Modern SAP Architecture: Why It Matters

The Problem Every SAP Organization Faces

The disconnect: Mobile speaks REST/JSON, SAP speaks ABAP/RFC

🎨 Users Want:

Beautiful, intuitive web apps

⚙️ Developers Have:

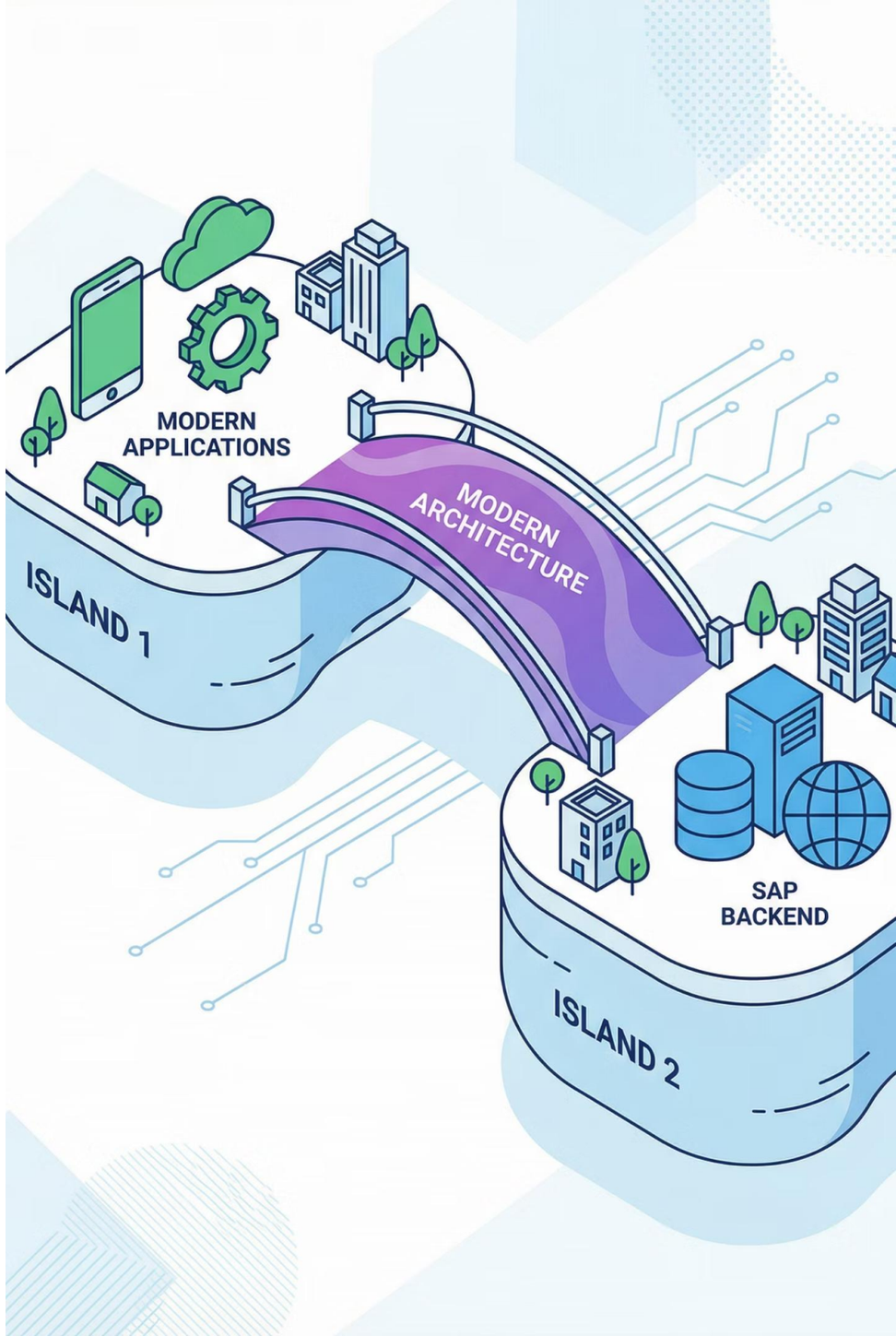
Powerful backends with decades of business logic

