IBM Db2
in Practice

A Complete Guide to Db2 for Developers and DBAs
From your first query to a secure, tuned, highly available production database

First Edition

THENEXBOOKS

IBM Db2 in Practice
A Complete Guide to Db2 for Developers and DBAs, from First Query to Production

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 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. Db2 and its tools evolve; commands, defaults, and configuration parameters can change between versions and fix packs. Always consult the official documentation for the version you run, and test administrative procedures — especially recovery — on a non-production system first.

IBM and Db2 are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Linux is a registered trademark of Linus Torvalds. Windows is a trademark of the Microsoft group of companies. Java and Oracle are registered trademarks of Oracle and/or its affiliates. Docker is a trademark of Docker, Inc. Kubernetes is a registered trademark of The Linux Foundation. PostgreSQL is a trademark of the PostgreSQL Community Association of Canada. 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 IBM.

Halverson Mutual, the insurer used as the running example in this book, is fictional, as are its branches, agents, customers, policies, claims, and data.

Preface

Db2 runs a remarkable share of the world's serious transaction processing. Banks, insurers, retailers, airlines, and governments rely on it for the systems that simply cannot lose data: payments, policies, reservations, ledgers. Many developers meet Db2 for the first time when they join one of those organizations and discover that the application they now own talks to a Db2 database. Many new DBAs inherit one.

Db2 rewards the people who understand it. It has a mature cost-based optimizer, strong concurrency control, first-class backup and recovery, high availability built in, fine-grained security, and analytics features that sit right next to transactional data. But it also has its own vocabulary — instances, table spaces, buffer pools, the CLP, RUNSTATS, HADR — and documentation written for people who already speak it.

This book is the guide we would have wanted on day one. We start from zero, install Db2, and build one real database step by step. Each idea appears at the moment it solves a problem the business actually has. By the end, that database is modeled, programmed, tuned, secured, backed up, and running with a hot standby — and you will understand not only *which* command to type, but *why*.

Who This Book Is For

- **Developers** who need to write SQL, stored procedures, and application code against Db2, whether or not they know another database.
- **New and accidental DBAs** who have inherited a Db2 system and need to keep it fast, safe, and recoverable.
- **Experienced DBAs from other platforms** — Oracle, SQL Server, PostgreSQL, MySQL — who need a fast, accurate map of how Db2 does things.
- **Students and career changers** who want a database skill that is in steady demand in large enterprises.

What You Will Learn

- How Db2 is organized — instances, databases, table spaces, buffer pools, logs, and the catalog — and how to install it, including in a container.
- Db2's SQL from the ground up: queries, joins, aggregation, window functions, CTEs, MERGE, identity columns, sequences, temporal tables, JSON, and XML.
- Views, triggers, SQL PL stored procedures and functions, transactions, isolation levels, and locking.
- Connecting applications from Java, Python, Node.js, and .NET.
- Indexes, the optimizer, statistics, EXPLAIN, query tuning, and monitoring with MON_GET functions and db2pd.

- Storage, compression, partitioning, data movement, logging, backup and recovery, security with row and column access control, and routine maintenance.
- High availability with HADR, column-organized tables for analytics, containers, upgrades, and migration to Db2.

How This Book Is Organized

Part I — Getting Started (Chapters 1–5) installs Db2, explains its architecture and tools, and teaches the basics of querying. **Part II — Data Modeling and SQL** (Chapters 6–10) designs the Halverson schema and covers Db2's SQL in depth. **Part III — Programming Db2** (Chapters 11–15) covers views, triggers, SQL PL, transactions and locking, and application connectivity. **Part IV — Performance** (Chapters 16–19) covers indexes, the optimizer, tuning, and monitoring. **Part V — Administration and Security** (Chapters 20–24) covers storage, data movement, backup and recovery, security, and maintenance. **Part VI — Production** (Chapters 25–28) covers HADR, analytics, containers and upgrades, and a complete production case study. The appendices provide a cheat sheet, templates, a production checklist, and a glossary.

Each chapter ends with exercises and a **Key Takeaways** box for quick review.

The Running Example: Halverson Mutual

Throughout the book we build and run the database for **Halverson Mutual**, a fictional regional insurer with a head office in Maple Falls, Ohio, and branches in Birchwood, Ohio, and Lakeview, Indiana. Halverson sells auto, home, renters, and life policies through its agents, collects premiums, and handles claims. The database, `HALVDB`, starts with a handful of customers, policies, and claims you can see on one screen, and grows into a production system with audit trails, stored procedures, tuned queries, row-level security, scheduled backups, and a hot standby.

