

RC-26003

Boom Lift 45 ft	12 d
Generator 20 kW	7 d
Charges	$3,300.00
Deposit	−$1,050.00
Due	**$2,250.00**

```
PRIVATE pnContrId
pnContrId = 3
lnResult = SQLEXEC(goConn.nHandle, ;
    "EXEC dbo.usp_GetContract ?pnContrId", "curContract")
```

line_no	asset_tag	descr	day_rate	week_rate
1	BR-LF-02	Boom Lift 45 ft	340.0000	1360.0000
2	BR-GN-01	Generator 20 kW	145.0000	580.0000

ฉบับภาษาไทย · CLIENT-SERVER

Visual FoxPro และ SQL Server

พัฒนาระบบ Client-Server ด้วย SQL Pass-Through, Remote View และ CursorAdapter

ODBC Driver · Connection · Parameter · ชนิดข้อมูล
Transaction · Stored Procedure · rowversion · การย้ายระบบ
ประสิทธิภาพ · ความปลอดภัย · การติดตั้งใช้งาน · โปรเจกต์สรุป

SQLEXEC · REMOTE VIEWS · CURSORADAPTER · T-SQL

THENEXBOOKS

Visual FoxPro และ SQL Server

การพัฒนาระบบ Client-Server ด้วย SQL Pass-Through, Remote View และ CursorAdapter
ย้ายข้อมูลของแอปพลิเคชัน VFP 9 จากตาราง DBF ไปสู่ SQL Server โดยยังคงหน้าจอเดิมที่คุณไว้วางใจ

ฉบับภาษาไทย · พิมพ์ครั้งที่ 1

THENEXBOOKS

Visual FoxPro และ SQL Server

การพัฒนาระบบ Client-Server ด้วย SQL Pass-Through, Remote View และ CursorAdapter

ฉบับภาษาไทย พิมพ์ครั้งที่ 1 ปี 2026

ฉบับภาษาไทยแปลและเรียบเรียงจากต้นฉบับภาษาอังกฤษของ thenexbooks

ลิขสิทธิ์ © 2026 thenexbooks (thenexbooks.gumroad.com) สงวนลิขสิทธิ์

ห้ามทำซ้ำ เผยแพร่ หรือส่งต่อส่วนหนึ่งส่วนใดของหนังสือเล่มนี้ ไม่ว่าในรูปแบบใดหรือด้วยวิธีใด โดยไม่ได้รับอนุญาตเป็นลายลักษณ์อักษรจากผู้จัดพิมพ์ ยกเว้นการอ้างอิงข้อความสั้น ๆ ในบทวิจารณ์ และการใช้งานที่ไม่ใช่เชิงพาณิชย์อื่น ๆ ที่กฎหมายลิขสิทธิ์อนุญาต ตัวอย่างโค้ด สคริปต์ T-SQL และโปรแกรม Visual FoxPro ในหนังสือ คุณนำไปใช้ในโปรเจกต์ของคุณเองได้อย่างอิสระ

ข้อมูลในหนังสือเล่มนี้ให้ไว้ "ตามสภาพ" โดยไม่มีการรับประกันใด ๆ แม้เราจะจัดทำด้วยความระมัดระวังทุกขั้นตอน ผู้จัดพิมพ์ไม่รับผิดชอบต่อข้อผิดพลาดหรือการตกหล่น หรือความเสียหายที่เกิดจากการนำข้อมูลในหนังสือไปใช้ Visual FoxPro 9.0 SP2 เป็นรุ่นสุดท้ายของผลิตภัณฑ์ และการสนับสนุนแบบขยายเวลา (extended support) ของ Microsoft สิ้นสุดลงในเดือนมกราคม 2015 ODBC driver, OLE DB provider, รุ่นของ SQL Server และการตั้งค่าความปลอดภัยของ Windows เปลี่ยนแปลงไปตามเวลา พฤติกรรมจึงอาจแตกต่างกันตามรุ่นและการตั้งค่า ควรอ่านเอกสารของ Microsoft สำหรับรุ่นที่คุณใช้เสมอ ทดสอบกับสำเนาข้อมูล และสำรองฐานข้อมูลก่อนรันตัวอย่างที่เปลี่ยนแปลงข้อมูลหรือ schema

Microsoft, Visual FoxPro, SQL Server, Windows, Azure, Excel และ Visual Studio เป็นเครื่องหมายการค้าของกลุ่มบริษัท Microsoft ชื่อผลิตภัณฑ์อื่น ๆ เป็นเครื่องหมายการค้าของเจ้าของแต่ละราย การใช้ชื่อเหล่านี้ในหนังสือมีไว้เพื่อระบุถึงผลิตภัณฑ์เท่านั้น และไม่ได้หมายความว่าได้รับการรับรอง หนังสือเล่มนี้เป็นผลงานอิสระ ไม่มีส่วนเกี่ยวข้อง ไม่ได้รับการสนับสนุน และไม่ได้รับการรับรองจาก Microsoft Corporation

Brackenridge Equipment Rentals แอปพลิเคชัน RentDesk ของบริษัท รวมถึงสาขา บุคคล ลูกค้า อุปกรณ์ สัญญาเช่า ใบแจ้งหนี้ และข้อมูลตัวอย่างทั้งหมดในหนังสือเล่มนี้เป็นเรื่องสมมติ หากคล้ายคลึงกับบริษัทหรือบุคคลจริงก็เป็นเพียงความบังเอิญ ที่อยู่อีเมลใช้โดเมนที่สงวนไว้สำหรับตัวอย่าง และหมายเลขโทรศัพท์ใช้รูปแบบ 555

คำนำ

แอปพลิเคชัน Visual FoxPro จำนวนมากยังคงทำงานได้ตรงตามที่ถูกสร้างขึ้นมาทุกประการ ทั้งรับคำสั่งซื้อ ติดตามสต็อก พิมพ์ใบแจ้งหนี้ และขับเคลื่อนธุรกิจไปวันต่อวัน ฟอร์มทำงานรวดเร็ว รายงานเชื่อถือได้ และผู้ใช้ก็รู้จักทุกคีย์ลัด สิ่งที่มักไม่เหมาะอีกต่อไปคือที่เก็บข้อมูล ตาราง DBF ในโฟลเดอร์แชร์เหมาะอย่างยิ่งกับสำนักงานเดียวที่ใช้เครือข่ายภายใน แต่ไม่เหมาะกับหลายสาขาที่เชื่อมต่อกันผ่าน VPN กับตารางที่โตขึ้นเรื่อย ๆ กับผู้ตรวจสอบที่ถามว่าใครเห็นข้อมูลอะไรได้บ้าง และกับระบบอื่นที่ต้องการอ่านข้อมูลชุดเดียวกัน

SQL Server แก้ปัญหาเหล่านี้ได้ และ Visual FoxPro ก็ถูกออกแบบมาให้คุยกับมันได้ VFP 9 มีวิธีทำงานกับ database server ที่สมบูรณ์แล้วถึงสามแบบ ได้แก่ SQL pass-through, remote view และคลาส CursorAdapter ทุกแบบคืนผลลัพธ์เป็น cursor ธรรมดาของ VFP ฟอร์ม grid และรายงานของคุณจึงยังทำงานกับข้อมูลที่คุ้นเคยได้เหมือนเดิม ส่วนที่ยากไม่ใช่ไวยากรณ์ แต่คือการรู้ว่าควรใช้ driver ตัวไหนและเพราะอะไร ชนิดข้อมูลเปลี่ยนไปอย่างไรระหว่างทาง จะทำให้การอัปเดตปลอดภัยได้อย่างไรเมื่อหลายคนแก้ไข record เดียวกัน จะย้ายข้อมูล DBF ที่สะสมมาหลายปีโดยไม่ให้ตกหล่น แม้แต่ record เดียวได้อย่างไร และจะ deploy ผลลัพธ์ไปยังเครื่องของผู้ใช้ทุกคนได้อย่างไร