This Module Solves That Gap

❌ Without rebuilding everything

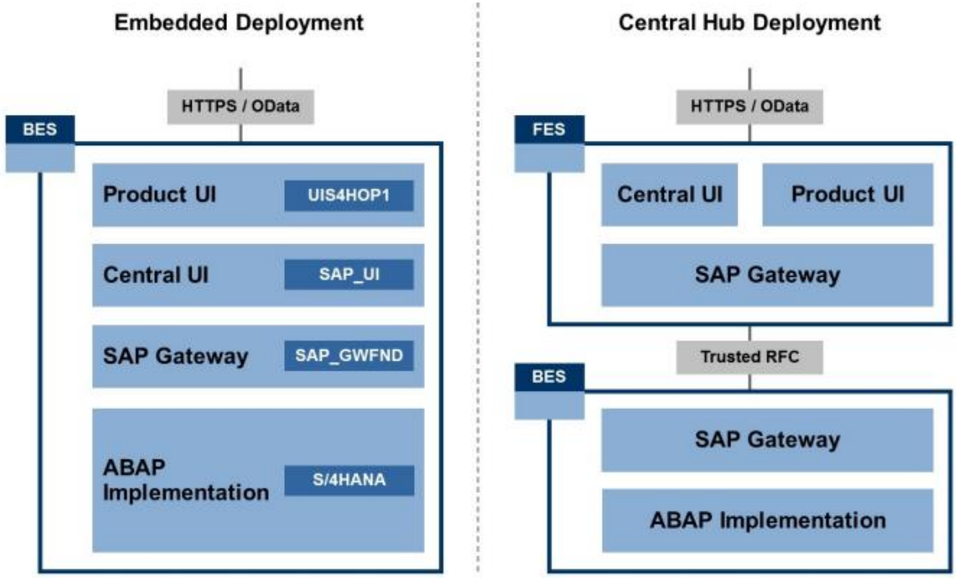
The Bridge: Modern SAP Architecture

How to connect two worlds speaking different languages



Your Learning Journey

📚 What You'll Master:



Front-end & Back-end Servers

SAP Gateway

(translation layer)

SAP HANA

(complete platform)

S/4HANA

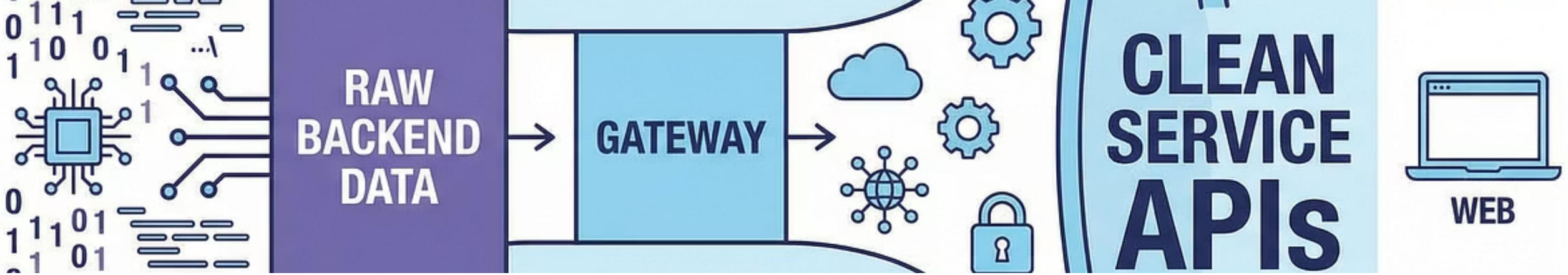
(modern paradigm)

We Start With the Big Picture

🏗️ System Landscape Components:

