

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

CLIENT-SERVER DEEP DIVE

Visual FoxPro and SQL Server

Client-Server Development with SQL Pass-Through, Remote Views, and CursorAdapter

ODBC drivers · Connections · Parameters · Type mapping
Transactions · Stored procedures · rowversion · Migration
Performance · Security · Deployment · Capstone

SQLEXEC · REMOTE VIEWS · CURSORADAPTER · T-SQL

THENEXBOOKS

Visual FoxPro and SQL Server

Client-Server Development with SQL Pass-Through, Remote Views, and CursorAdapter

Move a VFP 9 application's data from DBF tables to SQL Server — and keep the front end you already trust

First Edition

THENEXBOOKS

Visual FoxPro and SQL Server
Client-Server Development with SQL Pass-Through, Remote Views, and CursorAdapter

First Edition, 2026

Copyright © 2026 thenexbooks (thenexbooks.gumroad.com). All rights reserved.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means without the prior written permission of the publisher, except for brief quotations in reviews and certain other non-commercial uses permitted by copyright law. Code examples, T-SQL scripts, and Visual FoxPro programs may be used freely in your own projects.

The information in this book is provided "as is" without warranty of any kind. While every precaution has been taken in its preparation, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein. Visual FoxPro 9.0 SP2 was the final release of the product, and Microsoft's extended support for it ended in January 2015. ODBC drivers, OLE DB providers, SQL Server versions, and Windows security settings change over time; behavior can differ between versions and configurations. Always consult Microsoft's documentation for the versions you use, test against a copy of your data, and back up your databases before running examples that change data or schema.

Microsoft, Visual FoxPro, SQL Server, Windows, Azure, Excel, and Visual Studio are trademarks of the Microsoft group of companies. Other product names are trademarks of their respective owners. Their use in this book is for identification only and does not imply endorsement. This book is independent and is not affiliated with, sponsored by, or endorsed by Microsoft Corporation.

Brackenridge Equipment Rentals, its RentDesk application, its branches, people, customers, equipment, contracts, invoices, and all sample data in this book are fictional. Any resemblance to real companies or persons is coincidental. Email addresses use reserved example domains and phone numbers use the 555 pattern.

Preface

Many Visual FoxPro applications are still doing exactly what they were built to do: taking orders, tracking stock, printing invoices, and running a business day after day. Their forms are fast, their reports are trusted, and the people who use them know every shortcut. What often no longer fits is where the data lives. DBF tables in a shared folder were a perfect match for one office and a local network. They are a poor match for several branches connected over a VPN, for growing tables, for auditors who ask who can see what, and for other systems that want to read the same data.

SQL Server solves those problems, and Visual FoxPro was designed to talk to it. VFP 9 ships with three mature ways to work with a database server — SQL pass-through, remote views, and the CursorAdapter class — and each of them returns ordinary VFP cursors, so your forms, grids, and reports keep working with data they already understand. The hard part is not the syntax. It is knowing which driver to use and why, how data types change on the way, how to keep updates safe when several people edit the same row, how to move years of DBF data without losing a record, and how to deploy the result to every desk.

This book is a practical, verified guide to that journey. We keep the VFP front end and move the data. Every technique is applied to one realistic application, and the code in the book was run against a real SQL Server instance from a real copy of Visual FoxPro 9, with the actual output shown.

Who This Book Is For

- **Visual FoxPro developers** who maintain a DBF-based application and need to move its data to SQL Server without rewriting the whole system.
- **Teams planning a gradual modernization** who want the database on a server now, so that a later web or .NET front end can share it.
- **SQL Server DBAs and .NET developers** who must support a VFP client and want to understand what it sends to the server and why.
- **Consultants** who inherit VFP client-server code built with remote views or CursorAdapters and need to fix, secure, or speed it up.

You should know the Visual FoxPro language, the Command window, DBF tables, and SELECT-SQL. You do not need prior experience with SQL Server, ODBC, or T-SQL: we introduce each server concept from the VFP side when we need it.

What You Will Learn

- How ODBC drivers work with 32-bit VFP, how to choose one, and how to write DSN-less connection strings.
- How to connect, configure, and share connections with SQLSTRINGCONNECT, SQLCONNECT, and SQLSETPROP.