หนังสือเล่มนี้เป็นคู่มือภาคปฏิบัติที่ผ่านการตรวจสอบแล้วสำหรับการเดินทางครั้งนี้ เราคงหน้าจอ VFP ไว้และย้ายเฉพาะข้อมูล ทุกเทคนิคถูกนำไปใช้กับแอปพลิเคชันที่สมจริงเพียงตัวเดียว และโค้ดในหนังสือถูกรันจริงกับ SQL Server instance จริงจาก Visual FoxPro 9 ของจริง พร้อมแสดงผลลัพธ์ที่ได้จริง

หนังสือเล่มนี้เหมาะกับใคร

- **นักพัฒนา Visual FoxPro** ที่ดูแลแอปพลิเคชันที่ใช้ DBF และต้องย้ายข้อมูลไปยัง SQL Server โดยไม่ต้องเขียนระบบใหม่ทั้งหมด
- **ทีมที่วางแผนปรับปรุงระบบให้ทันสมัยแบบค่อยเป็นค่อยไป** ที่ต้องการนำฐานข้อมูลขึ้นเซิร์ฟเวอร์ตั้งแต่ตอนนี้ เพื่อให้หน้าจอแบบเว็บหรือ .NET ในอนาคตใช้ข้อมูลร่วมกันได้
- **DBA ของ SQL Server และนักพัฒนา .NET** ที่ต้องดูแล client ที่เป็น VFP และต้องการเข้าใจว่ามันส่งอะไรไปยังเซิร์ฟเวอร์และเพราะอะไร
- **ที่ปรึกษา** ที่รับช่วงโค้ด VFP แบบ client-server ที่สร้างด้วย remote view หรือ CursorAdapter และต้องแก้ไขเพิ่มความปลอดภัย หรือทำให้เร็วขึ้น

คุณควรรู้จักภาษา Visual FoxPro, Command window, ตาราง DBF และ SELECT-SQL มาก่อน แต่ไม่จำเป็นต้องมีประสบการณ์กับ SQL Server, ODBC หรือ T-SQL เพราะเราจะแนะนำแนวคิดฝั่งเซิร์ฟเวอร์แต่ละเรื่องจากมุมมองของ VFP เมื่อถึงคราวที่ต้องใช้

สิ่งที่คุณจะได้เรียนรู้

- ODBC driver ทำงานกับ VFP แบบ 32 บิตอย่างไร จะเลือก driver อย่างไร และจะเขียน connection string แบบไม่ใช้ DSN อย่างไร
- วิธีเชื่อมต่อ ตั้งค่า และใช้ connection ร่วมกันด้วย SQLSTRINGCONNECT, SQLCONNECT และ SQLSETPROP
- ชนิดข้อมูลของ SQL Server จับคู่กับชนิดข้อมูลของ VFP อย่างไร ทั้ง Currency, วันที่, Memo, Logical, NULL และ Unicode รวมถึงกับดักที่ต้องระวัง
- วิธีใช้ SQL pass-through อย่างปลอดภัย ทั้ง parameter, result set หลายชุด, prepared statement, การเรียกแบบ asynchronous และการจัดการข้อผิดพลาดอย่างรัดกุมด้วย AERROR
- วิธีสร้าง updatable cursor, remote view และคลาส CursorAdapter และการตรวจจับ update conflict
- วิธีใช้ transaction, stored procedure ที่มี output parameter, ค่า identity, sequence, การจัดการการทำงานพร้อมกันด้วย rowversion และ constraint ฝั่งเซิร์ฟเวอร์
- วิธีย้ายข้อมูล DBF ไปยัง SQL Server ด้วยสคริปต์ที่รันซ้ำได้ และพิสูจน์ว่าไม่มีข้อมูลสูญหาย
- วิธีทำให้ VFP แบบ client-server เร็ว ปลอดภัย deploy ได้ และวินิจฉัยปัญหาได้ รวมถึงการแปลงโมดูลทั้งโมดูล

โครงสร้างของหนังสือ

ภาคที่ 1 — จาก DBF สู่ Client-Server (บทที่ 1–5) อธิบายว่าอะไรเปลี่ยนไปเมื่อข้อมูลย้ายไปอยู่บนเซิร์ฟเวอร์ ติดตั้ง SQL Server และฐานข้อมูลตัวอย่าง และครอบคลุม ODBC driver, connection และการจับคู่ชนิดข้อมูล **ภาคที่ 2 — SQL Pass-Through** (บทที่ 6–10) สอน SQLEXEC อย่างละเอียด ทั้ง parameter, โค้ดที่ปลอดภัยจาก injection, result set หลายชุด, การรันแบบ prepared และ asynchronous, การจัดการข้อผิดพลาด และการแก้ไขข้อมูล **ภาคที่ 3 — Remote View และ CursorAdapter** (บทที่ 11–15) ครอบคลุม remote view ใน database container, updatable view และ WhereType, คลาส CursorAdapter กับ ODBC และ ADO รวมถึงการเลือกเทคนิคและผูกข้อมูลเข้ากับฟอร์ม **ภาคที่ 4 — ตรรกะฝั่งเซิร์ฟเวอร์และความถูกต้องของข้อมูล** (บทที่ 16–20) ครอบคลุม transaction, stored procedure, ค่า identity และ sequence, การทำงานพร้อมกันด้วย rowversion และการย้ายกฎไปไว้ที่เซิร์ฟเวอร์ **ภาคที่ 5 — การย้ายระบบ ประสิทธิภาพ และความปลอดภัย** (บทที่ 21–24) วางแผนและลงมือย้ายข้อมูล จากนั้นทำให้ระบบที่ได้เร็วและปลอดภัย **ภาคที่ 6 — การติดตั้งใช้งานและโปรเจกต์สรุป** (บทที่ 25–28) ครอบคลุมการ deploy รายงาน การแก้ปัญหา และการแปลงโมดูลทั้งโมดูล ส่วนภาคผนวกมีสรุปย่อ (cheat sheet) เทมเพลตและตารางอ้างอิง รายการตรวจสอบก่อนใช้งานจริง และอภิธานศัพท์

ทุกบทปิดท้ายด้วยหัวข้อ **ลองทำด้วยตัวเอง** และกล่อง **สรุปประเด็นสำคัญ** สำหรับทบทวนอย่างรวดเร็ว

ตัวอย่างหลักของหนังสือ: Brackenridge Equipment Rentals

ตลอดทั้งเล่ม เราจะทำงานกับ **RentDesk** ระบบงานเช่าของ **Brackenridge Equipment Rentals** บริษัทให้เช่าอุปกรณ์สมมติที่ให้เช่ารถขุด รถตัก รถกระเช้า เครื่องบดอัด และเครื่องปั่นไฟ แก่ผู้รับเหมาก่อสร้าง ผู้รับจัดสวน หน่วยงานโยธาของเมือง และเจ้าของบ้าน จากลานเก็บอุปกรณ์สามแห่ง RentDesk เขียนด้วย Visual FoxPro บนตาราง DBF อิสระ (free table) ได้แก่ สาขา ลูกค้า อุปกรณ์ สัญญาเช่า รายการในสัญญา การคืนอุปกรณ์ และใบแจ้งหนี้ เมื่อสามสาขาต้องใช้ข้อมูลร่วมกันผ่าน VPN บริษัทจึงย้ายตารางไปยัง SQL Server โดยยังคงแอปพลิเคชัน VFP ไว้ เรากำหนด schema ปลายทาง ย้ายข้อมูลด้วยสคริปต์ที่ตรวจสอบแล้ว และสร้างส่วนเข้าถึงข้อมูลขึ้นใหม่ทีละชั้น ตั้งแต่การเรียก SQLEXEC ครั้งแรกไปจนถึง stored procedure ที่เปิดสัญญา รับคืนอุปกรณ์ และออกใบแจ้งหนี้ พร้อมคำนวณจำนวนวันเช่าและภาษีได้อย่างแม่นยำ ข้อมูลตัวอย่างมีขนาดเล็กพอที่จะตรวจด้วยมือได้ ทุกตัวเลขในหนังสือจึงตรวจสอบได้