1. Front-end Server
2. Back-end Server

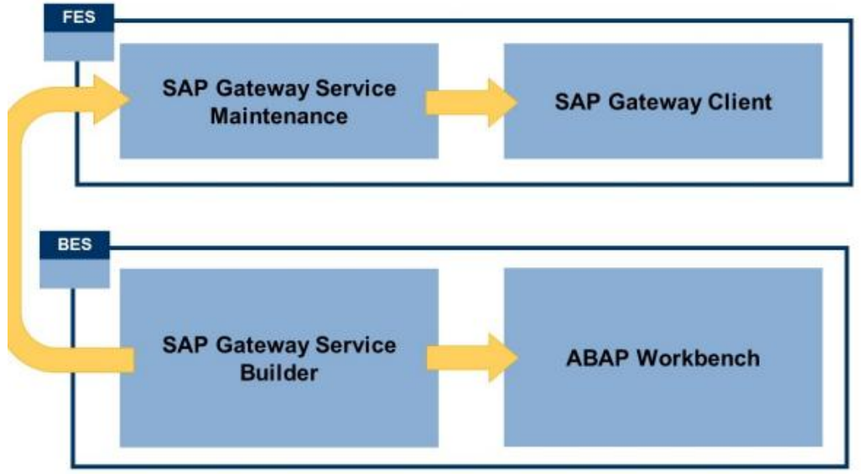
Understanding how they communicate



Then Gateway: The Translation Layer

SAP Gateway

Transforms: Backend data → Services any app can consume

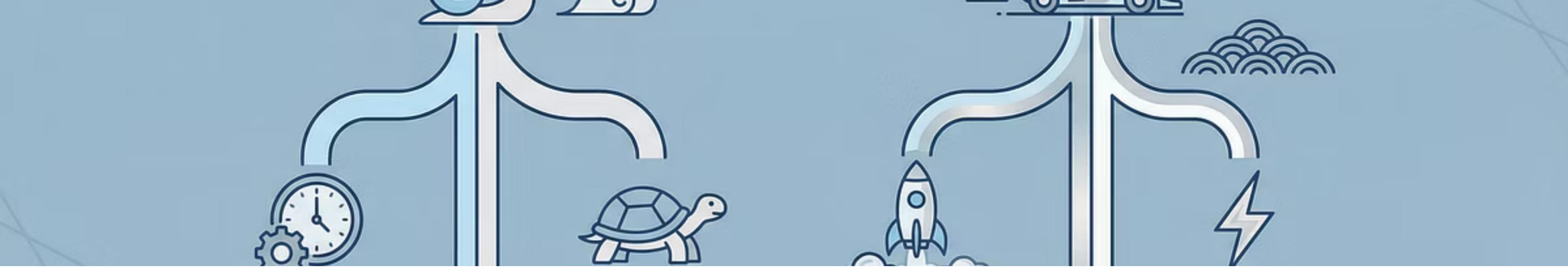


SAP HANA: More Than a Database

⚡ SAP HANA = Complete Platform

Processes data at lightning speed

Not just storage - a processing powerhouse



Why This Matters for Consultants

👔 Consultants Design System Landscapes

Wrong choice ❌

- Latency issues
- Frustrated users

Right choice ✅

- Apps load instantly
- Processing in milliseconds

Impact: Performance, scalability, cost for the next decade

Why This Matters for Developers

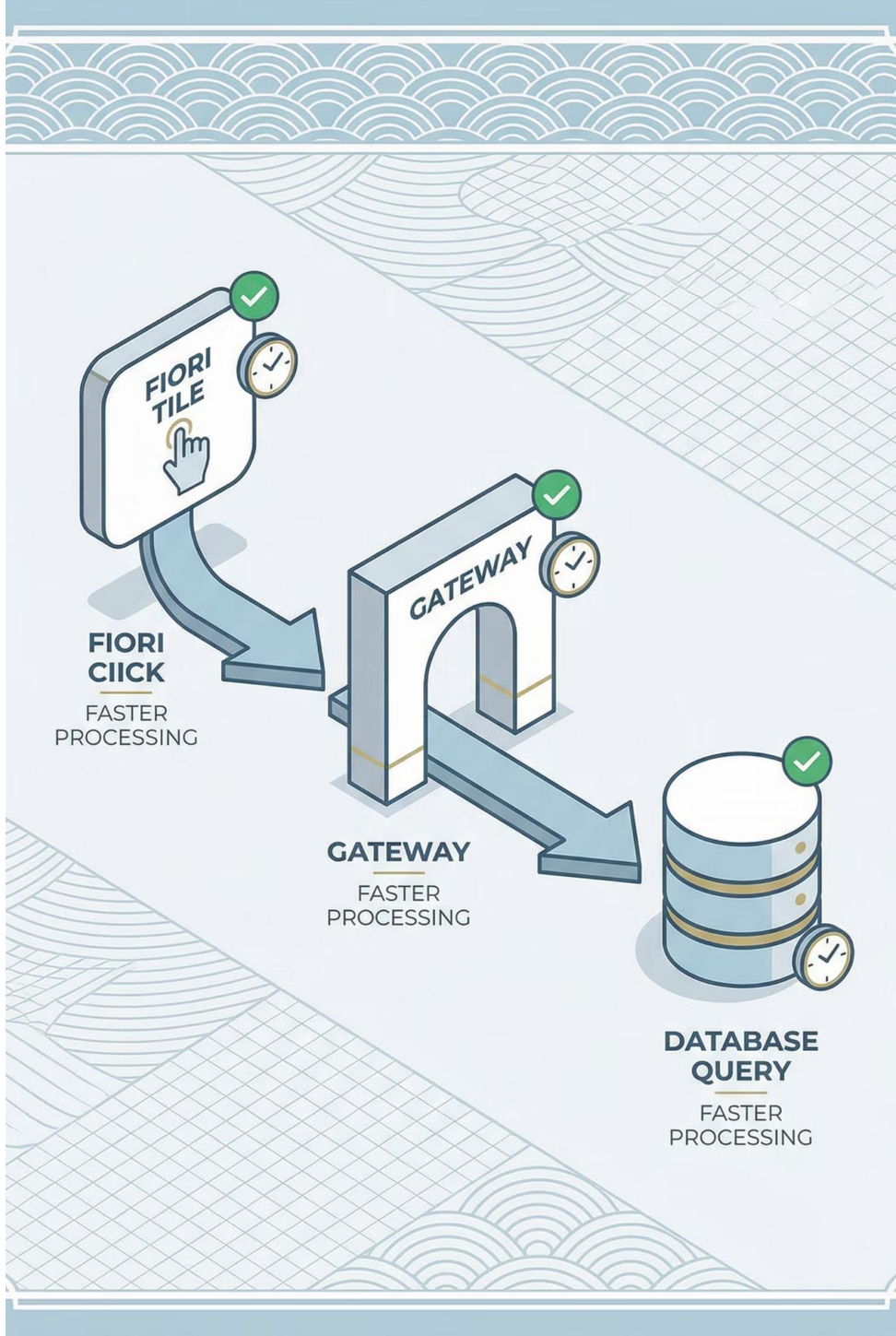
👩‍💻 Developers Write Scalable Services

Know where code runs

Troubleshoot performance in minutes

Understand: Fiori → Gateway → Database

From: Days debugging → To: Minutes solving



Why This Matters for Architects

🏛️ Architects & Decision-Makers Gain:

Foundation to evaluate:

- 📋 Migration strategies
- 📈 Upgrade plans
- 💬 Technical challenges

Speak confidently about complex technical decisions

How We'll Build Your Expertise

📖 Module Structure:

Lesson 2

Fiori System Landscape

Lesson 3

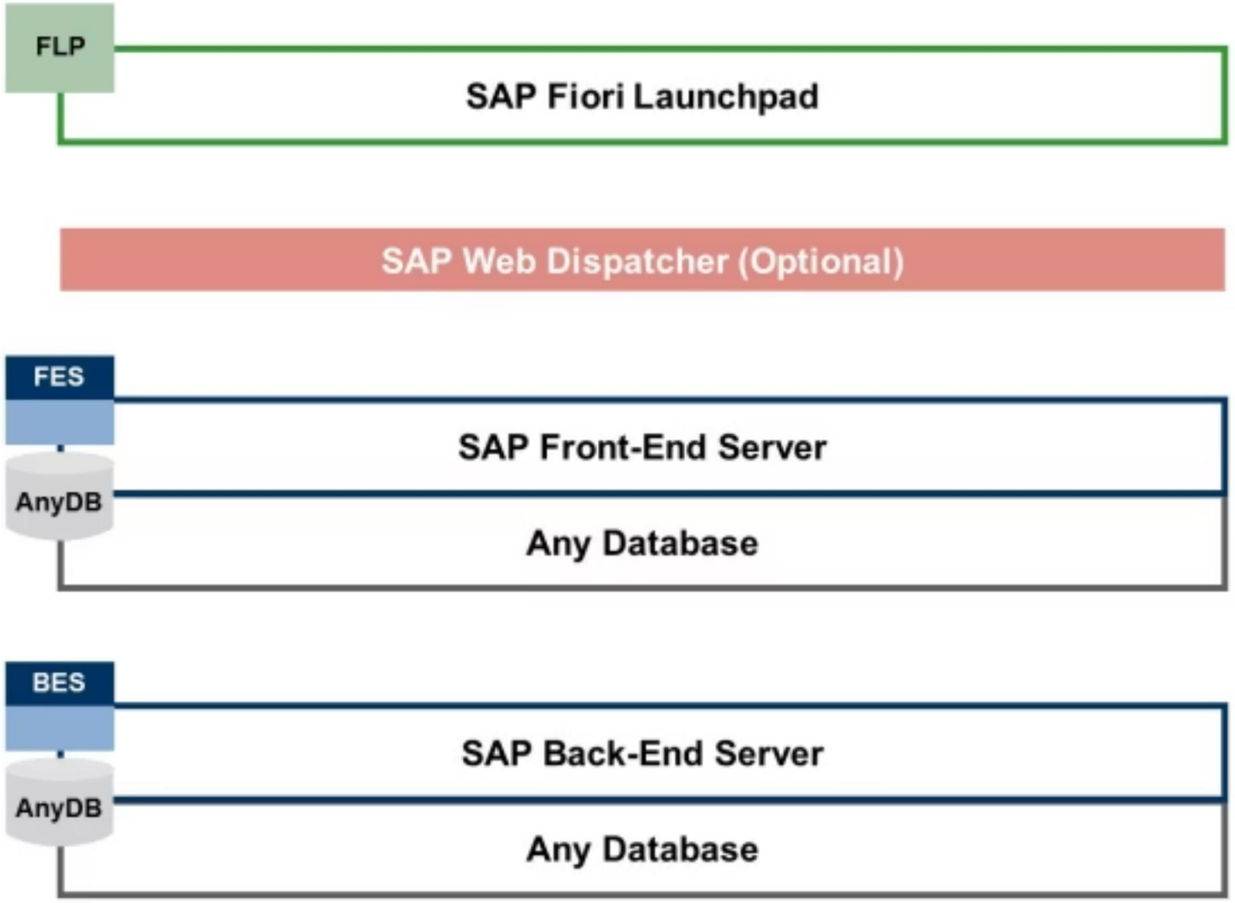
Gateway & OData Services

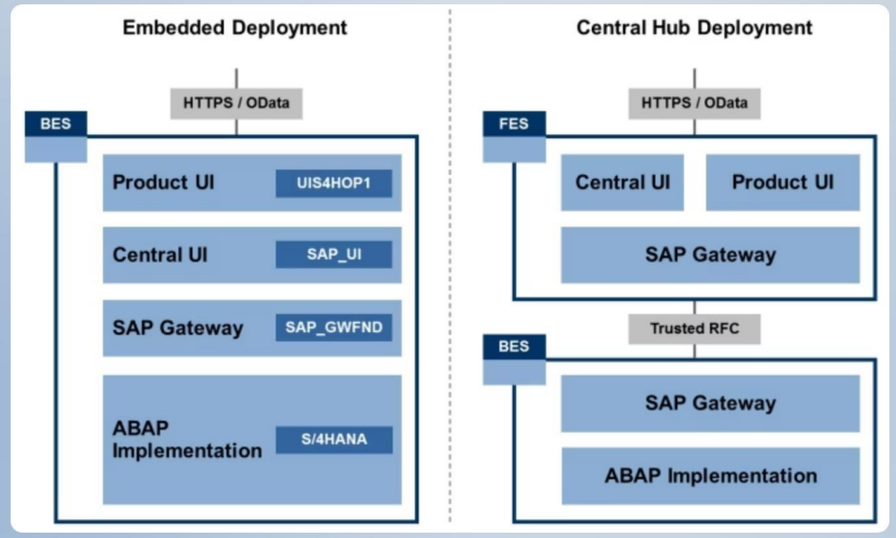
Lessons 4-5

HANA & S/4HANA Platform

Progressive mastery from landscape to implementation

LESSON 12 - One System, Two Systems, or Cloud? Choosing Your Fiori Architecture





The Conference Confusion

🤔 You're talking to another SAP admin at a conference

🖥️ Admin #1

Front-end + back-end servers

💻 Your setup

Single SAP system running Fiori

☁️ Admin #2

Fiori Launchpad in BTP cloud

The burning question:

Who's doing it right?

They're All Right (Depending on the Problem)

They're all right, depending on what problems they're solving

