query possono essere incluse in un'unica libreria BDO. Anche i file di preprocesso (DPP) vengono adesso compilati con estensione BDO.

È stato infine aggiunto il supporto per otto differenti formati di data tra cui, ovviamente, quello italiano.

La Shell

Forse il programmatore esperto non si dimostrerà eccezionalmente colpito dalla shell grafica, ma l'utente medio apprezzerà senz'altro la praticità di avere in un'unica finestra l'editor, il compilatore, il linker e il debugger (quest'ultimo solo se si possiede il dBASE IV). Sempre dalla shell è immediato ed intuitivo attivare gli switch di configurazione sia del compilatore che del linker ed altrettanto semplice è definire le directory di lavoro, i file da compilare e i nomi da assegnare agli eseguibili.

Tutto quello che deve fare l'utente medio per compilare un suo programma in dBASE IV o dBASE III è settare il bottone di AutoCompile, aprire il programma principale, o comunque il programma che poi chiamerà tutti gli altri, e poi scegliere "build all" dal menu "compile". In una nuova finestra compariranno le informazioni che il BDC sta emettendo durante la fase di compilazione, poi quelle del linker (BDL) ed eventualmente i messaggi di errore con la linea del codice che lo ha generato.

In caso di errore non si deve fare altro che aprire con l'editor una nuova finestra (peccato che non lo faccia da solo) e correggere il .PRG che ha creato problemi, salvarlo e rifare la compilazione. I comandi dell'editor sono stati scelti in modo da seguire quelli dell'Edit contenuto nel Dos 5 e quindi sono una via di mezzo tra quelli WordStar e quelli di Windows: i comandi di navigazione sono del WS, quelli di gestione dei blocchi usano invece le varie combinazioni shift-ins, shift-del, ecc. Usando la shell, in pochissimi minuti, si otterrà un codice eseguibile pronto per essere distribuito. Il linker crea di default un codice compresso; cioè che non contiene le librerie di runtime che vanno distribuite a parte; se volete invece che queste vengano linkate in un unico eseguibile basta andare nel menu Opzioni del Linker e settare lo switch "standalone".

La semplicità d'uso della shell è tale che è veramente difficile fare errori. Le uniche cose da verificare con un minimo di attenzione sono la configurazione delle opzioni sia del compilatore che del linker. Tutte le configurazioni scelte da dentro la shell vengono automaticamente salvate in un file apposito e non modificano quelle contenute nei file di configurazione del compilatore e del linker. Questo significa che potete tranquillamente avere due set di configurazioni, uno per quando lavorate con la shell e uno per quando lanciate la compilazione, o il link, direttamente dalla command line del Dos.

La configurazione

Un intero capitolo è dedicato ai file di configurazione (Config.db), ma per una trattazione migliore è preferibile rifarsi al manuale del dBASE IV. Il file di configurazione serve a definire una quantità incredibile di parametri prima dell'avvio di una procedura. Ad esempio vi si definiscono i colori da utilizzare, il formato della data, la presenza e la posizione dell'orologio e, importantissimo, le stampanti e i font.

Il file di configurazione permette di personalizzare un programma eseguibile senza bisogno di ricompilarlo. Se ad esempio un utente sostituisce una stampante ad aghi con una laser PostScript basta variare una sola riga del Config.db per aggiornare l'applicazione.