รุ่นของซอฟต์แวร์และข้อตกลงในการนำเสนอ

ตัวอย่างในหนังสือมุ่งใช้กับ **Visual FoxPro 9.0 SP2** และ **SQL Server 2022** (รุ่น Express ที่ใช้ได้ฟรีก็เพียงพอ) และเกือบทั้งหมดใช้ได้กับ SQL Server รุ่นใหม่ ๆ ด้วย VFP เป็นโปรแกรม 32 บิต จึงใช้ ODBC driver แบบ 32 บิต driver หลักของเราคือ **ODBC Driver 17 for SQL Server** และเราจะชี้ให้เห็นจุดที่ driver รุ่นเก่าซึ่งมาพร้อม Windows ทำงานต่างออกไป ในจุดที่พฤติกรรมขึ้นกับ driver หรือรุ่น จะมีกล่องหมายเหตุบอกไว้ และหนังสือจะอธิบายสิ่งที่เราพบจากการทดสอบ ผลลัพธ์แสดงด้วยการตั้งค่าแบบสหรัฐฯ ได้แก่ `SET CENTURY ON` วันที่ในรูปแบบ `03/21/2026` และสกุลเงินในรูปแบบ `$1,876.88`

> **เคล็ดลับ**
> แนวปฏิบัติที่แนะนำ หรือเทคนิคที่ช่วยให้งานของคุณง่ายขึ้น

> **หมายเหตุ**
> ข้อมูลเบื้องหลัง ความแตกต่างระหว่างรุ่นหรือ driver หรือเกร็ดความรู้ที่เป็นประโยชน์

> **คำเตือน**
> ข้อผิดพลาดที่พบบ่อย หรือสิ่งที่อาจทำให้ข้อมูลสูญหาย เกิดข้อผิดพลาด หรือเกิดปัญหาด้านความปลอดภัย

โค้ด VFP และ T-SQL แสดงด้วยตัวอักษร `monospace` และคีย์เวิร์ดของ VFP และ T-SQL เขียนด้วยตัวพิมพ์ใหญ่ บรรทัดที่ยาวจะถูกตัดด้วยอักขระต่อบรรทัดของ VFP คือ `;` การตั้งชื่อใช้ข้อตกลงเดียวกันตลอดทั้งเล่ม ได้แก่ cursor ขึ้นต้นด้วย `cur` remote view ขึ้นต้นด้วย `rv_` คลาส CursorAdapter ขึ้นต้นด้วย `ca` stored procedure ขึ้นต้นด้วย `usp_` และตัวแปรใช้รูปแบบ Hungarian ของ VFP (`lcName`, `pnContrId`) ซึ่งสรุปไว้ในภาคผนวก B

หมายเหตุสำหรับผู้อ่านชาวไทย

- โค้ด ชื่อ field และผลลัพธ์ที่โปรแกรมแสดง คงไว้เป็นภาษาอังกฤษตามต้นฉบับ ให้ตรงกับที่คุณจะเห็นในเครื่องมือจริง
- ผลลัพธ์ในหนังสือใช้รูปแบบวันที่และตัวเลขแบบสหรัฐฯ (MM/DD/YYYY และ $) หากคุณใช้ Windows ที่ตั้งค่าภาษาและภูมิภาคเป็นไทย สิ่งที่แสดงบนเครื่องของคุณอาจมีรูปแบบต่างออกไป
- ในจุดที่เป็นประโยชน์ เราได้เพิ่มหมายเหตุเกี่ยวกับข้อมูลภาษาไทย (code page 874 และ collation `Thai_CI_AS`) ไว้ในบทที่เกี่ยวข้อง

สารบัญ

คำนำ

ภาคที่ 1 — จาก DBF สู่ CLIENT-SERVER

1 ทำไมต้องย้ายแอปพลิเคชัน VFP ไปสู่ SQL Server: แนวคิด Client-Server และแผนงาน RentDesk

2 เตรียม SQL Server สำหรับงานพัฒนา VFP: Instance เครื่องมือ และฐานข้อมูล Brackenridge

3 ODBC Driver และ Connection String: DSN, DSN-less Connection และข้อเท็จจริงเรื่อง 32 บิต

4 การเชื่อมต่อจาก VFP: SQLSTRINGCONNECT, SQLCONNECT, SQLSETPROP และ Shared Connection

5 การจับคู่ชนิดข้อมูล: Currency, Date, Memo, Logical, NULL และ Unicode

ภาคที่ 2 — SQL PASS-THROUGH

6 พื้นฐาน SQL Pass-Through: SQLEXEC, Result Cursor และ SQL ฝั่งเซิร์ฟเวอร์

7 คิวรีแบบมีพารามิเตอร์: ?Parameter, ขอบเขตตัวแปร และการป้องกัน SQL Injection

8 Result Set หลายชุด, SQLMORERESULTS, SQLPREPARE และการทำงานแบบ Asynchronous

9 การจัดการข้อผิดพลาด: AERROR, ข้อผิดพลาดของ ODBC และคลาส BrConnection

10 การแก้ไขข้อมูลด้วย SQL Pass-Through: INSERT, UPDATE, DELETE และ Updatable Cursor

ภาคที่ 3 — REMOTE VIEW และ CURSORADAPTER

11 Remote View และ Connection Definition ใน DBC

12 Updatable Remote View: Key Field, Update Name, WhereType และ Buffering

13 พื้นฐาน CursorAdapter: แหล่งข้อมูล ODBC, CursorSchema และ CursorFill

14 CursorAdapter เชิงลึก: การอัปเดต, Event, ADO และคลาสฐาน BrCursorAdapter

15 เลือกเทคนิคการเข้าถึงข้อมูล และการผูกข้อมูลระยะไกลเข้ากับฟอร์ม

ภาคที่ 4 — ตรรกะฝั่งเซิร์ฟเวอร์และความถูกต้องของข้อมูล

16 ทรานแซกชัน: Manual Mode, SQLCOMMIT, SQLROLLBACK และทรานแซกชันฝั่งเซิร์ฟเวอร์

17 เรียกใช้ Stored Procedure จาก VFP: Input, Output และ Result Set

18 ค่า Identity, Sequence และเลขที่เอกสารทางธุรกิจ

19 การทำงานพร้อมกัน: Update Conflict, rowversion และการแก้ปัญหาการชนกันของข้อมูล

20 Constraint, Trigger และการตรวจสอบความถูกต้อง: ย้ายกฎไปไว้ที่เซิร์ฟเวอร์

ภาคที่ 5 — การย้ายระบบ ประสิทธิภาพ และความปลอดภัย

21 วางแผนการย้ายระบบ: สำรวจระบบเดิม ออกแบบ Schema และตัดสินใจเรื่องชนิดข้อมูล

22 ย้ายข้อมูลจริง: Upsizing จาก DBF ด้วยสคริปต์ และการพิสูจน์ว่าย้ายถูกต้อง

23 ประสิทธิภาพ: Fetch Size, Progressive Fetching, Index และการหลีกเลี่ยงคิวรีแบบ N+1

24 ความปลอดภัย: Windows Authentication, SQL Login, สิทธิ์น้อยที่สุด และ SQL Injection

ภาคที่ 6 — การติดตั้งใช้งานและโปรเจกต์สรุป

25 การติดตั้งใช้งาน: Driver, ไฟล์ตั้งค่า และการทยอยติดตั้งโปรแกรมฝั่ง Client

26 รายงานและหน้าจอที่อ่านข้อมูลหนักจาก SQL Server

27 การวินิจฉัยและแก้ปัญหา: Logging, Tracing และความผิดพลาดที่พบบ่อย

28 โปรเจกต์สรุป: แปลงโมดูลสัญญาเช่าของ RentDesk จาก DBF สู่ SQL Server

ภาคผนวก

A สรุปย่อการพัฒนาแบบ Client-Server

B เทมเพลตและข้อมูลอ้างอิง

C รายการตรวจสอบก่อนใช้งานจริง

D อภิธานศัพท์

ภาคที่ 1

จาก DBF สู่ Client-Server

อะไรเปลี่ยนไปจริง ๆ เมื่อข้อมูลของ RentDesk ออกจากโฟลเดอร์แชร์: แนวคิดแบบ client-server, SQL Server instance และฐานข้อมูล Brackenridge, ODBC driver แบบ 32 บิตและ connection string, connection handle และการตั้งค่า รวมถึงการเดินทางของชนิดข้อมูลแต่ละแบบของ VFP ไปยังเซิร์ฟเวอร์และกลับมา