✅ The Truth About Fiori Architecture:

- No single "correct" approach
- Each solves different business problems
- Architecture follows requirements

🎯 What You'll Master:

- Embedded vs. Central Hub deployments
- When to split systems (and when not to)
- How to match architecture to business needs

Where Technologies Actually Live

Now we're looking at where those technologies actually live

1

SAPUI5

User interface framework

2

SAP Gateway

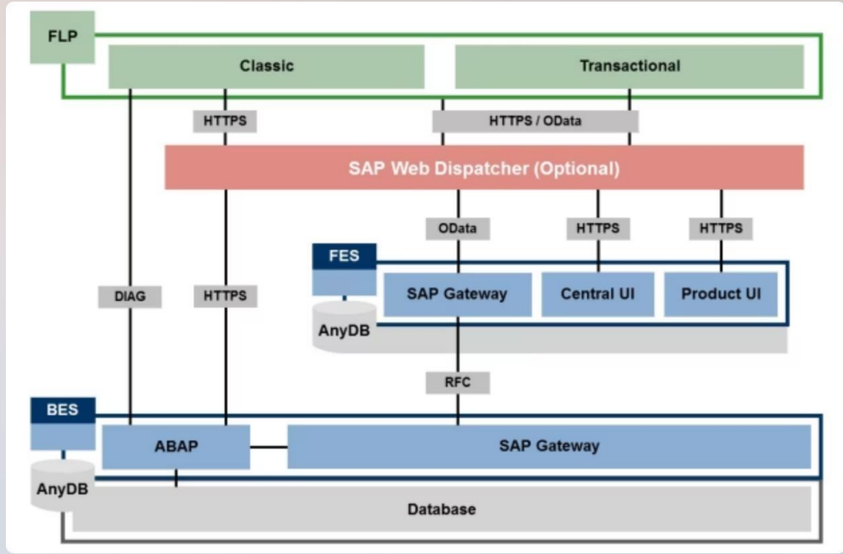
OData communication layer

3

Three App Types

Transactional, Analytical, Fact Sheets

📍 The Key Question: Where do these components physically reside in your landscape?