- How SQL Server types map to VFP types — Currency, dates, memos, logicals, NULLs, and Unicode — including the traps.
- How to use SQL pass-through safely: parameters, multiple result sets, prepared statements, asynchronous calls, and solid error handling with AERROR.
- How to build updatable cursors, remote views, and CursorAdapter classes, and how update conflicts are detected.
- How to use transactions, stored procedures with output parameters, identity values, sequences, rowversion concurrency, and server-side constraints.
- How to migrate DBF data to SQL Server with a repeatable script and prove that nothing was lost.
- How to make client-server VFP fast, secure, deployable, and diagnosable — and how to convert a complete module.

How This Book Is Organized

Part I — From DBF to Client-Server (Chapters 1–5) explains what changes when data moves to a server, sets up SQL Server and the sample database, and covers ODBC drivers, connections, and data type mapping. **Part II — SQL Pass-Through** (Chapters 6–10) teaches SQLEXEC in depth: parameters, injection-safe code, multiple result sets, prepared and asynchronous execution, error handling, and changing data. **Part III — Remote Views and CursorAdapter** (Chapters 11–15) covers remote views in a database container, updatable views and WhereType, the CursorAdapter class with ODBC and ADO, and choosing and binding a technique in forms. **Part IV — Server-Side Logic and Data Integrity** (Chapters 16–20) covers transactions, stored procedures, identity values and sequences, concurrency with rowversion, and moving rules to the server. **Part V — Migration, Performance, and Security** (Chapters 21–24) plans and performs the data migration, then makes the result fast and secure. **Part VI — Deployment and the Capstone** (Chapters 25–28) covers deployment, reporting, troubleshooting, and a complete module conversion. The appendices provide a cheat sheet, templates and reference tables, a production checklist, and a glossary.

Each chapter ends with a **Try It Yourself** section and a **Key Takeaways** box for quick review.

The Running Example: Brackenridge Equipment Rentals

Throughout the book we work on **RentDesk**, the rental system of **Brackenridge Equipment Rentals**, a fictional company that rents excavators, loaders, lifts, compactors, and generators to builders, landscapers, a city public-works department, and homeowners from three yards. RentDesk was written in Visual FoxPro on free DBF tables: branches, customers, equipment, rental contracts, contract lines, returns, and invoices. Now that three branches share the data over a VPN, the company moves the tables to SQL Server and keeps the VFP application. We define the target schema, migrate the data with a verified script, and rebuild data access step by step — from the first SQLEXEC call to

stored procedures that open contracts, return equipment, and create invoices with exact rental-day and tax calculations. The sample data is small enough to check by hand, so every number in the book can be verified.

Versions and Conventions

The examples target **Visual FoxPro 9.0 SP2** and **SQL Server 2022** (the free Express edition is enough); almost everything also works with recent versions of SQL Server. VFP is a 32-bit program, so it uses 32-bit ODBC drivers; our main driver is **ODBC Driver 17 for SQL Server**, and we show where the older driver that ships with Windows behaves differently. Where behavior depends on the driver or version, a Note says so and the book describes what we observed in our tests. Output is shown with US settings: `SET CENTURY ON`, dates as `03/21/2026`, and currency as `$1,876.88`.

> **TIP**
> A recommended practice or a technique that makes your work easier.

> **NOTE**
> Background information, version or driver differences, or a useful aside.

> **WARNING**
> A common mistake or something that can cause data loss, errors, or security problems.

VFP code and T-SQL appear in `monospace`; VFP and T-SQL keywords are written in uppercase. Long lines are broken with the VFP continuation character `;`. Names follow one convention throughout: cursors start with `cur`, remote views with `rv_`, CursorAdapter classes with `ca`, stored procedures with `usp_`, and variables use the VFP Hungarian style (`lcName`, `pnContrId`), summarized in Appendix B.