Tabella prestazioni
dBASE IV Compiler v. 2.0

- Crea 16 o 32 bit .EXE
- Compatibile al 100% con dBASE III PLUS e dBASE IV
- AutoCompile e AutoLink intelligenti
- Accesso a 4 Giga Byte di memoria
- Permette applicazioni single-user in ambiente multiuser
- Supporta vari livelli di password
- Gestisce archivi crittografati
- Gestisce le transazioni
- Non necessita di licenza né di una copia del dBASE

L'uso e la distribuzione

Come già accennato l'uso del compilatore è quasi immediato, a patto però che l'applicazione sia già correttamente funzionante sotto dBASE IV o DBIII. A mio avviso, comunque, non è consigliabile acquistare il solo compilatore visto che, oltre al debugger, ci sono moltissime cose che non si possono fare dalla Shell. Ad esempio il compilatore non gestisce i file .PRS di SQL che vanno prima convertiti in DBO da dentro al dBASE IV. Se comunque avete già delle applicazioni in dBASE e non pensate di svilupparne delle altre, il compilatore vi permetterà di trasformarle in veloci e sicuri programmi eseguibili. La praticità dell'eseguibile sta infatti, oltre che nella maggiore velocità e nella minore occupazione di spazio su disco, nel fatto che non essendo presente il sorgente è praticamente impossibile, per l'utente, la modifica, anche accidentale, dei programmi.

La distribuzione di un programma compilato è molto semplice; se è stato realizzato con il linker in modalità "standalone" basta distribuire l'eseguibile e, ovviamente, gli archivi, gli indici, le viste, ecc., se invece è stato linkato

"compact" si devono aggiungere le librerie di runtime: DBASE.RES e, a seconda del modo di compilazione utilizzato, DBASE16.RTL o DBASE32.RTL rispettivamente a 16 o a 32 bit. Nel caso il vostro codice utilizzi SQL le librerie da distribuire diventano DBASE16S.RTL e DBASE32S.RTL. La distribuzione di codice compact, invece di quello "standalone", permette di risparmiare spazio sul disco nel caso che le applicazioni in dBASE siano più di una; infatti i moduli "compact" possono condividere un'unica libreria, mentre quelli standalone contengono ciascuna la propria copia.

Le librerie sono liberamente distribuibili e quindi l'utente finale non deve acquistare nulla, nemmeno nel caso di uso in rete!

Le utility

Insieme al compilatore e alla MAKE vengono forniti altri quattro programmi.

I primi due, PMINFO e RMINFO, servono per testare le capacità del computer nella gestione del Protect Mode e del Real Mode. Insieme ai file Virtual.txt e Perform.txt aiutano a migliorare la configurazione del proprio computer in modo da avere a disposizione il massimo delle prestazioni che l'hardware può supportare.

Le altre due, dSPLIT e dJOIN servono, rispettivamente, a spezzare un file eseguibile in segmenti della dimensione desiderata e, successivamente, a ricomporli. Sono indispensabili per trasportare eseguibili di grosse dimensioni senza dover ricorrere allo scomodo Backup/Restore del MS-Dos.

Da Vulcan a dBASE IV

Era il lontano 1981; lontano relativamente, certo: alcuni dei nostri lettori all'epoca non avevano ancora imparato a leggere e scrivere, per altri i computer erano quelle cose sferraglianti e piene di luci ammiccanti che nei film di fantascienza sputavano le risposte su un nastro di carta, e questa stessa rivista vedeva la luce solo a settembre. Di sicuro, per l'informatica personale si trattava quasi degli albori: i PC IBM sarebbero arrivati l'anno successivo, il Macintosh doveva aspettare ancora tre anni. I sistemi operativi erano tutti a 8 bit e si chiamavano UCSD, HDOS, COSMOS, FLEX, CP/M; molti computer addirittura non li avevano proprio, come ad esempio l'Apple II e il Pet Commodore che utilizzavano un Basic esteso per sopperire alla loro mancanza.

Eppure nella primavera di quel lontano 1981 usciva la prima pubblicità del dBASE II (come molti sapranno non è mai esistito un dBASE I, il nome fu scelto per dare l'impressione che il prodotto fosse già maturo). Girava, all'inizio, sotto CP/M e solo l'anno successivo fu portato sotto il nuovo MS-Dos per divenire subito il principale standard nel formato dei database.

Ma ripartiamo un attimo da diversi anni prima e da un nome: l'anno era il 1975 e il nome Wayne C. Ratliff.