Versions and Conventions

Examples target **Db2 for Linux, UNIX, and Windows 12.1**, and we point out where Db2 11.5 differs. Almost everything also applies to 11.5, and much of the SQL applies to Db2 for z/OS and Db2 for i, which are separate products with their own administration. Db2 evolves through versions, mod packs, and fix packs; where a detail may differ in your version, a Note points you to the official documentation rather than guessing. Query results in this book are computed from the sample data you load in Chapter 6, so you can check them yourself; performance and monitoring output is shown as abridged examples, because your numbers will depend on your system.

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

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

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

SQL keywords and Db2 object names appear in UPPERCASE, as Db2 stores them, for example `HALV.POLICY`. File names above code blocks, such as `sql/halv-schema.sql`, tell you where a script lives. Terminal commands beginning with `$` are run as the instance owner (`db2inst1`) unless noted.

Contents

Preface

PART I — GETTING STARTED

1 Why Db2? Editions, Platforms, and Where Db2 Fits

2 Installing Db2: Community Edition, Containers, and Your First Database

3 Db2 Architecture: Instances, Databases, Table Spaces, and Buffer Pools

4 Working with Db2: The Command Line Processor, Scripts, and GUI Tools

5 SQL Fundamentals in Db2: Querying, Filtering, Sorting, and Functions

PART II — DATA MODELING AND SQL

6 Designing Tables: Data Types, Keys, and Constraints

7 Joins, Subqueries, and Set Operations

8 Aggregation, OLAP Functions, and Common Table Expressions

9 Changing Data: INSERT, UPDATE, DELETE, MERGE, Identity Columns, and Sequences

10 Dates, Times, Temporal Tables, JSON, and XML

PART III — PROGRAMMING DB2

11 Views, Triggers, and Generated Columns

12 SQL PL: Stored Procedures and User-Defined Functions

13 Transactions, Isolation Levels, and Locking

14 Connecting Java Applications with JDBC

15 Connecting from Python, Node.js, and .NET

PART IV — PERFORMANCE

16 Indexes: How Db2 Finds Your Data

17 The Optimizer, Statistics, and EXPLAIN

18 Query Tuning in Practice

19 Monitoring Db2: MON_GET Functions, db2pd, and Event Monitors

PART V — ADMINISTRATION AND SECURITY

20 Storage: Table Spaces, Compression, and Table Partitioning

21 Moving Data: EXPORT, IMPORT, LOAD, and db2move

22 Logging, Backup, and Recovery

23 Security: Authentication, Authorities, Privileges, and Row and Column Access Control

24 Routine Maintenance: RUNSTATS, REORG, Configuration, and Automation

PART VI — PRODUCTION

25 High Availability with HADR

26 Analytics with Column-Organized Tables

27 Db2 in Containers and the Cloud, Upgrades, and Migrating to Db2

28 Case Study: Running HALVDB in Production

APPENDICES

A Db2 SQL and CLP Cheat Sheet

B Templates and Reference

C Production Checklist

D Glossary

PART I

Getting Started

What Db2 is and where it fits, installing it on a server or in a container, how instances, databases, table spaces, and buffer pools fit together, the tools you will use every day, and your first queries against Halverson Mutual's data.

1 · Why Db2?

2 · Installing Db2

3 · Db2 Architecture

4 · The CLP, Scripts, and GUI Tools

5 · SQL Fundamentals

CHAPTER 1 · PART I — GETTING STARTED

Why Db2? Editions, Platforms, and Where Db2 Fits

Before we install anything, we need a clear picture of what Db2 is, which member of the Db2 family this book covers, which edition you should start with, and what kind of work Db2 is built to do. We also meet Halverson Mutual, the fictional insurer whose database we will design, program, tune, secure, and run in production over the next twenty-seven chapters.

What Db2 Is

IBM Db2 is a relational database management system (RDBMS). Like PostgreSQL, Oracle Database, Microsoft SQL Server, or MySQL, it stores data in tables made of rows and columns, enforces rules about that data with keys and constraints, and lets applications read and change the data with SQL. It also provides the machinery that a serious database needs behind the scenes: a transaction manager that guarantees all-or-nothing units of work, a lock manager that keeps concurrent users from corrupting each other's changes, a write-ahead log that makes committed changes survive crashes, a cost-based optimizer that turns declarative SQL into efficient access plans, and a set of utilities for backup, recovery, loading, and reorganizing data.

