Databender

FREE GUIDE

The ERP Integration Guide

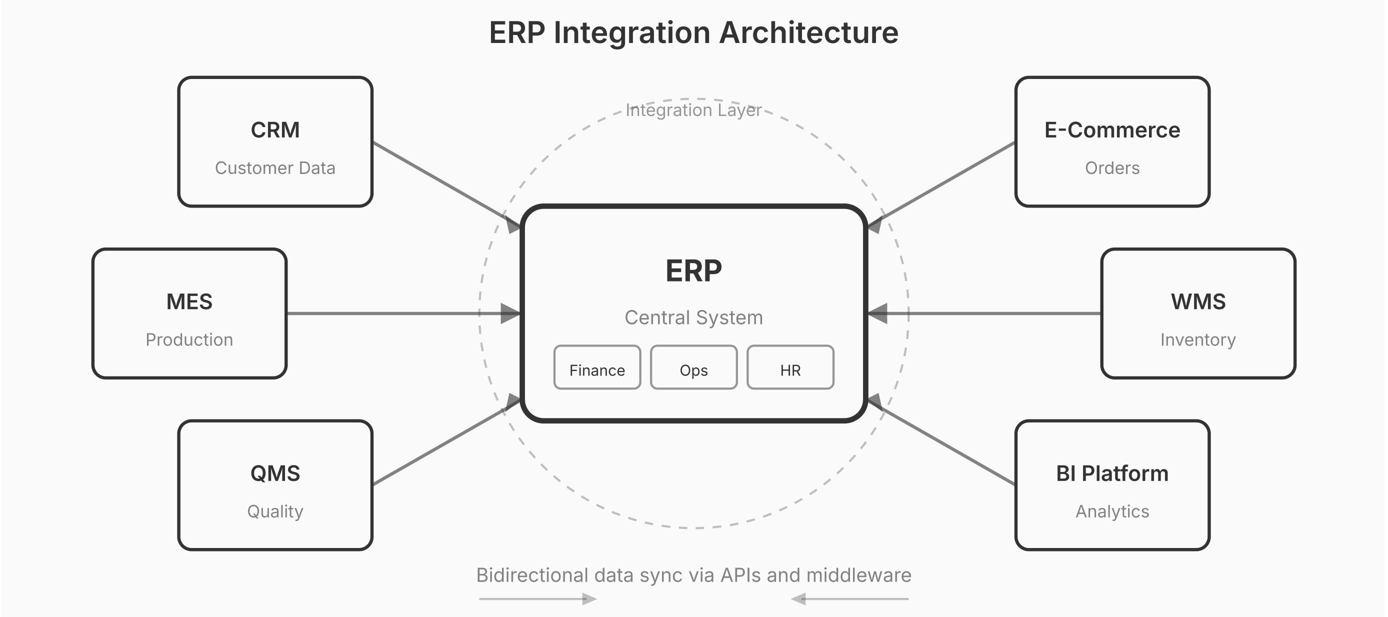
Connect Your Systems Without Replacing Anything

databender.co | Data & AI Consulting for Growing Businesses

$0

System Replacement

Connect existing systems without expensive replacements.



Your ERP vendor called again. They want to sell you more modules. Customer relationship management. Business intelligence. Shop floor control. Document management. All integrated, they promise. Just sign here and migrate your entire operation to their platform.

You've heard this pitch before. You know how it ends. The "integrated" modules are clunky compared to best-of-breed alternatives. The implementation takes twice as long as promised. Your team hates the new interfaces. And you've spent eighteen months disrupting operations to end up with software that's mediocre at everything instead of good at something.

There's another path. Keep your ERP doing what ERPs do well. Keep your other systems doing what they do well. Connect them so data flows without manual transcription, and build views that show what each person needs to see.

Integration without replacement. It's less disruptive, often cheaper, and delivers value faster than the "rip and replace" alternative.

What ERPs Actually Do Well

ERPs are record-keeping systems. They track transactions. They manage the financial truth of what happened. They're the system of record for orders, inventory, purchases, invoices, and payments.

NetSuite, Epicor, SAP Business One, Sage, SYSPRO, Acumatica. Different interfaces, different price points, similar core function. They record what happened and maintain the accounting reality.