Wayne lavorava per la Martin Marietta Corp. di Pasadena, California, e il nome di questa città dovrebbe immediatamente far pensare al più prolifico e interessante laboratorio di tutta la storia americana: il Jet Propulsion Laboratory. Per conto della Martin Marietta, Wayne stava sviluppando un database di supporto per le operazioni a terra della sonda Viking (che avrebbe raggiunto Marte nel 1976). Il programma si chiamava MFILE e, per quanto fosse in sostanza un database, non aveva in realtà nessuno degli attributi che oggi chiediamo a questi prodotti.

La stagione del football stava per iniziare e, come accade spesso in molti posti di lavoro, Wayne ed alcuni ingegneri del JPL avevano messo su un gruppo per fare un po' di scommesse. Wayne non era molto addentro al gioco del football e così si mise ad elaborare delle statistiche sulle stagioni precedenti e sulle partite in corso. Dopo poche settimane il pavimento di casa sua era completamente coperto dai giornali del lunedì, al punto che era impossibile andare in giro e ancor meno capirci qualcosa di tutti quei numeri. Essendo un programmatore decise che questo era lavoro per un computer.

Non avendo idea di quali fossero i dati significativi, Wayne decise che il programma avrebbe dovuto essere molto flessibile in modo da lasciare libero il campo alla sperimentazione. Nel 1975 sviluppare un programma significava perforare una gran quantità di schede, poi darle in pasto all'elaboratore, attendere che questo lo compilasse, lo linkasse e infine che lo eseguisse. Il lavoro del computer durava in genere pochi secondi, ma l'attesa del proprio turno poteva anche durare ore se non giorni. Per questa strada sarebbero passate diverse stagioni prima che il programma fosse pronto. Dimenticato quindi il football, affascinato dal film 2001 e, naturalmente, da HAL 9000, Wayne si dedicò per alcuni mesi ad un programma che fosse in grado di comprendere il linguaggio corrente. Spese in questo tempo e denaro acquistando libri sul linguaggio, sull'intelligenza artificiale, sul calcolo dei predicati e sulla gestione dei database. Alla prima occasione sfruttò le conoscenze acquisite per scrivere un programma che, grazie all'algebra di G. Spencer Brown, era in grado di risolvere calcoli proposizionali operando sui dati meteorologici mandati a terra dalla sonda Viking, che nel frattempo era atterrata su Marte. In quell'occasione Wayne scoprì gli indici B-tree e alcune tecniche di ricerca binaria.

Era ormai il 1976 e un amico mostrò a Wayne il suo nuovo «giocattolo»: un microcomputer IMSAI 8080. Wayne rimase letteralmente impressionato dalla potenza di calcolo di quel piccolo computer. Comprò immediatamente un kit e passò il mese di novembre col saldatore in mano. Ci volle un anno prima che il computer fosse completamente funzionale; otto mesi solo per farsi spedire un floppy disk e altri mesi ancora per avere un sistema operativo e un editor che funzionasse col resto della macchina. A quel punto si rese conto che i microcomputer erano del tutto simili ai loro cugini più grandi: certo erano limitati, ma ancora per poco.

Durante i periodi di tempo in cui Wayne non lavorava al computer stava ancora rimuginando al suo progetto di un database; l'idea era di qualcosa di semplice come MFILE, che fosse relazionale come suggerivano alcuni articoli di David Kroenke e naturalmente che girasse nei 48K del suo computer. Al JPL avevano un database che rispondeva alle prime due ipotesi, il JPLDIS; si trattava di un database semirelazionale con comandi in un linguaggio simile all'inglese. Wayne decise che si sarebbe ispirato al JPLDIS per la sua creatura.