Contents

Preface

PART I — FROM DBF TO CLIENT-SERVER

1 Why Move a VFP Application to SQL Server: Client-Server Thinking and the RentDesk Plan

2 Setting Up SQL Server for VFP Development: Instances, Tools, and the Brackenridge Database

3 ODBC Drivers and Connection Strings: DSNs, DSN-less Connections, and 32-bit Reality

4 Connecting from VFP: SQLSTRINGCONNECT, SQLCONNECT, SQLSETPROP, and Shared Connections

5 Data Type Mapping: Currency, Dates, Memos, Logicals, NULLs, and Unicode

PART II — SQL PASS-THROUGH

6 SQL Pass-Through Fundamentals: SQLEXEC, Result Cursors, and Server-Side SQL

7 Parameterized Queries: ?Parameters, Variable Scope, and Stopping SQL Injection

8 Multiple Result Sets, SQLMORERESULTS, SQLPREPARE, and Asynchronous Execution

9 Error Handling: AERROR, ODBC Errors, and the BrConnection Class

10 Changing Data with SQL Pass-Through: INSERT, UPDATE, DELETE, and Updatable Cursors

PART III — REMOTE VIEWS AND CURSORADAPTER

11 Remote Views and Connection Definitions in a DBC

12 Updatable Remote Views: Key Fields, Update Names, WhereType, and Buffering

13 CursorAdapter Fundamentals: ODBC Data Sources, CursorSchema, and CursorFill

14 CursorAdapter in Depth: Updates, Events, ADO, and the BrCursorAdapter Base Class

15 Choosing an Access Technique and Binding Remote Data to Forms

PART IV — SERVER-SIDE LOGIC AND DATA INTEGRITY

16 Transactions: Manual Mode, SQLCOMMIT, SQLROLLBACK, and Server Transactions

17 Stored Procedures from VFP: Input, Output, and Result Sets

18 Identity Values, Sequences, and Business Numbers

19 Concurrency: Update Conflicts, rowversion, and Resolving Collisions

20 Constraints, Triggers, and Validation: Moving Rules to the Server

PART V — MIGRATION, PERFORMANCE, AND SECURITY

21 Planning the Migration: Inventory, Schema Design, and Type Decisions

22 Migrating the Data: Scripted Upsizing from DBF and Proving It Worked

23 Performance: Fetch Size, Progressive Fetching, Indexes, and Avoiding N+1 Queries

24 Security: Windows Authentication, SQL Logins, Least Privilege, and SQL Injection

PART VI — DEPLOYMENT AND THE CAPSTONE

25 Deployment: Drivers, Configuration Files, and Rolling Out the Client

26 Reports and Read-Heavy Screens Against SQL Server

27 Diagnostics and Troubleshooting: Logging, Tracing, and Common Failures

28 Capstone: Converting the RentDesk Contracts Module from DBF to SQL Server

APPENDICES

A Client-Server Cheat Sheet

B Templates and Reference

C Production Checklist

D Glossary

PART I

From DBF to Client-Server

What really changes when RentDesk's data leaves the shared folder: client-server thinking, a SQL Server instance and the Brackenridge database, 32-bit ODBC drivers and connection strings, connection handles and their settings, and how every VFP data type travels to the server and back.

1 · Why Move to SQL Server

2 · Setting Up SQL Server

3 · ODBC Drivers and Connection Strings

4 · Connecting from VFP

5 · Data Type Mapping

CHAPTER 1 · PART I — FROM DBF TO CLIENT-SERVER

Why Move a VFP Application to SQL Server: Client-Server Thinking and the RentDesk Plan

Before we write a single connection string, we need a clear picture of what changes when a Visual FoxPro application stops opening DBF files on a share and starts sending requests to a database server. This chapter introduces RentDesk, the application we convert throughout the book, explains why its file-based design struggles across three yards, and lays out the plan that the remaining chapters follow.

The Application We Are Going to Move

Brackenridge Equipment Rentals rents construction equipment: mini and compact excavators, skid steer loaders, scissor and boom lifts, plate compactors, trench rollers, and generators. The company works from three yards. The original yard is Riverside Yard in Riverside, which is also where the office and the file server live. North Depot in Kellerton opened a few years later, and Harbor Yard in Port Lindell is the newest. Each yard has a counter where customers pick up equipment, a workshop that checks it when it comes back, and one or two PCs that run the rental system.