If you know another relational database, most of that knowledge transfers: joins, `GROUP BY`, primary keys, and check constraints mean what you expect. What differs is the vocabulary around the engine (instances, table spaces, buffer pools, the Command Line Processor), some dialect details such as `FETCH FIRST 10 ROWS ONLY` and the `VALUES` statement, and the administrative model. This book explains those differences from the ground up.

Db2 has been developed by IBM for decades. It is mature, and it also carries history: older commands and monitoring interfaces still exist alongside their modern replacements. We teach the modern way first and point out the legacy style so you can recognize it in existing systems.

Why teams choose Db2

People rarely pick a database in a vacuum. In practice, teams work with Db2 for a few recurring reasons:

- **It is already there.** Many organizations, particularly in insurance, banking, retail, government, and manufacturing, run core systems on Db2. If you join such a team as a developer or an accidental DBA, Db2 is simply the database you need to learn.

- **Reliability features are built in.** High availability disaster recovery (HADR), online backup, point-in-time recovery, and online reorganization are part of the product rather than add-ons you assemble yourself.
- **Mixed workloads on one engine.** Db2 can store ordinary row-organized tables for transaction processing and column-organized tables for analytics in the same database, so reporting queries do not always require a separate warehouse.
- **Strong SQL and procedural support.** Db2's SQL dialect is rich: common table expressions, OLAP window functions, `MERGE`, temporal tables, XML and JSON functions, and SQL PL for stored procedures, functions, and triggers.
- **Compatibility with other databases.** Db2 includes optional compatibility features that make it accept some Oracle and MySQL syntax, which can ease migrations. We touch on these in Chapter 27.
- **A free starting point.** Db2 Community Edition is free to download and use within its resource limits, so you can learn, develop, and even run small workloads without a license purchase.

None of this makes Db2 the right answer for every project. Our goal is narrower: when Db2 is the database in front of you, you should be able to use it well.

The Db2 Family

"Db2" is a brand that covers several related but separate products. They share a name, a common heritage, and a large overlap in SQL, but they run on different platforms, have different internals, and are administered differently. Knowing which one you are working with is the first thing to establish, because documentation, commands, and even error behavior differ between them.

Product	Runs on	Where you meet it	Covered in this book?
Db2 for Linux, UNIX, and Windows (Db2 LUW)	Linux, AIX, Windows; containers	Application databases, data marts, packaged applications, cloud deployments	Yes: this is the book's subject
Db2 for z/OS	IBM Z mainframes running z/OS	Large enterprise systems of record in banking, insurance, government	Orientation only
Db2 for i	IBM Power servers running IBM i	Integrated database of the IBM i operating system	Orientation only
Db2 Warehouse and managed cloud offerings	Containers, IBM Cloud and other clouds	Analytics and managed database services	Mentioned where relevant

Db2 for Linux, UNIX, and Windows

Db2 LUW is the distributed-systems edition of Db2 and the focus of this entire book. It runs on Linux (on x86-64 and on other hardware architectures that IBM supports), on IBM's AIX UNIX, and on Windows. It also runs in containers, which is how we will install it in Chapter 2. The exact list of supported operating system versions and processor architectures changes from release to release, so always check IBM's system requirements for the version you plan to use rather than relying on memory or on an old blog post.

This book targets **Db2 LUW 12.1**. Because many production systems still run **Db2 11.5**, we add notes wherever behavior or syntax differs in a way that matters. When you see "recent versions" in the text, it means a feature or behavior that exists in current releases but whose exact arrival version we do not want to state from memory; the documentation for your version is the authority.

Db2 for z/OS and Db2 for i

Db2 for z/OS runs on IBM mainframes and is a separate code base with its own installation, utilities, and operational culture. SQL written for Db2 LUW often runs there with few changes, but the administration chapters of this book do not apply to the mainframe.

Db2 for i is built into the IBM i operating system rather than installed separately. SQL knowledge carries over well; administration does not.

> **NOTE**
>
> If you search the web for help with a Db2 problem, check which platform an answer is about. A tip that is perfect for Db2 for z/OS can be meaningless or harmful on Db2 LUW. IBM's documentation is organized by product and version; make sure you are reading the Db2 for Linux, UNIX, and Windows pages for your release.

Warehouse and cloud offerings

IBM also packages Db2 technology for analytics and managed operation: Db2 Warehouse and managed Db2 services in IBM Cloud and, in recent years, on other clouds. Names and packaging here have changed over time, so check IBM's current catalog. The engine is Db2 LUW technology, so the SQL, design, programming, and tuning in this book apply; what changes is who handles patching, backup, and failover.