1 · ทำไมต้องย้ายไป SQL Server
2 · เตรียม SQL Server
3 · ODBC Driver และ Connection String
4 · การเชื่อมต่อจาก VFP
5 · การจับคู่ชนิดข้อมูล

บทที่ 1 · ภาคที่ 1 — จาก DBF สู่ CLIENT-SERVER

ทำไมต้องย้ายแอปพลิเคชัน VFP ไปสู่ SQL Server: แนวคิด Client-Server และแผนงาน RentDesk

ก่อนจะเขียน connection string แม้แต่บรรทัดเดียว เราต้องเห็นภาพให้ชัดก่อนว่าอะไรจะเปลี่ยนไปบ้าง เมื่อแอปพลิเคชัน Visual FoxPro เลิกเปิดไฟล์ DBF จากโฟลเดอร์แชร์ แล้วหันมาส่งคำขอไปยังเซิร์ฟเวอร์ฐานข้อมูลแทน บทนี้จะแนะนำ RentDesk ซึ่งเป็นแอปพลิเคชันที่เราจะแปลงตลอดทั้งเล่ม อธิบายว่าทำไมการออกแบบแบบอิงไฟล์ของมันจึงมีปัญหาเมื่อใช้งานข้ามสามลานเก็บอุปกรณ์ และวางแผนงานที่บทต่อ ๆ ไปจะเดินตาม

แอปพลิเคชันที่เรากำลังจะย้าย

Brackenridge Equipment Rentals เป็นบริษัทให้เช่าอุปกรณ์สมมติที่ให้เช่าเครื่องจักรก่อสร้าง ได้แก่ รถขุดขนาดเล็กและขนาดกะทัดรัด รถตักล้อยาง (skid steer loader) รถกระเช้าแบบกรรไกรและแบบบูม เครื่องตบดิน รถบดร่องดิน และเครื่องปั่นไฟ บริษัทดำเนินงานจากลานเก็บอุปกรณ์ (yard) สามแห่ง แห่งแรกคือ Riverside Yard ในเมือง Riverside ซึ่งเป็นที่ตั้งของสำนักงานและไฟล์เซิร์ฟเวอร์ด้วย ต่อมาอีกไม่กี่ปีจึงเปิด North Depot ในเมือง Kellerton และแห่งใหม่ล่าสุดคือ Harbor Yard ในเมือง Port Lindell แต่ละลานมีเคาน์เตอร์ที่ลูกค้ามารับอุปกรณ์ มีโรงซ่อมที่ตรวจสภาพอุปกรณ์เมื่อส่งคืน และมีพีซีหนึ่งหรือสองเครื่องที่รันระบบให้เช่า

ระบบให้เช่านี้ชื่อว่า **RentDesk** เป็นแอปพลิเคชัน Visual FoxPro ที่คอมไพล์เป็น `rentdesk.exe` และทำทุกอย่างที่พนักงานหน้าเคาน์เตอร์ต้องใช้ ตั้งแต่เก็บรายชื่อลูกค้า ติดตามอุปกรณ์ทุกชิ้นและสถานะของมัน เปิดสัญญาเช่า เพิ่มรายการอุปกรณ์ลงในสัญญา บันทึกการส่งคืนพร้อมสภาพของแต่ละชิ้น คำนวณจำนวนวันเช่าและค่าเช่า ไปจนถึงพิมพ์ใบแจ้งหนี้ คนที่เขียนโปรแกรมนี้และยังดูแลมาจนถึงทุกวันนี้คือ **Nora Kessler** นักพัฒนาประจำบริษัท ตลอดทั้งเล่มเราจะทำงานเคียงข้าง Nora เราจะอ่านโค้ดของเธอ เคารพการตัดสินใจที่เธอทำไว้เมื่อหลายปีก่อน และช่วยเธอย้ายข้อมูลที่อยู่เบื้องหลัง RentDesk ไปไว้บน Microsoft SQL Server โดยไม่ต้องเขียนส่วนที่ทำงานได้ดีอยู่แล้วของแอปพลิเคชันขึ้นมาใหม่

RentDesk เก็บข้อมูลไว้ใน free table แบบ DBF ในโฟลเดอร์เดียวบนไฟล์เซิร์ฟเวอร์ที่ Riverside บนพีซีของนักพัฒนา table ชุดเดียวกันจะอยู่ที่ `C:\Brackenridge\Data\` และเราจะใช้ path นี้ในตัวอย่าง ส่วนซอร์สโค้ดอยู่ที่ `C:\Brackenridge\RentDesk\` โดย `main.prg` ใช้เริ่มแอปพลิเคชัน `brlib.prg` เก็บกฎทางธุรกิจ (จำนวนวันเช่า ค่าเช่า ภาษีขาย และการจัดรูปแบบตัวเลขเงิน) ส่วน form และรายงานเป็นส่วนหนึ่งของโปรเจกต์ `rentdesk.pjx` บทหลัง ๆ จะเพิ่ม `brdata.prg`, `brmigrate.prg`, `brca.prg`, database container ชื่อ `brremote.dbc` และไฟล์ตั้งค่าชื่อ `rentdesk.ini` ส่วนสคริปต์ฝั่งเซิร์ฟเวอร์จะเก็บไว้ใน `C:\Brackenridge\Sql\`

table แบบ DBF

RentDesk ใช้ free table แปด table เจ็ด table เก็บข้อมูลทางธุรกิจ และอีกหนึ่ง table คือ `counters` ทำหน้าที่แจกเลขที่สัญญาและเลขที่ใบแจ้งหนี้ถัดไป ตารางด้านล่างสรุปไว้ทั้งหมด ชื่อ field ทุกชื่อยาวไม่เกินสิบตัวอักษร ซึ่งจะมีความสำคัญในภายหลัง เพราะเราจะใช้ชื่อเดิมทุกตัวอักษรเป็นชื่อคอลัมน์บน SQL Server ทำให้โค้ด VFP ที่อ้างถึง `day_rate` หรือ `contr_no` ไม่ต้องแก้ไขเลย

Table	เก็บอะไร	Field หลัก	จำนวนแถว
branches	ลานเก็บอุปกรณ์ทั้งสามแห่ง	branch_id C(3)	3
customers	ลูกค้าแบบมีบัญชีและลูกค้าเงินสด	cust_id I, cust_no C(6)	5
equipment	อุปกรณ์ทุกชิ้นที่ให้เช่าได้ อัตราค่าเช่า และสถานะ	equip_id I, asset_tag C(8)	8
contracts	สัญญาเช่า (ส่วนหัว)	contr_id I, contr_no C(8)	4
conlines	รายการอุปกรณ์ในสัญญา	line_id I, contr_id I	6
returns	บันทึกการส่งคืนหนึ่งรายการต่อหนึ่งบรรทัด	return_id I, line_id I	3
invoices	ใบแจ้งหนี้หนึ่งใบต่อสัญญาที่ปิดแล้ว	invoice_id I, contr_id I	2
counters	เลขที่สัญญาและเลขที่ใบแจ้งหนี้ถัดไป	name C(10)	2

ข้อมูลตัวอย่างตั้งใจให้มีขนาดเล็ก เพื่อให้คุณตรวจสอบตัวเลขทุกตัวในเล่มด้วยมือได้ มีลูกค้าห้าราย ตั้งแต่ Halvorsen Builders LLC ผู้รับเหมาก่อสร้างที่มีวงเงินเครดิต $15,000.00 ไปจนถึง Dale Okonkwo เจ้าของบ้านที่จ่ายเงินสด มีอุปกรณ์แปดชิ้น ตั้งแต่ Mini Excavator 3.5 t (asset tag `BR-EX-01` วันละ $285.00) ไปจนถึง Trench Roller (`BR-TR-01`) ที่ตอนนี้อยู่ในโรงซ่อม และมีสัญญาสี่ฉบับ สองฉบับปิดและออกใบแจ้งหนี้แล้ว อีกสองฉบับยังเปิดอยู่และมีอุปกรณ์ออกไปอยู่ที่หน้างาน