That rental system is called **RentDesk**. It is a Visual FoxPro application compiled to `rentdesk.exe`, and it does everything the counter staff need: it keeps the customer list, tracks every piece of equipment and its status, opens rental contracts, adds equipment lines to them, records returns with the condition of each item, calculates rental days and charges, and prints invoices. It was written and is still maintained by **Nora Kessler**, the company's in-house developer. Throughout this book we work alongside Nora: we read her code, respect the decisions she made years ago, and help her move the data underneath RentDesk to Microsoft SQL Server without rewriting the parts of the application that work well.

RentDesk stores its data in free DBF tables in a single folder on the Riverside file server. On a developer PC, the same tables live in `C:\Brackenridge\Data\`, and that is the path we use in examples. The source code sits in `C:\Brackenridge\RentDesk\`: `main.prg` starts the application, `brlib.prg` holds the business rules (rental days, charges, sales tax, money formatting), and the forms and reports are part of the project `rentdesk.pjx`. Later chapters add `brdata.prg`, `brmigrate.prg`, `brca.prg`, a database container named `brremote.dbc`, and a configuration file named `rentdesk.ini`. Server-side scripts go in `C:\Brackenridge\Sql\`.

The DBF tables

RentDesk uses eight free tables. Seven of them hold business data, and one, `counters`, hands out the next contract and invoice numbers. The table below summarizes them. Field names are ten characters or fewer, which matters later: we keep exactly the same names for the SQL Server columns, so the VFP code that refers to `day_rate` or `contr_no` does not have to change.

Table	Holds	Key fields	Rows
branches	The three yards	branch_id C(3)	3
customers	Account and cash customers	cust_id I, cust_no C(6)	5
equipment	Every rentable item, its rates and status	equip_id I, asset_tag C(8)	8
contracts	Rental contracts (header)	contr_id I, contr_no C(8)	4
conlines	Equipment lines on a contract	line_id I, contr_id I	6
returns	One return record per line	return_id I, line_id I	3
invoices	One invoice per closed contract	invoice_id I, contr_id I	2
counters	Next contract and invoice numbers	name C(10)	2

The sample data is small on purpose, so that every number in the book can be checked by hand. There are five customers, from Halvorsen Builders LLC, a builder with a $15,000.00 credit limit, to Dale Okonkwo, a homeowner who pays cash. There are eight items of equipment, from the Mini Excavator 3.5 t (asset tag `BR-EX-01`, $285.00 a day) to the Trench Roller (`BR-TR-01`) that is currently in the workshop. Four contracts exist: two are closed and invoiced, and two are still open with equipment out on site.

You can create the tables yourself with the program `setup-dbf.prg`, which ships with the book's sample files. It builds the free tables, their CDX indexes, and the sample rows in whatever folder you pass to it. The program below runs it and then lists each table with its record count and field count:

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
```

Output

```
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
```

The `counters` rows tell us that the next contract will be **RC-26005** and the next invoice **IN-26003**. Keep those numbers in mind; they reappear when we replace the counters table with SQL Server sequences in Chapter 18.

How RentDesk reads its data today

Nora's code uses the classic Xbase commands that every VFP developer knows. To list the available equipment at Riverside, the equipment screen opens the table in the order of its `br_status` tag (an index on `branch_id + status`), seeks the first matching key, and scans while the key still matches:

Old RentDesk code (DBF version)