Editions of Db2 LUW

Db2 LUW itself comes in editions that share one code base but differ in licensing terms, resource limits, and which advanced features you are entitled to use. The engine you install is essentially the same; the license you apply determines what you may do with it. Edition names and entitlements have been reorganized more than once, so treat the following as an orientation and confirm the details against IBM's current terms before you make a purchasing or production decision.

Edition	Cost	Typical use
Db2 Community Edition	Free	Learning, development, testing, small production workloads within its limits
Db2 Standard Edition	Licensed	General production workloads on moderate hardware
Db2 Advanced Edition	Licensed	Large production systems needing the full set of advanced features

Db2 Community Edition

Community Edition is the edition we use in this book. It is free to download, it includes the same SQL engine as the paid editions, and it is available both as a traditional installer and as a container image. The trade-off is that it is limited in the amount of processor and memory resources it will use, and it does not come with IBM's paid support. Those limits are generous enough for learning, for building and testing applications, and for many small workloads, but they are deliberately below what a large production system needs. The exact limits and conditions have changed over time, so we do not quote numbers here: check IBM's current terms for Community Edition.

Because Community Edition is the same engine, everything you learn on it transfers directly to the paid editions. When your application moves from a laptop to a licensed production server, you do not rewrite SQL or relearn administration; you apply a different license and size the hardware accordingly.

Standard and Advanced editions

The paid editions raise or remove the resource limits and include IBM support; Advanced Edition adds entitlement to features aimed at large systems. Exactly which features each edition includes changes between releases, so when we use a feature that may need a particular edition, we say so and suggest checking your entitlement.

> **TIP**
>
> You can see which license is applied to an installation with the `db2licm -l` command. Running it early, before you build anything important, avoids surprises when you later try to use a feature your edition does not include.

Typical Db2 Workloads

Databases are tuned and designed differently depending on the work they do. Db2 is used for three broad kinds of workload, and Halverson Mutual will exercise all three.

Online transaction processing (OLTP)

OLTP systems handle many short transactions from many users at once: an agent issues a policy, a customer pays a premium online, a claims handler changes the status of a claim. Each transaction touches a small number of rows, must complete quickly, and must be completely reliable. OLTP

design favors normalized tables, carefully chosen indexes, short transactions, and well-understood locking behavior. Db2's row-organized tables, its lock manager, its transaction log, and its isolation levels are the core tools here, covered in Parts II and III.

Analytics

Analytic queries scan large amounts of data to answer business questions: total claims paid by product and quarter, loss ratios by branch, premium trends over several years. These queries read many rows but only a few columns, and they aggregate heavily. Db2 supports analytics with column-organized tables (marketed as BLU Acceleration), compression, range partitioning, and a parallel query engine. Chapter 26 builds a column-organized claims fact table for exactly this purpose.

Mixed workloads

Many real systems do both: the insurer that needs fast, safe policy updates also wants dashboards that do not wait for an overnight extract. Db2 lets row-organized and column-organized tables live side by side, and workload management keeps reporting from starving transactions. We revisit this tension in the performance and production chapters.

Distinctive Db2 Features at a Glance

Every RDBMS has its strengths. Here are the Db2 LUW features you will meet in this book that newcomers from other databases most often find unfamiliar or particularly useful. Do not worry about the details yet; each gets a full treatment later.

- **Instances and the Command Line Processor.** A Db2 *instance* is an independent database server environment that can host several databases. You administer it largely through the `db2` command, the Command Line Processor (CLP). Chapters 3 and 4.
- **Automatic storage and table spaces.** Db2 places tables in table spaces, which draw space from storage groups. With automatic storage, Db2 manages the files for you. Chapters 3 and 20.
- **SQL PL.** Db2's procedural SQL language for stored procedures, user-defined functions, and triggers, based on the SQL standard's persistent stored modules. Chapters 11 and 12.
- **Currently committed semantics.** Under the default isolation level, readers in recent versions do not wait for writers by default; they see the last committed version of a row instead. Chapter 13.
- **Temporal tables.** Db2 can keep the full history of a table automatically and let you query it "as of" any past time. Chapter 10.
- **pureXML and JSON functions.** Native XML storage and querying, plus SQL functions for working with JSON. Chapter 10.
- **The MON_GET monitoring functions.** Table functions that expose detailed, low-overhead performance metrics you can query with ordinary SQL. Chapter 19.
- **HADR.** Built-in log shipping to one or more standby databases, with several synchronization modes and planned or forced takeover. Chapter 25.