คุณสร้าง table เหล่านี้เองได้ด้วยโปรแกรม `setup-dbf.prg` ซึ่งมากับไฟล์ตัวอย่างของหนังสือ โปรแกรมนี้จะสร้าง free table พร้อม index แบบ CDX และแถวข้อมูลตัวอย่างลงในโฟลเดอร์ใดก็ได้ที่คุณส่งให้ โปรแกรมด้านล่างเรียกใช้มัน แล้วแสดงรายชื่อแต่ละ table พร้อมจำนวน record และจำนวน field:

`C:\Brackenridge\RentDesk\showdbf.prg`

```
LOCAL lcData, laT[8], lnI
DO setup-dbf WITH "C:\Brackenridge\Data"
lcData = "C:\Brackenridge\Data\"
SET EXCLUSIVE OFF
laT[1] = "branches"
laT[2] = "customers"
laT[3] = "equipment"
laT[4] = "contracts"
```

```
laT[5] = "conlines"
laT[6] = "returns"
laT[7] = "invoices"
laT[8] = "counters"
FOR lnI = 1 TO 8
   USE (lcData + laT[lnI]) SHARED
   ? PADR(laT[lnI], 10), RECCOUNT(), FCOUNT()
   USE
ENDFOR
USE (lcData + "counters") SHARED
LIST
USE
```

ผลลัพธ์ที่แสดง

```
branches            3          4
customers           5         11
equipment           8         11
contracts           4         11
conlines            6          6
returns             3          9
invoices            2         10
counters            2          2
Record#  NAME          NEXT_VAL
      1  CONTRACT         26005
      2  INVOICE          26003
```

แถวใน `counters` บอกเราว่าสัญญาฉบับถัดไปจะเป็น **RC-26005** และใบแจ้งหนี้ใบถัดไปจะเป็น **IN-26003** จำตัวเลขเหล่านี้ไว้ เพราะจะกลับมาอีกครั้งเมื่อเราแทนที่ table counters ด้วย sequence ของ SQL Server ในบทที่ 18

วิธีที่ RentDesk อ่านข้อมูลในปัจจุบัน

โค้ดของ Nora ใช้คำสั่ง Xbase แบบดั้งเดิมที่นักพัฒนา VFP ทุกคนรู้จัก เมื่อต้องการแสดงอุปกรณ์ที่ว่างอยู่ที่ Riverside หน้าจออุปกรณ์จะเปิด table ตามลำดับของ tag `br_status` (index บน `branch_id + status`) แล้ว seek หาคีย์แรกที่ตรง และ scan ไปเรื่อย ๆ ตราบที่คีย์ยังตรงอยู่:

โค้ด RentDesk เดิม (เวอร์ชัน DBF)

```
USE C:\Brackenridge\Data\equipment ORDER br_status SHARED
SEEK "RVSA"                              && branch RVS, status A (available)
SCAN WHILE branch_id + status = "RVSA"
   ? asset_tag, descr, day_rate
ENDSCAN
USE
```

ผลลัพธ์ที่แสดง

```
BR-EX-01 Mini Excavator 3.5 t                              285.0000
BR-SL-01 Skid Steer Loader                                 245.0000
BR-CP-01 Plate Compactor                                    65.0000
```

โค้ดนี้ทำงานเร็วและอ่านเข้าใจง่ายเมื่อใช้บนดิสก์ในเครื่องหรือบนเครือข่ายสำนักงานที่เร็ว แต่ก็ "คุยกันบ่อย" มากในแบบที่มองข้ามได้ง่าย ในที่นี้ VFP ไม่ใช่เซิร์ฟเวอร์ เป็นผู้ทำงานทั้งหมด มันอ่านส่วนหัวของ CDX และหน้า index ผ่านเครือข่าย ไล่ index เพื่อหา `RVSA` แล้วจึงอ่านแต่ละ record ที่ตรงจากไฟล์ DBF ไฟล์เซิร์ฟเวอร์ทำหน้าที่เพียงส่งบล็อกของไบต์ให้เท่านั้น มันไม่รู้เลยว่ามี query กำลังทำงาน ผู้ใช้ต้องการแถวไหน หรือผู้ใช้อีกคนกำลังแก้ไขแถวไหนอยู่ในขณะเดียวกัน

ทำไมไฟล์ DBF แบบแชร์จึงมีปัญหาเมื่อใช้ข้ามสามลาน

สมัยที่ RentDesk ใช้งานเฉพาะที่ Riverside พีซีทุกเครื่องอยู่บนเครือข่ายสำนักงานเดียวกับไฟล์เซิร์ฟเวอร์ และไม่มีใครบ่นอะไร เมื่อเปิด North Depot และ Harbor Yard บริษัทเชื่อมทั้งสองแห่งเข้ามาผ่าน VPN และเปิด RentDesk จากโฟลเดอร์แชร์ที่ทั้งสามลาน ปัญหาเริ่มต้นตรงนั้น ปัญหาทั้งหมดด้านล่างไม่ใช่บั๊กในโค้ดของ Nora แต่เป็นธรรมชาติของการออกแบบแบบแชร์ไฟล์ที่ถูกยืดออกไปใช้บนเครือข่ายระยะไกล (wide-area network)

ทุกคำขอต้องขนบล็อกไฟล์ดิบไปมา

ฐานข้อมูลแบบไฟล์เซิร์ฟเวอร์ขนหน้าของไฟล์ ไม่ได้ขนคำตอบ การเปิด table คือการอ่านส่วนหัว `SEEK` อ่านหน้า index ส่วน `SCAN FOR` ที่ไม่มี index ซึ่ง optimize ได้อาจต้องอ่านทั้ง table รายงานที่รวมยอดใบแจ้งหนี้ของทั้งเดือน ต้องอ่านทุก record ของใบแจ้งหนี้ผ่านเครือข่าย บวกยอดที่ฝั่ง client แล้วทิ้งไบต์ส่วนใหญ่ไป บนเครือข่ายภายใน ต้นทุนนี้ถูกซ่อนไว้ด้วยแบนด์วิดท์สูง และ latency ต่ำ แต่เมื่อผ่าน VPN การรับส่งแต่ละรอบ (round trip) ใช้เวลานานขึ้นอย่างเห็นได้ชัด และงานที่ต้องอ่านเล็ก ๆ หลายร้อยครั้ง ก็ช้าจนพนักงานหน้าเคาน์เตอร์รู้สึกได้ Rushmore optimization ช่วยได้ด้วยการอ่านหน้า index แทน record แต่ก็ยังทำงานที่ฝั่ง client และยังต้องดึงหน้า index ผ่านสายอยู่ดี

lock เดินทางผ่านระบบไฟล์ และการเขียนที่ถูกขัดจังหวะก่อความเสียหาย

VFP ประสานการใช้งานหลายผู้ใช้ด้วย lock ระดับไฟล์และระดับ record ที่ไฟล์เซิร์ฟเวอร์จัดการแทนให้ และ client แต่ละเครื่องก็แคชบางส่วนของไฟล์ไว้ การรักษาให้ lock และแคชสอดคล้องกันต้องอาศัยการสื่อสารเล็ก ๆ จำนวนมาก ซึ่งแต่ละครั้งช้าลงเมื่อผ่าน VPN ที่แย่กว่านั้นคือ table DBF ถูกเขียนโดย client เครื่องใดก็ตามที่กำลังอัปเดตมันอยู่ หากการเชื่อมต่อของ client เครื่องนั้นหลุดกลางคันระหว่างเขียน ส่วนหัวของ table ไฟล์ memo หรือ index CDX อาจอยู่ในสภาพที่ไม่สอดคล้องกัน นักพัฒนา VFP ที่ทำงานมานานทุกคนเคยต้องสั่ง `REINDEX` หลังเจอข้อผิดพลาด "index does not match the table" มาแล้ว รูทีน pack และ reindex ทุกคืนของ Nora ใช้งานได้ แต่มันเป็นอาการของการออกแบบไม่ใช่การรักษา