Nel gennaio 1987 cominciò a scrivere un interprete semplificato dei set di comandi del JPLDIS usando direttamente il linguaggio assembler dell'8080. Il Basic lo attirava per la sua semplicità e potenza, ma non sarebbe mai riuscito ad infilarlo in soli 48K (molti dei quali oltre tutto occupati dal sistema operativo). Il sistema operativo era il PT-DOS, visto che il CP/M non era ancora emerso.

Il 29 gennaio 1978, all'incirca a mezzanotte, nacque il formato DBF: i primi comandi che seguirono furono CREATE, DISPLAY, APPEND ed EDIT. Nel giro di un mese il programma aveva abbastanza primitive da essere usabile: il nome era Vulcan! Nel mese seguente a Vulcan vennero aggiunte altre capacità derivate da JPLDIS, in particolare l'aritmetica BCD e il calcolo in virgola mobile.

Poi il disastro: il floppy disk si ruppe e ci vollero tre mesi per averlo indietro riparato. Nell'attesa del ritorno del floppy drive Wayne fece molti piani. Nell'aprile del 1978 mise a punto uno schema denominato "total parsing", che però non venne mai implementato in Vulcan; David Fulton fece esattamente una cosa simile quando aggiunse la precompilazione per aumentare le prestazioni del suo FoxBase.

Al ritorno del floppy drive Wayne decide di allontanarsi dal modus operandi del JPLDIS perché questo non permetteva di modificare le variabili una volta assegnate e non possedeva i comandi per eseguire dei cicli (ad esempio DO WHILE); persino le IF..THEN..ELSE erano limitate ad un solo comando.

Così prende a prestito comandi un po' da tutti i linguaggi del momento: INPUT e "?" dal Basic, DO WHILE dal Fortran, ACCEPT dal Cobol e altri simili. Fu sorprendente la differenza che questi nuovi comandi impressero Vulcan. Divenne subito una cosa viva, chi lo usava poteva imprimere al linguaggio la propria volontà: poteva scrivere programmi dentro Vulcan! JPLDIS non aveva mai avuto niente di simile, Vulcan era proprio quello che Wayne desiderava.

Il disastro colpì ancora nella primavera del '79 quando il floppy si ruppe nuovamente. Dopo i soliti tre mesi per la riparazione Wayne invitò a casa sua Jeb Long, un program-

Conclusioni

Come scrissi nella prova del dBASE IV (MCmicrocomputer n. 131) il punto di forza del dBASE sta nella gran mole di applicativi esistenti e nella possibilità di compilare lo stesso identico programma che sta girando in modo interpretato. Non dover cambiare nemmeno una riga nel passaggio tra interprete e programma eseguibile autonomo significa poter fare un debug sul campo utilizzando tutta la potenza, la flessibilità e la trasparenza dell'interprete e poi, una volta eliminati i bug, compilare un'applicazione "standalone" e consegnarla al cliente, o tenerla per sé sfruttando la maggior velocità dell'eseguibile.

Il fatto che il compilatore sia della Borland significa da un canto la garanzia di esperienza e capacità dimostrata con compilatori ben più potenti e complessi, dall'altro la sicurezza che qualsiasi futura implementazione del dBASE non vanificherà il lavoro finora svolto.

L'offerta del compilatore in omaggio a chi acquista una nuova copia di dBASE IV nei primi tre mesi e lo sconto per chi già possiede una qualsiasi versione di dBASE sta a dimostrare l'interesse della Borland su questo prodotto; interesse che mi è stato direttamente confermato dagli stessi responsabili dello sviluppo del dBASE.

Il prezzo del compilatore, soprattutto se si tiene conto che è possibile distribuire tutte le applicazioni che si vuole senza alcuna royalty, è assolutamente giustificato. MC

matore del JPL, per fargli vedere Vulcan; questi fu immediatamente colpito dalle possibilità commerciali di un simile prodotto soprattutto dentro al JPL stesso.