- **Row and column access control.** Fine-grained security rules that filter rows and mask column values for every query, regardless of which application issues it. Chapter 23.

A first taste of Db2 SQL

Here is a small example to show that Db2 SQL will feel familiar. This query lists the five most recently started policies. It uses `FETCH FIRST`, the SQL-standard way to limit rows that Db2 has supported for a long time. Db2 can also accept `LIMIT`, depending on version and compatibility settings, but we use the standard form throughout the book.

SQL

```
SELECT POLICY_NO, PRODUCT, START_DATE
FROM HALV.POLICY
ORDER BY START_DATE DESC
FETCH FIRST 5 ROWS ONLY;
```

Another Db2 feature that may be new to you is the `VALUES` statement, which evaluates expressions without needing a table. It is the quickest way to test a function or check a special register:

SQL

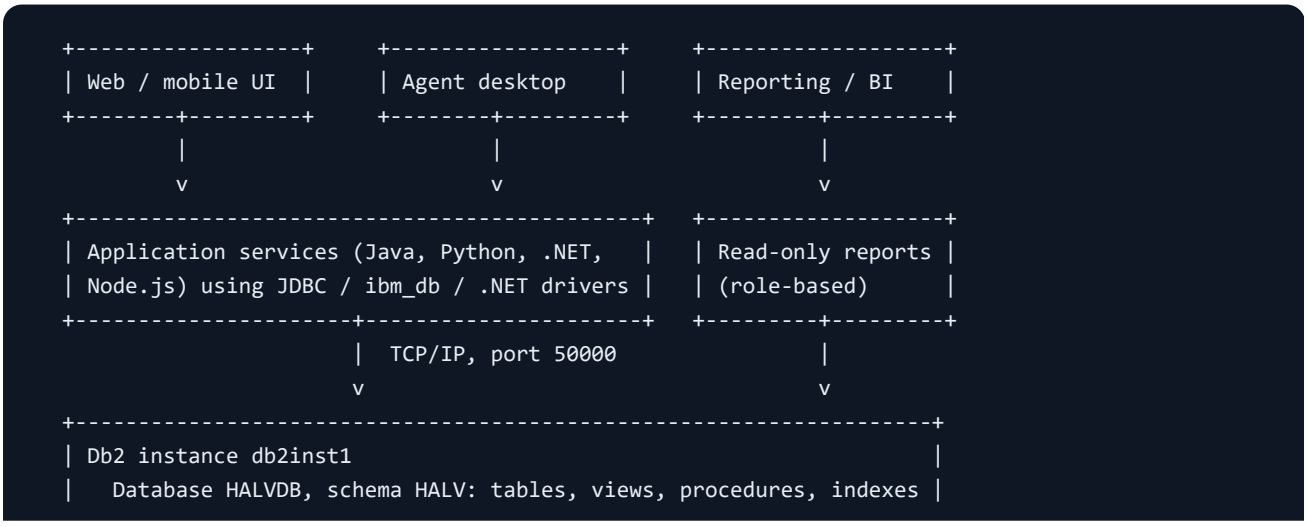
```
VALUES CURRENT DATE + 30 DAYS;
```

The expression `CURRENT DATE + 30 DAYS` uses a *labeled duration*, Db2's readable syntax for date arithmetic. If you are coming from a database that uses functions such as `DATEADD` or interval literals, this is one of the small dialect differences that Chapter 5 and Chapter 10 cover in detail.

Where Db2 Fits in Your Architecture

A database rarely stands alone. In a typical modern deployment, Db2 sits behind one or more application services and next to a set of supporting tools. It is worth sketching this picture now, because each part of the sketch corresponds to a part of the book.

Diagram



```
|   Transaction logs  ->  archive  ->  backups                    |
+---------------------------------+-------------------------------+
                                  |  HADR log shipping
                                  v
                     +----------------------------+
                     | Standby database (HADR)    |
                     +----------------------------+
```

Applications connect over TCP/IP through a driver (Chapters 14 and 15); the instance hosts the database (Chapters 3 and 6); logs feed backup and recovery (Chapter 22); a standby protects against server loss (Chapter 25); and reporting users connect through limited roles (Chapter 23).

Db2 compared with other databases

It helps to map Db2 terms onto what you may already know. The mapping is approximate; similar names do not always mean identical behavior, and later chapters explain the details.