ข้อจำกัดขนาดต่อไฟล์

ไฟล์ DBF, FPT และ CDX แต่ละไฟล์มีขนาดได้ไม่เกิน 2 GB ตอนนี้ RentDesk ยังห่างไกลจากขีดนั้นมาก แต่ประวัติสัญญาเพิ่มขึ้นทุกปี และการหลบเลี่ยงข้อจำกัดนี้หมายถึงต้องย้ายแถวเก่าไปเก็บถาวร และสอนรายงานทุกตัวให้ไปหาข้อมูลสองที่

ไม่มีความปลอดภัยหรือกฎฝั่งเซิร์ฟเวอร์

ใครก็ตามที่เปิดโฟลเดอร์แชร์ได้ก็คัดลอก table ได้ เปิดด้วยโปรแกรมอื่นได้ หรือแก้ไขข้อมูลด้วยโค้ดไม่กี่บรรทัดที่ข้ามการตรวจสอบของ RentDesk ไปได้ free table ไม่มีกฎของตัวเอง การตรวจสอบทั้งหมดอยู่ในแอปพลิเคชัน database container เพิ่มกฎระดับ field และระดับแถวรวมถึง trigger ได้ แต่กฎเหล่านั้นก็ยังถูกบังคับใช้โดย process ของ VFP ที่เปิดไฟล์อยู่ ไม่ใช่โดยเซิร์ฟเวอร์ที่ client ทุกเครื่องต้องผ่าน

การสำรองข้อมูลต้องการไฟล์ที่นิ่ง

การคัดลอกชุดไฟล์ DBF ให้ได้สำเนาที่เชื่อถือได้ ต้องมีช่วงเวลาที่ไม่มีใครเขียนไฟล์เลย ซึ่งในทางปฏิบัติคือต้องขอให้ทุกคนออกจาก RentDesk ไม่มี transaction log ให้เล่นซ้ำ และไม่มีทางกู้ข้อมูลกลับไปยังสถานะ ณ เวลา 10:15 น. ของเช้าวันนี้ได้

> **หมายเหตุ**
> ทั้งหมดนี้ไม่ได้แปลว่า table DBF เป็นการออกแบบที่แย่ สำหรับสำนักงานเดียวบนเครือข่ายภายใน มันเร็ว เรียบง่าย และทนทานอย่างน่าทึ่ง แอปพลิเคชัน VFP จำนวนมากใช้งานมันมาหลายสิบปี ปัญหาจะปรากฏเมื่อเครือข่ายระหว่าง client กับไฟล์ช้าลงหรือไม่เสถียร หรือเมื่อธุรกิจต้องการความปลอดภัย การตรวจสอบย้อนหลัง (auditing) และการสำรองข้อมูลที่โฟลเดอร์แชร์ให้ไม่ได้

Client-Server เปลี่ยนอะไรบ้าง

ในการออกแบบแบบ client-server จะมีโปรแกรมเดียวบนเครื่องเดียวที่เป็นเจ้าของไฟล์ข้อมูล นั่นคือเซิร์ฟเวอร์ฐานข้อมูล client ไม่เคยเปิดไฟล์เหล่านั้นเอง แต่จะเชื่อมต่อกับเซิร์ฟเวอร์แล้วส่งคำขอที่เขียนเป็น SQL ไปให้ เซิร์ฟเวอร์อ่านไฟล์ของตัวเองบนดิสก์ของตัวเอง ทำการกรอง join เรียงลำดับ และรวมยอด แล้วส่งกลับมาเฉพาะผลลัพธ์ สำหรับ RentDesk เซิร์ฟเวอร์คือ Microsoft SQL Server และคำขอเขียนด้วย Transact-SQL (T-SQL) ซึ่งเป็นภาษา SQL สำเนียงของ SQL Server

นี่คือคำถามเรื่องอุปกรณ์ว่างที่ Riverside อีกครั้ง คราวนี้ถาม SQL Server ผ่าน stored procedure ที่เป็นส่วนหนึ่งของ schema ของเรา โค้ด VFP เปิด connection ส่งคำขอหนึ่งครั้ง และได้รับ cursor ที่มีคำตอบกลับมา:

เวอร์ชัน client-server (SQL pass-through)

```
LOCAL lnHandle, lnResult
SQLSETPROP(0, "DispLogin", 3)               && never show the ODBC login dialog
lnHandle = SQLSTRINGCONNECT("Driver={ODBC Driver 17 for SQL Server};" + ;
   "Server=.\SQLEXPRESS;Database=Brackenridge;Trusted_Connection=Yes;")
lnResult = SQLEXEC(lnHandle, ;
   "EXEC dbo.usp_EquipmentAvailability @branch_id = 'RVS'", "curAvail")
? lnResult, RECCOUNT("curAvail")
SELECT curAvail
SCAN
```

```
    ? asset_tag, descr, day_rate
ENDSCAN
SQLDISCONNECT(lnHandle)
```

ผลลัพธ์ที่แสดง

```
         1          3
BR-CP-01 Plate Compactor                          65.0000
BR-EX-01 Mini Excavator 3.5 t                    285.0000
BR-SL-01 Skid Steer Loader                       245.0000
```

อุปกรณ์สามชิ้นเดิมกลับมา คราวนี้เรียงตาม asset tag เพราะ procedure ระบุ `ORDER BY e.asset_tag` ไว้ `SQLEXEC()` คืนค่า 1 ซึ่งหมายถึงได้ result set หนึ่งชุด และ `curAvail` ก็เป็น cursor ธรรมดาของ VFP คุณ `SCAN` มันได้ ผูกเข้ากับ grid ได้ หรือรัน `SELECT` แบบ local กับมันก็ได้ สิ่งที่วิ่งผ่านเครือข่ายคือคำขอหนึ่งครั้งกับข้อมูลสามแถว ไม่ใช่หน้า index และ record ตอนนี้ยังไม่ต้องกังวลกับรายละเอียด บทที่ 3 และ 4 จะอธิบาย connection string และ handle ส่วนบทที่ 6 จะอธิบาย `SQLEXEC()`

การเปลี่ยนแปลงนี้สรุปได้ด้วยแนวคิดสามข้อ ซึ่งเนื้อหาที่เหลือของเล่มจะวนกลับมาหาอยู่เสมอ

ขอข้อมูลเป็นชุด ไม่ใช่ทีละ record

โค้ดแบบอิงไฟล์จะ "เดินไปทีละก้าว" คือ เปิด seek skip ทดสอบ แล้ว skip อีก ส่วนโค้ดแบบ client-server จะบรรยายว่าต้องการแถวแบบไหน แล้วให้เซิร์ฟเวอร์หาเอง โดยขอเฉพาะแถวและคอลัมน์ที่หน้าจอต้องใช้จริงภายใน round trip เดียว การโหลดทั้ง table มา "เผื่อไว้" เป็นสาเหตุที่พบบ่อยที่ทำให้แอปพลิเคชันที่แปลงแล้วรู้สึกช้า (บทที่ 6, 15 และ 23)

เซิร์ฟเวอร์เป็นผู้บังคับใช้กฎ

SQL Server ตรวจสอบ primary key ค่าที่ต้องไม่ซ้ำ foreign key รหัสสถานะที่อนุญาต และค่า default ให้กับ client ทุกตัว ไม่ว่าการเปลี่ยนแปลงจะมาจาก RentDesk เครื่องมือออกรายงาน หรือสคริปต์ของผู้ดูแลระบบ เซิร์ฟเวอร์ห่อการเปลี่ยนแปลงที่เกี่ยวข้องกันไว้ใน ทรานแซกชัน (transaction) เพื่อให้รายการในสัญญากับสถานะของอุปกรณ์เปลี่ยนไปพร้อมกัน หรือไม่เปลี่ยนเลยทั้งคู่ RentDesk ยังคงตรวจสอบข้อมูลที่ป้อนเข้ามาเพื่อแสดงข้อความที่เป็นมิตร แต่เซิร์ฟเวอร์คือผู้ตัดสินขั้นสุดท้าย