Ma il PTDOS stava lasciando il posto al più giovane CP/M (il predecessore dell'MS-Dos) così Wayne, per garantirsi un mercato più ampio, dovette convertire Vulcan al nuovo sistema operativo. Più tardi, sempre nel 1979, il JPL acquistò da Wayne una licenza multipla per l'uso di Vulcan: fu il primo cliente. Il JPL usò Vulcan intensamente e proprio dagli ingegneri del JPL arrivavano continuamente suggerimenti e avvisi di bug. Proprio su richiesta del JPL Vulcan fu modificato per renderlo più simile al JPLDIS che loro sapevano già utilizzare.

Una sera Stan Brin,un utente assiduo, raccontò a Wayne del nuovo programma chiamato DataStar della MicroPro. Si trattava di un data manager a tutto schermo, all'opposto di Vulcan che era a riga di comando. Vulcan era divertente, ma non reggeva al confronto con un prodotto della MicroPro, gli stessi che avevano creato e stavano vendendo il mitico WordStar.

Pochi giorni dopo Wayne era al lavoro per aggiungere dei comandi a tutto schermo. Vennero modificati l'EDIT, l'APPEND e l'INSERT in modo da utilizzare una maschera video chiamata FMT. In seguito all'FMT venne aggiunto il comando "@". Nello stesso periodo Wayne decise di implementare gli indici e visto che aveva qualche esperienza con i B-tree incluse un metodo di indicizzazione trasparente basato proprio su di essi.

Le vendite di Vulcan cominciavano ad andare bene. In nove mesi Wayne vendette 61 copie del programma. Certo se oggi uno dei grossi produttori vendesse solo 60 pacchetti anche in una sola ora avrebbe seri problemi commerciali; a quel tempo invece era un risultato discreto. Senonché, tra il matrimonio e il lavoro a tempo pieno al JPL, le ore dedicate a Vulcan stavano diventando troppo poche. Nell'estate del 1980 si doveva prendere una decisione: o lasciar perdere Vulcan o metter su un sistema commerciale con tanto di ufficio vendite e marketing.

Nel frattempo la Harris Computer di Ft. Lauderdale (Florida) rivendicò la paternità del nome Vulcan che loro stavano usando per un sistema operativo. Anche se Wayne credeva che in tribunale l'avrebbe spuntata decise che cambiare nome: era preferibile ad una causa da 100.000 dollari.

Dopo un po' di telefonate con dei tizi che si chiamavano Hal Pawluk e Hal Lashlee si decise un incontro al quale partecipò, oltre a Wayne ed ai due, un certo George Tate. Questi videro un demo di Vulcan e dopo qualche giorno decisero di prendere i diritti di distribuzione per un anno. Seduti nell'ufficio di George (il suo garage), Wayne e gli Hal cominciarono a pensare al nome del prodotto, al suo lancio pubblicitario e al nome della società. Hal Lashlee, un genio della pubblicità di Beverly Hills, creò tutto in un sol colpo: *"dBASE II vs. the Bilge Pump" — by Ashton-Tate!* Dal momento che si cambiava nome si decise anche di aggiungere qualche cosa al prodotto. Venne migliorata la gestione degli overlay, venne aggiunto il comando *"@"* e furono sostituiti i B-tree con i più efficienti B*-tree.

Al JPL Wayne stava intanto insegnando ad usare Vulcan ad una studentessa: Carolyn. In realtà, senza che lo sapesse lei stava già usando dBASE II. Nel frattempo George stava già andando in giro a presentare dBASE II a varie ditte e società. E fece il miracolo. Moltissimi infatti scoprirono per la prima volta le potenzialità di dBASE II.