Concept	Db2 LUW	PostgreSQL	Oracle	SQL Server
Server environment	Instance	Cluster	Instance	Instance
Namespace for objects	Schema	Schema	Schema (user)	Schema
Storage container for tables	Table space	Tablespace	Tablespace	Filegroup
Data cache	Buffer pool	Shared buffers	Buffer cache	Buffer pool
Procedural language	SQL PL	PL/pgSQL	PL/SQL	T-SQL
Command-line client	CLP (`db2`), CLPPlus	`psql`	SQL*Plus, SQLcl	`sqlcmd`
Data dictionary	`SYSCAT` views	`pg_catalog`	Data dictionary views	`sys` views

One difference deserves early emphasis. A Db2 *database* is a self-contained unit: it has its own transaction logs, configuration, catalog, buffer pools, and table spaces, and it is backed up and recovered as a whole. An application connects to one database at a time. If you come from MySQL, where "database" and "schema" are near synonyms, note the shift: in Db2 you usually have one database per application and organize objects with schemas inside it.

Meet Halverson Mutual

Learning a database from disconnected snippets is hard. Instead, this book follows one realistic, small, and consistent example from beginning to end. Halverson Mutual is a fictional regional insurer that sells property and casualty insurance (auto, home, renters) and life insurance. It has three branch offices, a handful of agents, and a growing book of customers, policies, claims, and premium payments. It is small enough to see every row, yet rich enough to exercise joins, constraints, transactions, procedures, security rules, and production operations.

The company and everyone in it are invented; any resemblance to real people or organizations is coincidental.

The business

Halverson Mutual runs from three offices. Its head office is in Maple Falls, Ohio; a second Ohio branch is in Birchwood; and its newest office is in Lakeview, Indiana. Agents belong to a branch and sell policies to customers. A policy covers one product for a period, usually a year (life policies run much longer), and carries an annual premium. Customers pay premiums by card, bank transfer (ACH), or check. When something goes wrong, a customer reports a claim against a policy; the claim moves through a workflow from open to review, approval or denial, payment, and closure.

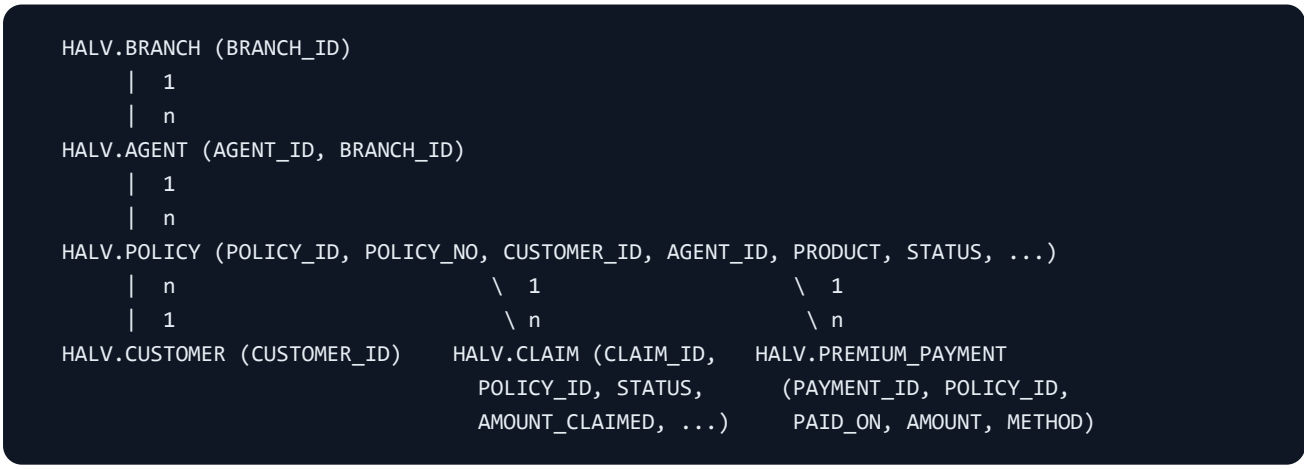
Branch ID	Code	Name	City	State
1	MPF	Maple Falls Head Office	Maple Falls	OH
2	BRW	Birchwood	Birchwood	OH
3	LKV	Lakeview	Lakeview	IN

Four agents work across the branches: Hannah Cole and Marcus Bell at the Maple Falls head office, Priya Raman in Birchwood, and Tomas Herrera in Lakeview. Five customers hold seven policies between them. Olivia Grant has auto and home cover; Ethan Brooks has auto; Sofia Martinez holds a long-term life policy; Liam Chen's renters policy has lapsed; and Ava Patel has a home policy and a cancelled auto policy.