```
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
```

Output

```
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
```

This code is fast and clear on a local disk or on a fast office network. It is also very chatty in a way that is easy to miss. VFP, not a server, does all the work here: it reads the CDX header and index pages over the network, walks the index to find `RVSA`, and then reads each matching record from the DBF file. The file server only hands out blocks of bytes. It has no idea that a query is running, which rows the user wants, or which rows another user is changing at the same moment.

Why Shared DBF Files Struggle Across Three Yards

When RentDesk ran only at Riverside, every PC sat on the same office network as the file server, and nobody complained. When North Depot and Harbor Yard opened, the company connected them over a VPN, and RentDesk was started from the shared folder at all three yards. That is where the trouble began. None of the problems below is a bug in Nora's code; they are properties of a file-sharing design stretched over a wide-area network.

Every request moves raw file blocks

A file-server database moves pages of files, not answers. Opening a table reads its header. A `SEEK` reads index pages. A `SCAN FOR` without an optimizable index may read the whole table. A report that totals a month of invoices reads every invoice record across the network, adds them up on the client, and throws most of the bytes away. On a local network the cost is hidden by bandwidth and low latency. Across a VPN, each round trip takes noticeably longer, and an operation that needs hundreds of small reads becomes slow enough that counter staff notice. Rushmore optimization helps by reading index pages instead of records, but it still runs on the client and still pulls the index pages over the wire.

Locks travel through the file system, and interrupted writes do damage

VFP coordinates multi-user access with file and record locks that the file server manages on its behalf, and each client caches pieces of the files. Keeping locks and caches consistent takes many small conversations, each slower over a VPN. Worse, a DBF table is written by whichever client is updating it. If that client's connection drops in the middle of a write, the table header, a memo file, or a CDX index can be left inconsistent. Every long-time VFP developer has run `REINDEX` after an "index does not match the table" error. Nora's nightly pack-and-reindex routine works, but it is a symptom of the design rather than a cure.

Size limits per file

Each DBF, FPT, and CDX file is limited to 2 GB. RentDesk is nowhere near that today, but contract history grows every year, and working around the limit means archiving rows and teaching every report to look in two places.

No server-side security or rules

Anyone who can open the shared folder can copy the tables, open them in another program, or change them with a few lines of code that bypass RentDesk's validation. Free tables have no rules of their own; all the checks live in the application. A database container can add field and row rules and triggers, but they are still enforced by whichever VFP process opens the files, not by a server that every client must go through.

Backups need quiet files

A reliable copy of a set of DBF files needs a moment when nobody is writing to them, which in practice means asking everyone to exit RentDesk. There is no transaction log to replay and no way to restore the data to the state it had at 10:15 this morning.

> **NOTE**
>
> None of this means DBF tables are a poor design. For a single office on a local network they are fast, simple, and remarkably robust, and many VFP applications have run on them for decades. The problems appear when the network between the client and the files becomes slow or unreliable, or when the business needs security, auditing, and backups that a file share cannot provide.

What Client-Server Changes

In a client-server design, one program on one machine owns the data files: the database server. Clients never open those files. Instead, they connect to the server and send it requests written in SQL. The server reads its own files on its own disks, does the filtering, joining, sorting, and totaling, and sends back only the results. For RentDesk the server is Microsoft SQL Server, and the requests are written in Transact-SQL (T-SQL), SQL Server's dialect of SQL.

Here is the Riverside availability question again, asked of SQL Server through a stored procedure that is part of our schema. The VFP code opens a connection, sends one request, and receives a cursor with the answer:

Client-server version (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)
```

Output

```
         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
```

The same three items come back, this time sorted by asset tag because the procedure says `ORDER BY e.asset_tag`. `SQLEXEC()` returned 1, meaning one result set, and `curAvail` is an ordinary VFP cursor: you can `SCAN` it, bind it to a grid, or run a local `SELECT` against it. What crossed the network was one request and three rows, not index pages and records. Do not worry about the details yet; Chapters 3 and 4 cover the connection string and the handle, and Chapter 6 covers `SQLEXEC()`.

Three ideas sum up the change, and the rest of the book keeps returning to them.

Ask for sets, not records

File-based code navigates: open, seek, skip, test, skip again. Client-server code describes the rows it wants and lets the server find them, asking for exactly the rows and columns a screen needs in one round trip. Loading a whole table "just in case" is a common reason a converted application feels slow (Chapters 6, 15, and 23).