The Restaurant Analogy

Think of your SAP landscape like a restaurant

🍽️ Restaurant = SAP Landscape

👩‍🍳 Back-End Server = Kitchen:

- Chefs = ABAP code
- Ingredients = Database
- Food preparation = Business logic

👥 Front-End Server = Dining Room:

- Menu = Fiori Launchpad
- Wait staff = UI components
- Customer experience = User interaction

Back-End Server: The Kitchen

The back-end server is the kitchen

🍳 Where the Real Work Happens:

- ⚙️ Business applications (S/4HANA, ECC)
- 📊 Business data and logic
- 🗄️ Database storage

📋 Technical Requirements:

- ABAP 7.40 or higher
- Any database (Oracle, DB2, HANA)
- Full business suite installation

Front-End Server: The Dining Room

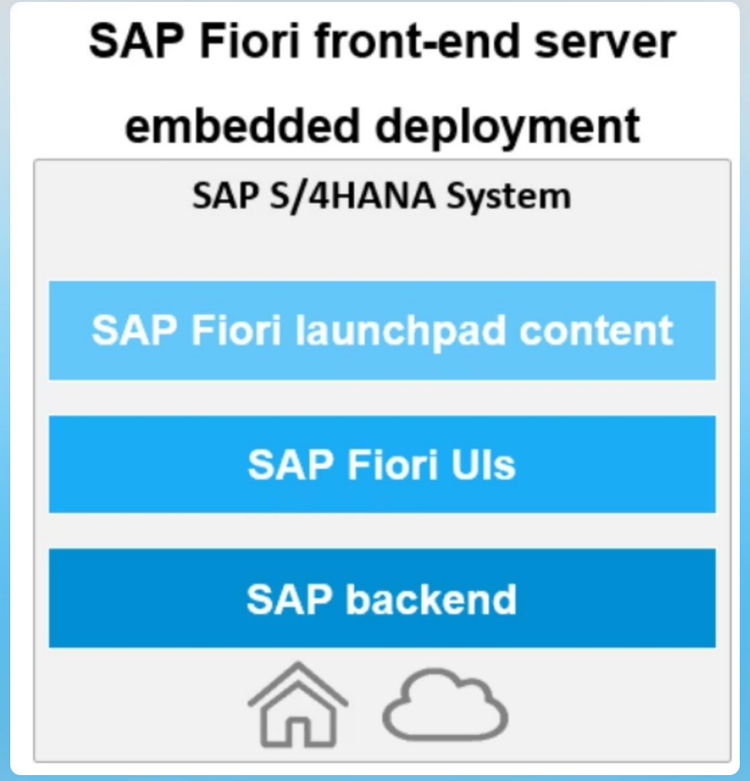
The front end server is your dining room and wait staff

🎨 Customer-Facing Experience:

- 📱 Fiori Launchpad (the menu)
- 🖥️ SAPUI5 components
- 👥 User interaction handling

⚙️ Technical Setup:

- Basic ABAP system (no business products)
- ABAP 7.40+ with any database
- Only Fiori interface components



Embedded Deployment: Open Kitchen

Embedded deployment is like having an open kitchen restaurant

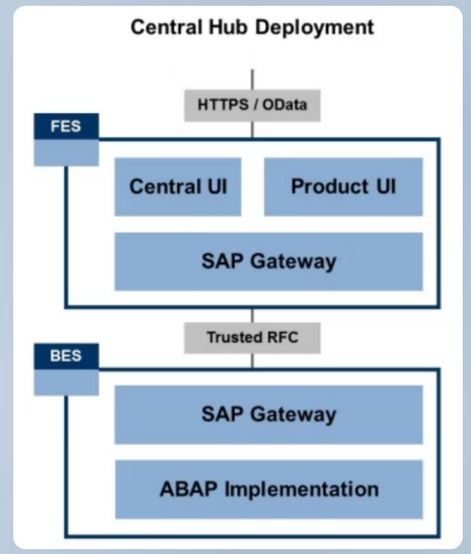
🏠 Everything in One Space:

- ✅ Kitchen + dining room together
- ✅ Front-end + back-end on same server
- ✅ One system does everything

📈 Benefits:

- Simpler maintenance
- Better performance (no network hops)
- Lower infrastructure costs

SAP's direction: Only option for S/4HANA 2025+



Central Hub: Separate Buildings

Central hub deployment is like having a separate front of house building

🏢 Split Architecture:

- 🍽️ Dining room = Front-End Server
- 🍳 Kitchen = Back-End Server
- 🔗 Communication via OData/RFC

🎯 When This Makes Sense:

- Multiple backend systems
- Organizational separation needs
- Security and governance requirements

Why Split Systems? Three Big Reasons

Why would anyone do this? Well, three big reasons

01

🌐 Reason 1: Multiple Backend Systems

- ECC for finance, CRM for sales, BW for analytics
- Single entry point for users
- One front door, multiple kitchens

02

🔒 Reason 2: Organizational Benefits

- Separate teams and change windows
- UI updates without touching production
- Enhanced security policies