What ERPs don't do well: user interfaces designed for speed. Real-time visibility across departments. Flexible reporting for people who aren't accountants. Workflow automation that adapts to how your business actually works.

The ERP vendor's answer is always more modules. But those modules are designed by the same people who designed the ERP interface. They bring the same strengths (transaction accuracy) and weaknesses (usability, flexibility) to every function they touch.

Integration takes a different approach. Let the ERP be the ERP. Build usability, visibility, and automation on top of it.

Read-Only Connections That Don't Disrupt Operations

The first rule of ERP integration: don't break the ERP.

Every manufacturer has horror stories. The integration that corrupted order data. The automation that created phantom inventory. The workflow that duplicated invoices. IT teams are rightfully protective of their ERPs because the consequences of getting it wrong are severe.

Start read-only. Pull data out of the ERP without writing anything back. This approach carries zero risk to the source system. You can build views, reports, and alerts without any possibility of corrupting production data.

Read-only integration covers most visibility needs:

- Order status for customer service
- Inventory levels for sales
- Payment history for collections
- Production schedules for planning
- Backlog reports for leadership

None of these require writing to the ERP. They just need to read what's already there and present it usefully.

One building products manufacturer had been warned for years about integration risks. Their IT director had seen a botched implementation at a previous company. No way he was letting anyone touch the ERP.

We built their first integration completely read-only. Daily extracts of order data, inventory positions, and shipment records. No writes. No risk. The IT director could verify exactly what data was flowing.

Six months later, he asked about automating order entry. The trust was earned through cautious first steps.

What Your ERP Can Tell You

ERPs contain far more useful data than most people realize. The information is there. It's just locked behind interfaces designed for transaction entry, not analysis.

Customer patterns. Who orders frequently? Who used to order frequently but stopped? Who buys which products? Who pays quickly versus slowly? The data is in the sales history and payment records. The ERP just doesn't present it usefully.

Product performance. What's selling? What's sitting? What has growing demand versus declining? What generates margin versus just volume? The transaction history contains the answers.

Operational reality. What's the actual lead time for different products (not the promised time, the actual time)? Where do orders get stuck? Which processes create delays? The timestamps are all recorded.

Cash flow signals. When do customers actually pay versus when they're supposed to? Which customers are stretching terms? Which have changed payment patterns? AR aging is standard. But trend analysis on AR patterns isn't.

Most manufacturers use their ERP for 20% of what it can tell them. The other 80% sits unused because nobody has built the views that make it accessible.

A precision machining company discovered that their largest customer had been quietly reducing order size over 18 months. The decline was clear in the ERP data. Nobody had looked at it that way. A simple monthly report showing order trends by top customers would have spotted the pattern a year earlier.

Building Applications Your Team Will Actually Use

The best application is the one people use without being forced to.

ERP modules fail adoption tests because they're designed for transaction accuracy, not daily workflow. They're dense, menu-driven interfaces built for accountants. When you force salespeople or production supervisors to use them, they find workarounds. Spreadsheets. Paper. Asking someone else to look things up.

Effective integration applications follow different principles.

One purpose per screen. The customer service rep needs to see order status. Give them one screen that shows order status. Don't make them navigate through inventory, accounting, and shipping menus to find the answer to a simple question.

Default to what they usually need. If a sales rep checks the same 30 accounts every day, start with those accounts displayed. Don't make them search every time. Smart defaults save clicks.

Mobile-friendly without being mobile-first. The production supervisor who checks job status from the floor needs a view that works on a phone. The CFO reviewing financials at their desk doesn't need a mobile interface taking up half the screen real estate.

Speed over features. A simple view that loads in one second beats a complex view that loads in ten. People won't use slow software, no matter how powerful it is.

We built a sales dashboard for an industrial distributor. The ERP had a sales module with hundreds of features. Adoption was maybe 20%. The new dashboard did exactly three things: showed today's orders, highlighted overdue payments, and displayed year-over-year trends by account. Adoption hit 90% in the first month. Salespeople checked it because it answered questions faster than asking someone.

The 30-Day Path to Your First Live View

Thirty days is enough time to build something useful.