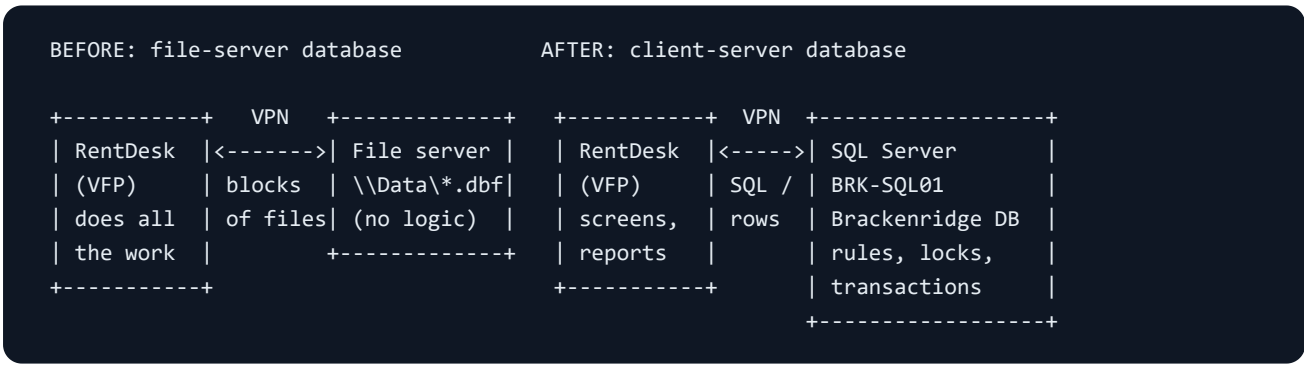
The server enforces the rules

SQL Server checks primary keys, unique values, foreign keys, allowed status codes, and default values for every client, whether the change comes from RentDesk, a report tool, or an administrator's script. It wraps related changes in transactions so that a contract line and the equipment status change together or not at all. RentDesk still validates input to give friendly messages, but the server has the last word.

Only results cross the network

Because the server does the work next to its own disks, the traffic between a yard and the server is the size of the request plus the size of the answer. A monthly revenue total is one row. That is what makes the design suitable for a WAN: the number of round trips and the size of the results matter, not the size of the tables.

Before and after



> **TIP**
>
> When you are unsure how to structure a piece of converted code, ask yourself how many times it talks to the server and how many rows it brings back. Fewer round trips and fewer rows almost always win. This one question explains most of the performance advice later in the book.

What Stays in Visual FoxPro

Moving the data does not mean rewriting the application. RentDesk's forms, menus, reports, and most of its business code stay in VFP. The counter staff keep the screens they know. VFP remains an excellent tool for building data-entry forms and reports against cursors, and a cursor filled from SQL Server behaves like any other cursor once it is on the client.

- **Forms and grids** stay. Instead of binding to tables opened with `USE`, they bind to cursors filled from the server (Chapter 15).
- **Reports** stay. They run from a cursor that one request gathered on the server (Chapter 26).
- **Business functions** in `brlib.prg` stay. `RentalDays()`, `RentalCharge()`, `CalcTax()`, and `MoneyStr()` remain useful for previews on screen, and the same rules exist on the server as T-SQL functions so that both sides agree.
- **The executable and its deployment** stay, with two additions on each PC: a 32-bit ODBC driver for SQL Server and a small configuration file that says which server to use (Chapter 25).

What changes is the data-access layer: every `USE`, `SEEK`, `SCAN`, `REPLACE`, `APPEND BLANK`, `RLOCK()`, and `FLOCK()` that touches the shared tables. In RentDesk those commands are spread across forms and procedures, which is typical of applications that grew over many years. Part of the work is finding them all, which Chapter 21 turns into an inventory, and part is replacing them with a small number of well-tested patterns. The record-locking code goes away entirely, because SQL Server handles locking itself, and the `counters` table with its lock-increment-unlock routine is replaced by server sequences.

> **NOTE**
>
> If you come from .NET and ADO.NET, think of a VFP cursor filled by `SQLEXEC()` as a disconnected `DataTable` that the VFP language can query, index, and bind directly. The connection handle plays the role of a `SqlConnection`, and an updatable cursor is roughly what a `DataAdapter` with update commands gives you.

Three Ways VFP Talks to SQL Server

Visual FoxPro 9 gives us three built-in techniques for working with remote data. They all use the same connection underneath (ODBC in most of this book, with ADO as an option for CursorAdapter), and they all produce ordinary VFP cursors. They differ in how much VFP does for you and how much control you keep.