The data model

The Halverson database is called `HALVDB`, and all application objects live in the schema `HALV`. The core model has six tables. Chapter 6 creates them with full data types, keys, and constraints; for now, the shape is what matters.

Diagram



Read the diagram as follows: a branch has many agents; an agent sells many policies; a customer holds many policies; and each policy can have many claims and many premium payments. Each table has a surrogate primary key, and policies and claims also carry human-readable business numbers such as `HM-AU-000001` and `CL-26-000001` that staff quote on the phone.

Table	Holds	Sample rows
`HALV.BRANCH`	Branch offices	3
`HALV.AGENT`	Agents and their branch	4
`HALV.CUSTOMER`	Policyholders	5
`HALV.POLICY`	Policies with product, status, dates, premium	7
`HALV.CLAIM`	Claims against policies and their workflow status	5
`HALV.PREMIUM_PAYMENT`	Premium payments received	6

Policy status is a single letter: `A` for active, `L` for lapsed, and `C` for cancelled. Products are four-letter codes: `AUTO`, `HOME`, `LIFE`, and `RENT`. Claim status is a word from a fixed list: `OPEN`, `REVIEW`, `APPROVED`, `DENIED`, `PAID`, and `CLOSED`. These small, fixed vocabularies are enforced with check constraints, which you will write in Chapter 6.

A preview of what the data says

Because the sample data is small and fixed, every result in this book can be checked by hand. That is deliberate: when you run a query and see a number, you should be able to verify it. Here is a preview of a query you will be able to write by Chapter 8. It joins policies to agents and branches, keeps only active policies, and totals the annual premium for each branch.

SQL

```
SELECT B.CODE,
       B.NAME AS BRANCH_NAME,
       COUNT(*) AS POLICIES,
       SUM(P.ANNUAL_PREMIUM) AS PREMIUM
FROM HALV.POLICY P
JOIN HALV.AGENT  A ON A.AGENT_ID  = P.AGENT_ID
JOIN HALV.BRANCH B ON B.BRANCH_ID = A.BRANCH_ID
WHERE P.STATUS = 'A'
GROUP BY B.CODE, B.NAME
ORDER BY PREMIUM DESC;
```

Against the canonical sample data, the result is:

Output (column padding trimmed)

```
CODE BRANCH_NAME                POLICIES    PREMIUM
---- -------------------------- ----------- ------------
MPF  Maple Falls Head Office              3      4095.50
```

```
BRW  Birchwood                          1       2400.00
LKV  Lakeview                           1       1520.00

  3 record(s) selected.
```

Five of the seven policies are active (Liam Chen's renters policy has lapsed and Ava Patel's auto policy was cancelled), and their premiums add up to 8015.50. The head office accounts for three of those policies because both of its agents, Hannah Cole and Marcus Bell, are based there. When you reach Chapter 8, you will build branch reports like this one yourself and see exactly these numbers.

> **NOTE**
>
> The CLP pads each column to the full width of its data type, so a `VARCHAR(60)` column produces a very wide display. In this book we trim that padding in output listings so they fit the page. The values and row order are exactly what Db2 returns; only the blank space differs.

A reference date

Insurance data is full of dates: policy start and end dates, loss dates, payment dates. Queries that use `CURRENT DATE` return different answers depending on when you run them, which makes printed results hard to compare. To keep the book consistent, we treat **31 March 2026** as "today" in every example that depends on the current date, and we say so whenever it matters. If you run such a query on a different day, your result may differ, and we usually show how to substitute the fixed date `DATE('2026-03-31')` so you can reproduce the printed output exactly.

What You Will Build

This book is organized in six parts that take Halverson Mutual from an empty server to a production system with a runbook.

1. **Part I — Getting Started** (Chapters 1–5). Install Db2 Community Edition in a container, create `HALVDB`, learn the architecture, master the Command Line Processor and GUI tools, and write your first queries.
2. **Part II — Data Modeling and SQL** (Chapters 6–10). Create the six Halverson tables with keys and constraints using the scripts `sql/halv-schema.sql` and `sql/halv-seed.sql`, then learn joins, aggregation, window functions, data changes with `MERGE`, and temporal, JSON, and XML features.
3. **Part III — Programming Db2** (Chapters 11–15). Add views, triggers that audit claim status changes, SQL PL procedures to renew policies and record claim payments, and then connect from Java, Python, Node.js, and .NET with proper transaction handling.
4. **Part IV — Performance** (Chapters 16–19). Create the Halverson indexes, read access plans with EXPLAIN, tune slow queries methodically, and monitor the database with the `MON_GET` functions and `db2pd`.