Le vendite ufficiali iniziarono nel febbraio del 1981 e così iniziò l'era d'oro (in ogni senso) dall'azienda. George, Hal e Wayne continuarono per tutto l'anno a presentare il prodotto e a metter su la struttura commerciale della Ashton-Tate. Verso la fine del 1981 Wayne da solo non bastava più per lo sviluppo del prodotto e altri programmatori si aggiunsero a lui; il primo fu Jeb Long che modificò la gestione dello schermo per supportare i 52 caratteri per riga dell'Osborne, ed in seguito convertì il codice da 8080 a 8086 per potersi introdurre nel mondo dei PC IBM. A lui seguirono Jim Rowe, Scott Mesch, John Gainsborough e Perry Lawrence che lavorarono soprattutto alla nuova versione in C.

Verso la metà del 1982 Wayne lasciò la Martin Marietta per dedicarsi a tempo pieno al dBASE (dove per tempo pieno si deve intendere 16 ore al giorno per sette giorni alla settimana). Il risultato fu la versione 2.5 e molte idee per il dBASE III.

Nei primi mesi del 1983 Wayne scopre che la Ashton-Tate gli nascondeva parte dei diritti che gli spettavano e che metteva il copyright A-T a molte cose che in realtà erano sue. Per farla breve, la Ashton-Tate si accordò con Wayne per lo sviluppo di dBASE sotto Unix per conto della AT&T, ma questa volta avrebbero acquistato da lui tutti i diritti. Wayne ottenne comunque di entrare nella società con il titolo di vice presidente.

Più tardi, sempre nel 1983, la Microrim iniziò una campagna pubblicitaria puntata su un prodotto che avrebbe fatto scomparire dBASE, si chiamava R:BASE. La Ashton-Tate era in difficoltà e Wayne propose un completo restyling del prodotto e iniziò lo sviluppo del dBASE III; era il 14 giugno del 1984. Il team di sviluppo composto da sei programmatori (Jordan Brown, Alistair Dallas, John Gieselman, Jeb Long, Chip Morton e Jim Rowe) aveva lavorato 16 ore al giorno per quattro mesi consecutivi; uno sforzo incredibile, se si considera anche che il dBASE III era relativamente esente da bug.

Mentre il team continuava lo sviluppo del successivo dBASE III PLUS, Wayne tornò al suo passatempo preferito: la comprensione del linguaggio naturale. Ma non senza gettare le basi di quello che gli sarebbe piaciuto fosse il dBASE IV, nel quale avrebbe voluto inserire i risultati dei suoi studi. Ma George Tate morì nell'agosto del 1984 e Wayne non si trovò mai più bene con la nuova direzione, tanto che nel 1986, dopo un anno e mezzo di inutili lavori se ne andò via.

La Ashton-Tate non si riprese mai completamente dalla morte del suo fondatore e nel 1989 il dBASE IV arrivò sul mercato non solo in ritardo ma, cosa ancora più grave, pieno di bug, tanto da costringere la società ad annunciare per il 1990 una nuova versione priva di errori.

La storia finisce, o meglio ricomincia, nel 1991 quando la Borland si fonde con quel che rimaneva della Ashton-Tate (ma forse è più corretto dire che la assorbì). L'arrivo della Borland ha dato un nuovo impulso allo sviluppo e immesso nuova linfa nel progetto dBASE dapprima con la versione 2.0 del dBASE IV, poi nel 1993 con il compilatore e oggi, primavera del 1994, con la stupefacente versione per Windows.

Quest'ultima, dal prossimo rilascio, introduce gradualmente i programmatori dBASE al mondo degli oggetti e della programmazione visuale, pur consentendo una completa compatibilità all'indietro del codice e del modo di lavorare tradizionale. La completa unificazione a basso livello con Paradox (dBASE e Paradox possono infatti liberamente leggere l'uno i file dell'altro) e l'integrazione di data base esterni via SQL rendono il moderno dBASE uno strumento potentissimo di lavoro soprattutto in ambiente client-server, che sarà lo scenario aziendale tipico dei prossimi anni. In quest'ottica dBASE consente di mantenere l'investimento preesistente in termini di programmi e programmatori, consentendo ad entrambi una facile migrazione al nuovo mondo della programmazione ad oggetti sotto Windows. MC