Technique	What it is	Strengths	Chapters
SQL pass-through (SPT)	Functions such as `SQLEXEC()` send SQL text you write	Full control, any T-SQL, stored procedures, fewest surprises	6–10, 16–20
Remote views	Views stored in a DBC that VFP turns into SQL	Familiar `USE`, easy updatable lists, View Designer	11–12

CursorAdapter	A class that fills and updates one cursor	Reusable classes, events, ODBC or ADO, works in code	13–14

We start with SQL pass-through because it is the foundation: remote views and CursorAdapters generate the same kind of requests that SPT sends explicitly, and when something goes wrong you diagnose it with SPT. RentDesk ends up using all three. Ad-hoc reads and stored-procedure calls use SPT through a small connection class named `BrConnection`, which we build in Chapter 4 and complete in Chapter 9. A few simple editable lists use remote views. The contract screen uses CursorAdapter classes. Chapter 15 compares the three side by side and explains which to use when.

The Target: The Brackenridge Database

On SQL Server, the data lives in a database named **Brackenridge**. On a developer PC it runs on a local SQL Server Express instance named `.\SQLEXPRESS`; in production it runs on a server that Victor, the company's IT contractor, calls `BRK-SQL01`. The tables keep the DBF names and field names, so that `equipment.day_rate` means the same thing on both sides. The diagram shows the seven tables and how they relate:

Brackenridge schema (dbo)



Read the diagram from the branches outward: each branch owns many equipment items and many contracts; each customer has many contracts; each contract has many lines; each line refers to one item of equipment and has at most one return; and each contract has at most one invoice. A few things are new compared with the DBFs, and each gets its own chapter:

- **IDENTITY keys.** `cust_id`, `equip_id`, `contr_id`, and the other integer keys are generated by the server. During the migration the existing DBF values are kept, so customer 1 is still customer 1 (Chapters 18 and 22).
- **Constraints.** Primary keys, unique keys such as `uq_equipment_asset_tag`, foreign keys such as `fk_contracts_customers`, and checks such as `ck_equipment_status` (A, R, or M) are declared on the server and enforced for everyone (Chapter 20).
- **rowversion columns.** `customers`, `equipment`, and `contracts` have a `row_ver` column that SQL Server changes automatically on every update. It is the basis of our conflict detection in Chapter 19.
- **Sequences.** `dbo.seq_contract` and `dbo.seq_invoice` replace the `counters` table, so two yards can open contracts at the same moment without fighting over a lock.
- **Server-side logic.** Two functions, `dbo.fn_RentalDays` and `dbo.fn_RentalCharge`, implement the same rules as `brlib.prg`, and six stored procedures handle the main workflow: opening a contract, adding a line, returning equipment, creating an invoice, reading a contract, and listing the available equipment.

A preview of the type decisions: customer names are `nvarchar(40)`, money columns are `money` (VFP's Currency), DateTime fields are `datetime2(0)`, dates are `date`, logicals are `bit`, and memos are `varchar(max)`. Chapter 5 shows what each becomes in a VFP cursor, including one `varchar(max)` surprise.

The business rules, briefly

The rules are worth stating once here, because the numbers that prove our code correct all follow from them. Rental days are counted in started 24-hour periods after a 60-minute grace period, with a minimum of one day. The charge is the number of full weeks times the weekly rate, plus the leftover days times the daily rate, with the leftover part capped at one weekly rate. Sales tax is 7.25% of charges plus damage, rounded to cents, and zero for tax-exempt customers such as the City of Kellerton. The rates are copied onto the contract line when equipment goes out, so a price change does not affect rentals already on site. For example, the mini excavator on contract RC-26001 went out on 03/02/2026 at 07:30 and came back on 03/09/2026 at 16:10: that is 8 rental days, charged as one week at $1,140.00 plus one day at $285.00, for $1,425.00.

The People

Four people appear in the book. All of them, like the company, are fictional.

- **Gail Brackenridge** is the owner and general manager. She asks the business questions: why the North Depot counter waits for the screen, what the yards earned last month, and whether the move is worth the money.