03

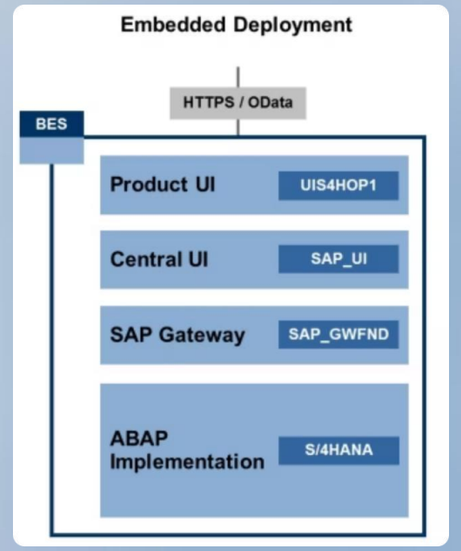
🔄 Reason 3: Release Independence

- Different update cycles
- Fiori UI separate from ERP releases

Scenario One: S/4HANA Greenfield

Your company is planning an S4 HANA Greenfield implementation

✅ Recommendation: Use Embedded Deployment



📋 Situation Analysis:

- Single system implementation
- No legacy integrations
- Straightforward requirements

💰 Business Impact:

- Simpler architecture
- Better performance
- Lower costs
- Aligns with SAP strategy

Scenario Two: Heterogeneous Landscape

You are managing a heterogeneous landscape that includes an old ECC system

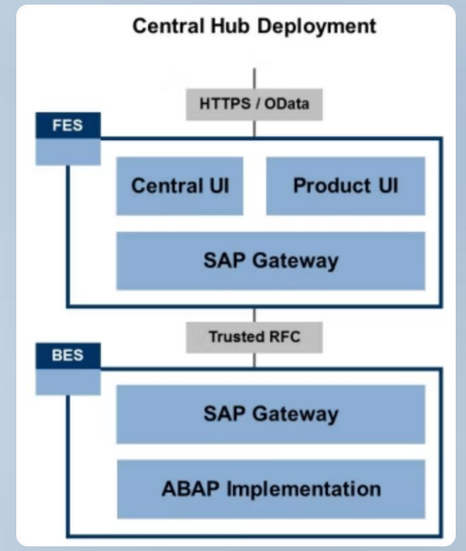
🌐 Recommendation: Central Hub Deployment

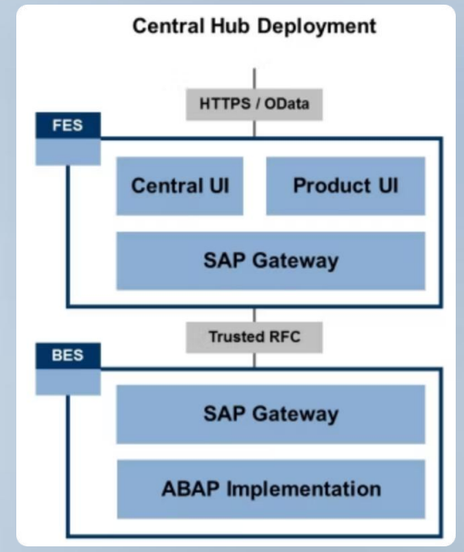
🔧 Complex Environment:

- Legacy ECC system
- New S/4HANA system
- Multiple cloud applications

🎯 Problem Solved:

- Unified user access
- Single sign-on experience
- Integrated landscape management