5. **Part V — Administration and Security** (Chapters 20–24). Manage storage and compression, move data in and out, set up archive logging, practice backup and point-in-time recovery, create the application user `halvapp` and the roles `HALV_REPORTING` and `HALV_CLAIMS`, and automate routine maintenance.
6. **Part VI — Production** (Chapters 25–28). Protect `HALVDB` with HADR, add column-organized analytics, run Db2 in containers and the cloud, upgrade between versions, migrate from other databases, and finish with a complete production runbook.

Developers who mainly query and program Db2 should focus on Parts I to III, adding Part IV when performance matters. New DBAs should read Part I carefully, skim Part II for the schema, and concentrate on Parts IV to VI.

How to read the examples

Most examples in the book are one of three kinds, and each is labeled with a small caption above the listing:

- **SQL listings** are statements you can paste into the CLP, a GUI tool, or a script. Keywords are uppercase, table names are schema-qualified (`HALV.POLICY`), and statements end with a semicolon. SQL PL listings, such as procedure definitions, use `@` as the terminator so that semicolons inside the body are not misread; Chapter 4 explains why.
- **Terminal listings** show commands typed at a shell prompt, such as `db2 connect to HALVDB`, followed by what the command prints. A leading `$` stands for your shell prompt; do not type it.
- **Output listings** show query results in the familiar CLP layout: column headers, a line of dashes, the rows, and then a count such as `3 record(s) selected.`

> **WARNING**
> Do not practice the administration chapters on a database that matters to anyone. Commands such as `RESTORE DATABASE`, `LOAD ... REPLACE`, or `DROP TABLE` are destructive by design. Use the disposable container from Chapter 2, which you can delete and recreate in minutes, until you are confident about what each command does.

Choosing a Setup for Learning

You need a running Db2 server to follow along. There are three practical options, and Chapter 2 walks through the first two in detail.

- **A container (recommended).** IBM publishes Db2 Community Edition as a container image. With Docker or Podman, one command gives you a working instance and database that is isolated, reproducible, and easy to throw away. The book's examples use this route.
- **A native installation.** Installing directly on Linux or Windows teaches the operating system side (users, groups, instance creation, services) and is closer to how many production servers are built.

- **A managed cloud instance.** This removes installation work but hides administration you will want to practice; it suits the SQL and programming chapters better than the administration chapters.

Whichever you choose, give the server enough memory: Db2 uses what you give it for buffer pools and sorting, and a starved instance is slow to learn on.

> **TIP**
> On Apple silicon Macs and other ARM-based machines, check whether the Db2 container image you plan to use supports your processor architecture. If it does not, consider a small x86-64 Linux virtual machine or a cloud server for the exercises, rather than relying on slow emulation.

A Note on Versions and Accuracy

Db2 evolves through versions, fix packs, and mod packs, and the tooling around it changes even faster. To keep the book useful, we prefer long-established syntax that works on both 11.5 and 12.1; we flag defaults and option names that are version-dependent instead of stating them confidently; every query result is computed from the canonical Halverson data; and performance examples never present timings as facts. When something matters, check the IBM documentation for your exact version: this book teaches concepts and workflow, and the reference fills in every option.

Try It Yourself

This chapter is conceptual, so the exercises are about orientation and planning rather than typing commands.

1. Find out which Db2 product your organization (or a system you are curious about) uses: Db2 LUW, Db2 for z/OS, or Db2 for i. If it is Db2 LUW, find out the version and, if possible, the edition. Note where you found the answer.
2. Find the IBM Documentation pages for Db2 LUW 12.1 and bookmark the SQL and command references.
3. Using the mapping table in this chapter, write down the equivalents of "instance", "schema", "table space", and "buffer pool" in the database you know best, and note one way in which you expect Db2 to behave differently.
4. Decide which setup you will use for the exercises (container, native installation, or managed service) and check that your machine meets the basic requirements: enough free memory and disk, and a supported processor architecture.
5. Using only the description of Halverson Mutual in this chapter, predict how many active policies each agent has. You will verify your answer with SQL in Chapter 8.