- **Nora Kessler** is the VFP developer who wrote RentDesk. She knows every form and every quirk of the data. She is new to SQL Server, which makes her the perfect colleague for this book: we explain client-server ideas to her from the VFP side.
- **Tomas Reyes** manages Riverside Yard. He checks returned equipment and records its condition; his name appears in the `checked_by` column of every return in the sample data.
- **Victor Lam** is the external IT contractor. He installs and administers SQL Server, manages Windows accounts and groups, and owns backups and security. Nora writes the application; Victor decides who may connect and how the server is protected.

The split between Nora and Victor is deliberate: it helps to know which decisions belong to the application and which belong to the server administrator.

The Migration Roadmap

A migration like this succeeds when it is done in small, verifiable steps, with the old system still working until the new one is proven. The book follows the order in which we recommend doing the work.

1. **Part I — From DBF to Client-Server (Chapters 1–5).** Install and understand the server, create the `Brackenridge` database, choose a 32-bit ODBC driver, connect from VFP, and learn how every SQL Server type arrives in a cursor.
2. **Part II — SQL Pass-Through (Chapters 6–10).** Read and change data with `SQLEXEC()`, use parameters safely, handle several result sets and long-running queries, catch and translate errors, and make pass-through cursors updatable. `BrConnection` is finished here.
3. **Part III — Remote Views and CursorAdapter (Chapters 11–15).** Build remote views in `brremote.dbc`, then CursorAdapter classes in `brca.prg`, and bind them to forms.
4. **Part IV — Server-Side Logic and Data Integrity (Chapters 16–20).** Transactions, stored procedures, identity values and sequences, update conflicts, constraints, and triggers.
5. **Part V — Migration, Performance, and Security (Chapters 21–24).** Plan and script the data migration with `brmigrate.prg`, prove it moved every row correctly, tune round trips and indexes, and lock the server down to least privilege.
6. **Part VI — Deployment and the Capstone (Chapters 25–28).** Roll out drivers and configuration, adapt reports, troubleshoot, and finally convert the whole contracts module from DBF code to SQL Server.

Notice that the data migration itself is not first. The sample database we use from Chapter 2 onward is precisely the state right after a migration of the sample DBFs (Chapter 22 proves it row by row), so we learn the access techniques before we move real data.

> **WARNING**
>
> Do not plan a "big bang" weekend where the data moves, every screen is rewritten, and the new version goes live on Monday. Convert one module at a time, test it against a copy of the real data, and keep a way back. The capstone in Chapter 28 converts a single module for exactly this reason.

An Honest Word About Visual FoxPro's Future

Visual FoxPro 9.0 was released in December 2004, and Service Pack 2 followed in 2007. In March 2007 Microsoft announced that there would be no Visual FoxPro 10, so VFP 9 (SP2) is the final release. Microsoft's extended support ended in January 2015. VFP is a 32-bit application, which has practical consequences we meet in Chapter 3: it can only load 32-bit ODBC drivers and OLE DB providers. Community projects on VFPX continue to publish tools and libraries, but they are not Microsoft products, and you should evaluate each one on its own merits.

Many companies keep VFP applications in daily use regardless, because they work and a rewrite is expensive and risky. Moving the data to SQL Server is sound in that situation: it fixes today's problems (speed across the WAN, corruption risk, security, backups), and once the data, rules, and workflow live on the server, a future front end in any language can use them unchanged.

> **TIP**
>
> When you decide where a piece of logic goes, prefer the server for anything that must be true for every client (numbering, status changes, totals that end up on an invoice), and keep VFP for what belongs to the user interface (formatting, previews, navigation). That split serves RentDesk now and any replacement later.

Setting Up the "Before" State

Everything in this book can be reproduced on one Windows PC. You need Visual FoxPro 9.0 SP2, SQL Server 2022 Express (or any edition; everything shown also works on recent versions of SQL Server unless a Note says otherwise), and the 32-bit ODBC Driver 17 for SQL Server. Chapter 2 installs and configures the server side; Chapter 3 covers the driver. For now, create the folders and the original DBF tables so that you can see the application's starting point:

Command window

```
MD C:\Brackenridge
MD C:\Brackenridge\Data
MD C:\Brackenridge\RentDesk
MD C:\Brackenridge\Sql
CD C:\Brackenridge\RentDesk
DO setup-dbf WITH "C:\Brackenridge\Data"
```