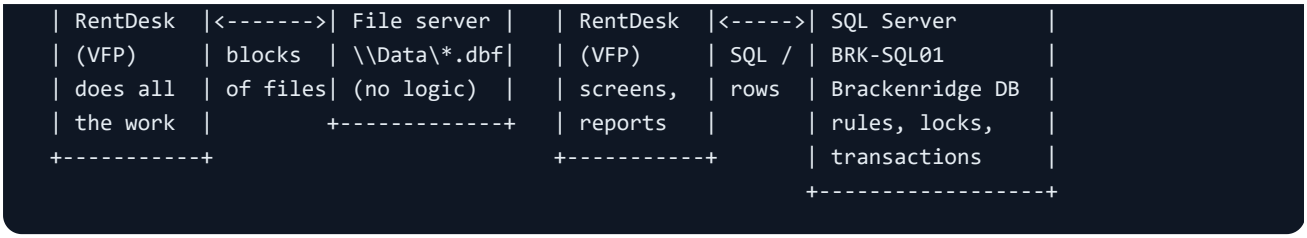
มีเพียงผลลัพธ์ที่วิ่งผ่านเครือข่าย

เพราะเซิร์ฟเวอร์ทำงานอยู่ติดกับดิสก์ของตัวเอง ปริมาณข้อมูลระหว่างลานเก็บอุปกรณ์กับเซิร์ฟเวอร์จึงเท่ากับขนาดของคำขอบวกขนาดของคำตอบ ยอดรายได้รายเดือนก็คือแถวเดียว นี่คือสิ่งที่ทำให้การออกแบบนี้เหมาะกับ WAN สิ่งที่สำคัญคือจำนวน round trip และขนาดของผลลัพธ์ ไม่ใช่ขนาดของ table

ก่อนและหลัง

```
BEFORE: file-server database          AFTER: client-server database

+-----------+   VPN   +-------------+   +-----------+  VPN  +------------------+
```



> **เคล็ดลับ**
>
> เมื่อคุณไม่แน่ใจว่าจะจัดโครงสร้างโค้ดที่แปลงแล้วอย่างไร ให้ถามตัวเองว่าโค้ดนั้นคุยกับเซิร์ฟเวอร์กี่ครั้ง และดึงแถวกลับมากี่แถว round trip น้อยกว่าและแถวน้อยกว่าแทบจะชนะเสมอ คำถามข้อเดียวนี้อธิบายคำแนะนำด้านประสิทธิภาพส่วนใหญ่ในเล่มได้

สิ่งที่ยังคงอยู่ใน Visual FoxPro

การย้ายข้อมูลไม่ได้หมายความว่าต้องเขียนแอปพลิเคชันใหม่ form เมนู รายงาน และโค้ดทางธุรกิจส่วนใหญ่ของ RentDesk ยังอยู่ใน VFP พนักงานหน้าเคาน์เตอร์ยังได้ใช้หน้าจอที่คุ้นเคย VFP ยังคงเป็นเครื่องมือชั้นยอดสำหรับสร้าง form บันทึกข้อมูลและรายงานบน cursor และ cursor ที่เติมข้อมูลมาจาก SQL Server ก็ทำงานเหมือน cursor อื่น ๆ ทุกประการเมื่อมาอยู่ที่ฝั่ง client แล้ว

- **Form และ grid** ยังอยู่ เพียงแต่แทนที่จะผูกกับ table ที่เปิดด้วย `USE` ก็เปลี่ยนไปผูกกับ cursor ที่เติมข้อมูลจากเซิร์ฟเวอร์ (บทที่ 15)
- **รายงาน** ยังอยู่ โดยรันจาก cursor ที่รวบรวมข้อมูลบนเซิร์ฟเวอร์ด้วยคำขอเดียว (บทที่ 26)
- **ฟังก์ชันทางธุรกิจ** ใน `brlib.prg` ยังอยู่ `RentalDays()`, `RentalCharge()`, `CalcTax()` และ `MoneyStr()` ยังมีประโยชน์สำหรับแสดงตัวอย่างบนหน้าจอ และกฎชุดเดียวกันก็มีอยู่บนเซิร์ฟเวอร์ในรูปฟังก์ชัน T-SQL เพื่อให้ทั้งสองฝั่งได้ผลตรงกัน
- **ไฟล์ executable และการ deploy** ยังอยู่ โดยเพิ่มสองอย่างบนพีซีแต่ละเครื่อง คือ ODBC driver แบบ 32 บิต สำหรับ SQL Server และไฟล์ตั้งค่าขนาดเล็กที่บอกว่าจะใช้เซิร์ฟเวอร์ตัวไหน (บทที่ 25)

สิ่งที่เปลี่ยนคือชั้นการเข้าถึงข้อมูล (data-access layer) ได้แก่ `USE`, `SEEK`, `SCAN`, `REPLACE`, `APPEND BLANK`, `RLOCK()` และ `FLOCK()` ทุกจุดที่แตะ table แบบแชร์ ใน RentDesk คำสั่งเหล่านี้กระจายอยู่ทั่ว form และ procedure ซึ่งเป็นเรื่องปกติของแอปพลิเคชันที่เติบโตมาหลายปี งานส่วนหนึ่งคือการหาคำสั่งเหล่านี้ให้ครบ ซึ่งบทที่ 21 จะทำเป็นบัญชีรายการ (inventory) อีกส่วนคือการแทนที่ด้วยรูปแบบ (pattern) จำนวนไม่มากที่ทดสอบมาอย่างดี โค้ดล็อก record จะหายไปทั้งหมด เพราะ SQL Server จัดการ locking เอง และ table `counters` ที่มีรูทีน lock-เพิ่มค่า-unlock ก็จะถูกแทนที่ด้วย sequence ของเซิร์ฟเวอร์

> **หมายเหตุ**
> หากคุณมาจากสาย .NET และ ADO.NET ให้นึกถึง cursor ของ VFP ที่เติมข้อมูลด้วย `SQLEXEC()` ว่าเป็น `DataTable` แบบ disconnected ที่ภาษา VFP คิวรี สร้าง index และผูกกับหน้าจอได้โดยตรง connection handle ทำหน้าที่แบบเดียวกับ `SqlConnection` และ updatable cursor ก็คล้ายกับสิ่งที่ `DataAdapter` ที่มีคำสั่งอัปเดตให้คุณ

สามวิธีที่ VFP คุยกับ SQL Server

Visual FoxPro 9 มีเทคนิคในตัวสามแบบสำหรับทำงานกับข้อมูลระยะไกล ทั้งสามแบบใช้ connection ชนิดเดียวกันอยู่เบื้องล่าง (เล่มนี้ใช้ ODBC เป็นหลัก โดยมี ADO เป็นทางเลือกสำหรับ CursorAdapter) และทั้งหมดให้ผลเป็น cursor ธรรมดาของ VFP ความแตกต่างอยู่ที่ว่า VFP ทำงานแทนคุณมากแค่ไหน และคุณยังควบคุมได้มากเพียงใด

เทคนิค	คืออะไร	จุดแข็ง	บทที่
SQL pass-through (SPT)	ฟังก์ชันอย่าง `SQLEXEC()` ที่ส่งข้อความ SQL ที่คุณเขียนเอง	ควบคุมได้เต็มที่ ใช้ T-SQL ได้ทุกแบบ เรียก stored procedure ได้ มีเรื่องให้ประหลาดใจน้อยที่สุด	6–10, 16–20
Remote view	view ที่เก็บใน DBC ซึ่ง VFP แปลงเป็น SQL ให้	ใช้ `USE` แบบที่คุ้นเคย ทำรายการที่แก้ไขได้ง่าย มี View Designer	11–12
CursorAdapter	class ที่เติมข้อมูลและอัปเดต cursor หนึ่งตัว	class นำกลับมาใช้ซ้ำได้ มี event ใช้ได้ทั้ง ODBC และ ADO ทำงานในโค้ดได้	13–14

เราเริ่มจาก SQL pass-through เพราะเป็นรากฐาน remote view และ CursorAdapter สร้างคำขอชนิดเดียวกับที่ SPT ส่งออกไปอย่างชัดแจ้ง และเมื่อเกิดปัญหา คุณก็วินิจฉัยด้วย SPT ในที่สุด RentDesk ใช้ทั้งสามแบบ การอ่านข้อมูลเฉพาะกิจและการเรียก stored procedure ใช้ SPT ผ่าน connection class ขนาดเล็กชื่อ `BrConnection` ซึ่งเราสร้างในบทที่ 4 และทำให้สมบูรณ์ในบทที่ 9 รายการแก้ไขได้แบบง่าย ๆ บางรายการใช้ remote view ส่วนหน้าจอสัญญาใช้ CursorAdapter class บทที่ 15 จะเปรียบเทียบทั้งสามแบบ เคียงกันและอธิบายว่าควรใช้แบบไหนเมื่อใด