Days 1-5: Define the use case. Pick one question that one group of people asks regularly. "What orders shipped today?" "Which customers have overdue payments?" "What's the backlog for next week?" Be specific about who needs the answer and how often.

Days 6-10: Map the data. Where does the answer live in the ERP? Which tables, which fields, which connections? This is technical work that requires someone who knows the ERP's database structure. Most ERP vendors document this. Your implementation partner should know it.

Days 11-15: Build the connection. Connect to the ERP database or API. Pull the required data into an integration layer. Verify the data matches what you see in the ERP interface.

Days 16-25: Build the view. Create the application that displays the data. Keep it simple. Solve the original use case. Don't add features nobody asked for.

Days 26-30: Deploy and iterate. Get it in front of users. Watch them use it. Listen to feedback. Fix the obvious issues.

This timeline assumes you know what you're building and have access to technical resources. First projects sometimes take longer because you're also learning. But the principle holds: valuable applications don't require months of development.

Common Integration Patterns

Most integration projects follow familiar patterns. Knowing what works helps you plan.

Order-to-shipment visibility. Connect ERP orders to shipping carrier tracking. When an order ships, the tracking number flows back to the customer record. Customer service can see status without logging into the carrier website.

CRM to ERP sync. When a quote converts to an order in the CRM, create the sales order in the ERP automatically. Eliminate the manual re-entry that creates errors and delays.

Production schedule visibility. Pull work orders and completion status from the ERP. Display them in a format that production supervisors can actually use. Add alerts when jobs fall behind.

Inventory alerts. Monitor stock levels in the ERP. When inventory for a critical component drops below threshold, alert purchasing. Don't wait for someone to notice during the weekly review.

AR intelligence. Pull payment patterns from the ERP. Identify customers whose payment behavior is changing. Alert collections before invoices become severely past due.

These patterns work across different ERPs because the underlying business processes are similar. The technical implementation varies, but the concept translates.

What Integration Tools Exist

You don't build integrations from scratch anymore. Tools exist that make connecting systems much faster than custom development.

Integration platforms (Workato, Celigo, Boomi, MuleSoft): Visual tools for connecting systems. Pre-built connectors for common ERPs. Drag-and-drop workflow design. Good for teams that want to manage their own integrations long-term.

ERP-specific tools (various): Some ERPs have integration ecosystems. NetSuite has SuiteScript and SuiteFlow. SAP has integration services. These work well but lock you into the ERP platform.

Custom development: Still necessary sometimes. Unusual systems, complex business logic, high-performance requirements. But the percentage of integrations that need custom code has dropped dramatically.

For most mid-sized manufacturers, integration platforms offer the best balance. Flexible enough to handle unusual requirements, structured enough that you're not maintaining custom code forever.

What This Typically Costs

Integration costs depend on scope. Here are realistic ranges.

Single-view application (order status, inventory levels): $15,000 to $25,000. Connecting one or two data sources, building one targeted application. Four to six weeks.

Multi-system visibility (order + shipment + payment): $35,000 to $55,000. Connecting three or four systems, building several views for different users. Eight to twelve weeks.

Integration layer with automation: $60,000 to $100,000. Connecting most core systems, building applications and alerts, automating key workflows. Twelve to twenty weeks.

These are project costs. Ongoing costs include integration platform licensing ($500 to $2,000 monthly depending on volume) and occasional maintenance when source systems change.

Compare to the alternatives. ERP module implementations typically start at $50,000 and run into six figures. Full ERP replacements exceed $200,000 for mid-sized companies and take one to two years. Integration delivers value faster at lower cost and lower risk.

Getting Started

You don't need executive sponsorship or steering committees to start improving ERP visibility. You need one specific problem and permission to solve it.

Find the question someone asks every day that takes too long to answer. "Where's this order?" "Do we have that in stock?" "Did they pay yet?" Pick the question with the clearest answer that the most people ask most often.

Build one view that answers it. Connect to the ERP read-only. Display the information simply. Get it in front of the people who need it.

Once you've solved one problem, the next one gets easier. You have the connection. You have the pattern. You have proof that integration works in your environment.

The goal isn't to replace your ERP. It's to unlock the value that's already trapped inside it.

> *Ready to get more value from your ERP without replacing it? Talk to us about integration that works with what you have, or explore our full manufacturing solutions.*