เป้าหมาย: ฐานข้อมูล Brackenridge

บน SQL Server ข้อมูลอยู่ใน database ชื่อ **Brackenridge** บนพีซีของนักพัฒนา มันรันบน instance ของ SQL Server Express ในเครื่องชื่อ `.\SQLEXPRESS` ส่วนใน production มันรันบนเซิร์ฟเวอร์ที่ Victor ผู้รับจ้างดูแลไอทีของบริษัท ตั้งชื่อว่า `BRK-SQL01` table ต่าง ๆ ใช้ชื่อ table และชื่อ field เดิมจาก DBF เพื่อให้ `equipment.day_rate` มีความหมายเดียวกันทั้งสองฝั่ง แผนภาพแสดง table ทั้งเจ็ดและความสัมพันธ์ระหว่างกัน:

schema ของ Brackenridge (dbo)



อ่านแผนภาพโดยเริ่มจาก branches ออกไป แต่ละสาขามีอุปกรณ์หลายชิ้นและสัญญาหลายฉบับ ลูกค้าแต่ละรายมีสัญญาหลายฉบับ แต่ละสัญญามีหลายบรรทัด แต่ละบรรทัดอ้างถึงอุปกรณ์หนึ่งชิ้นและมีการส่งคืนได้ไม่เกินหนึ่งรายการ และแต่ละสัญญามีใบแจ้งหนี้ได้ไม่เกินหนึ่งใบ มีบางอย่างที่ใหม่เมื่อเทียบกับ DBF และแต่ละเรื่องมีบทของตัวเอง:

- **คีย์แบบ IDENTITY** `cust_id`, `equip_id`, `contr_id` และคีย์จำนวนเต็มอื่น ๆ สร้างโดยเซิร์ฟเวอร์ ระหว่างการย้ายข้อมูลเราจะคงค่าเดิมจาก DBF ไว้ ลูกค้าหมายเลข 1 จึงยังเป็นลูกค้าหมายเลข 1 (บทที่ 18 และ 22)
- **Constraint** primary key, unique key อย่าง `uq_equipment_asset_tag`, foreign key อย่าง `fk_contracts_customers` และ check อย่าง `ck_equipment_status` (A, R หรือ M) ถูกประกาศไว้บนเซิร์ฟเวอร์และบังคับใช้กับทุกคน (บทที่ 20)
- **คอลัมน์ rowversion** `customers`, `equipment` และ `contracts` มีคอลัมน์ `row_ver` ที่ SQL Server เปลี่ยนค่าให้อัตโนมัติทุกครั้งที่อัปเดต ซึ่งเป็นพื้นฐานของการตรวจจับการชนกันของข้อมูลในบทที่ 19
- **Sequence** `dbo.seq_contract` และ `dbo.seq_invoice` มาแทน table `counters` ทำให้สองลานเปิดสัญญาพร้อมกันได้โดยไม่ต้องแย่ง lock กัน
- **ตรรกะฝั่งเซิร์ฟเวอร์** ฟังก์ชันสองตัวคือ `dbo.fn_RentalDays` และ `dbo.fn_RentalCharge` ใช้กฎเดียวกับ `brlib.prg` และ stored procedure หกตัวจัดการขั้นตอนงานหลัก ได้แก่ เปิดสัญญา เพิ่มบรรทัด รับคืนอุปกรณ์ ออกใบแจ้งหนี้ อ่านข้อมูลสัญญา และแสดงรายการอุปกรณ์ที่ว่าง

ตัวอย่างล่วงหน้าของการตัดสินใจเรื่องชนิดข้อมูล ชื่อลูกค้าเป็น `nvarchar(40)` คอลัมน์เงินเป็น `money` (Currency ของ VFP) field แบบ DateTime เป็น `datetime2(0)` วันที่เป็น `date` ค่า logical เป็น `bit` และ memo เป็น `varchar(max)` บทที่ 5 จะแสดงว่าแต่ละชนิดกลายเป็นอะไรใน cursor ของ VFP รวมถึงเรื่องน่าประหลาดใจหนึ่งเรื่องของ `varchar(max)`

กฎทางธุรกิจโดยย่อ

กฎเหล่านี้ควรกล่าวไว้สักครั้งตรงนี้ เพราะตัวเลขที่พิสูจน์ว่าโค้ดของเราถูกต้องล้วนมาจากกฎเหล่านี้ จำนวนวันเช่านับเป็นช่วง 24 ชั่วโมง ที่เริ่มแล้ว หลังจากช่วงผ่อนผัน 60 นาที โดยขั้นต่ำหนึ่งวัน ค่าเช่าคือจำนวนสัปดาห์เต็มคูณอัตรารายสัปดาห์ บวกจำนวนวันที่เหลือคูณอัตรารายวัน โดยส่วนที่เหลือนี้ต้องไม่เกินอัตรารายสัปดาห์หนึ่งสัปดาห์ ภาษีขายคือ 7.25% ของค่าเช่าบวกค่าเสียหาย ปัดเศษเป็นเซนต์ และเป็นศูนย์สำหรับลูกค้าที่ได้รับยกเว้นภาษี เช่น City of Kellerton อัตราค่าเช่าจะถูกคัดลอกลงในบรรทัดของสัญญาเมื่ออุปกรณ์ออกไป การเปลี่ยนราคาจึงไม่กระทบการเช่าที่อุปกรณ์อยู่หน้างานแล้ว ตัวอย่างเช่น รถขุดขนาดเล็กในสัญญา RC-26001 ออกไปเมื่อ 03/02/2026 เวลา 07:30 และกลับมาเมื่อ 03/09/2026 เวลา 16:10 คิดเป็น 8 วันเช่า เรียกเก็บเป็นหนึ่งสัปดาห์ $1,140.00 บวกหนึ่งวัน $285.00 รวม $1,425.00

ตัวละคร

มีบุคคลสี่คนปรากฏในเล่มนี้ ทุกคนเป็นบุคคลสมมติเช่นเดียวกับบริษัท

- **Gail Brackenridge** เป็นเจ้าของและผู้จัดการทั่วไป เธอเป็นคนถามคำถามทางธุรกิจ เช่น ทำไมเคาน์เตอร์ที่ North Depot ต้องรอหน้าจอ เดือนที่แล้วแต่ละลานทำรายได้เท่าไร และการย้ายระบบคุ้มค่าเงินหรือไม่
- **Nora Kessler** เป็นนักพัฒนา VFP ที่เขียน RentDesk เธอรู้จักทุก form และทุกความแปลกของข้อมูล เธอยังใหม่กับ SQL Server ซึ่งทำให้เธอเป็นเพื่อนร่วมงานที่เหมาะที่สุดสำหรับเล่มนี้ เราจะอธิบายแนวคิด client-server ให้เธอฟังจากมุมมองของ VFP
- **Tomas Reyes** เป็นผู้จัดการ Riverside Yard เขาตรวจอุปกรณ์ที่ส่งคืนและบันทึกสภาพ ชื่อของเขาปรากฏในคอลัมน์ `checked_by` ของการส่งคืนทุกรายการในข้อมูลตัวอย่าง
- **Victor Lam** เป็นผู้รับจ้างดูแลไอทีจากภายนอก เขาติดตั้งและดูแล SQL Server จัดการบัญชีและกลุ่มของ Windows และรับผิดชอบการสำรองข้อมูลและความปลอดภัย Nora เขียนแอปพลิเคชัน ส่วน Victor ตัดสินว่าใครเชื่อมต่อได้และเซิร์ฟเวอร์ถูกปกป้องอย่างไร

การแบ่งหน้าที่ระหว่าง Nora กับ Victor เป็นความตั้งใจ เพราะช่วยให้รู้ว่าการตัดสินใจใดเป็นของแอปพลิเคชัน และการตัดสินใจใดเป็นของผู้ดูแลเซิร์ฟเวอร์