Scenario Three: Security and Governance

A security audit has flagged concerns about your Fiori setup

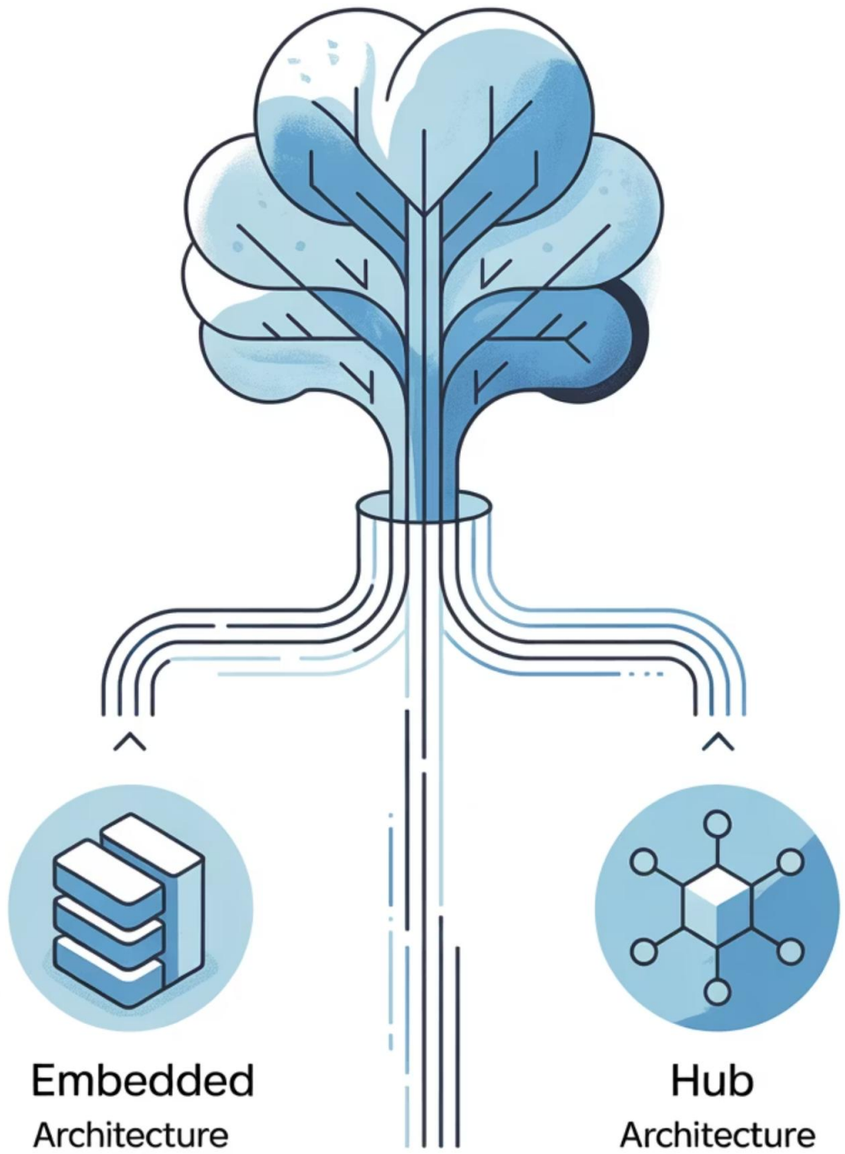
🔒 Recommendation: Central Hub for Separation

⚠️ Security Requirements:

- No development on production backend
- Separate authorization concepts
- Stronger governance controls

✅ Architecture Benefits:

- Development team isolation
- Independent change management
- Enhanced security boundaries



The Game Changer: It's About Business, Not Technology

Here's the game changer. The embedded versus hub decision isn't about technology

🧠 Decision Framework:

🏠 Choose Embedded When:

- Single system + straightforward needs
- Performance is critical
- Simplicity preferred

🌐 Choose Hub When:

- Multiple backends
- Complex integrations
- Organizational separation needed

Key insight: Don't over-engineer simple scenarios

The Cloud Evolution: WorkZone

And increasingly, if you need true multi-system access, the answer isn't an on-premise hub anymore

SAP's Strategic Direction:

🚀 WorkZone for Multi-System Access:

- Cloud-based integration layer
- True multi-system connectivity
- Future-proof architecture

📈 Strategic Principle:

- Simplify where you can
- Cloud-enable where you must

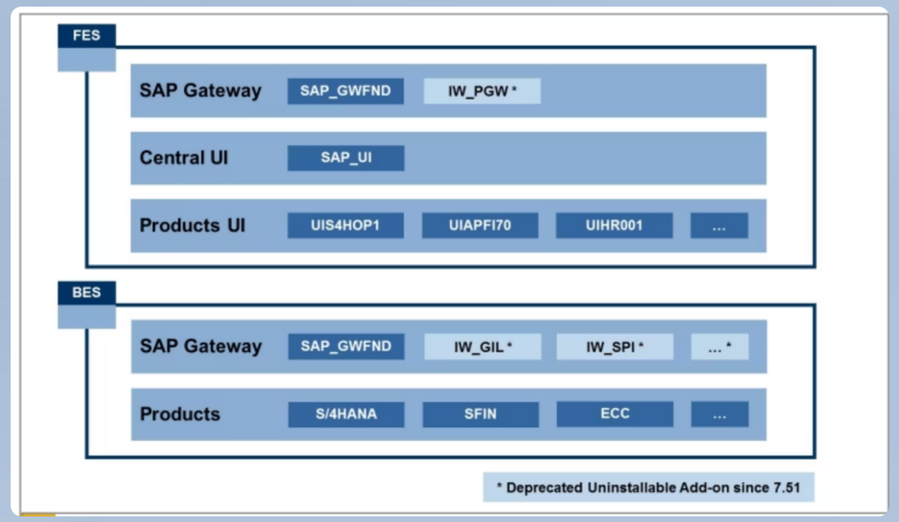
Your Architecture Decision Framework

SAP's strategic direction is clear. Simplify where you can, cloud enable where you must

🎯 Quick Decision Guide:

1. Single S/4HANA System:
   → Embedded (SAP's future direction)
2. Multiple On-Premise Systems:
   → Central Hub (organizational benefits)
3. Multi-System + Cloud Strategy:
   → WorkZone (strategic direction)

Remember: Architecture serves business needs, not technical preferences

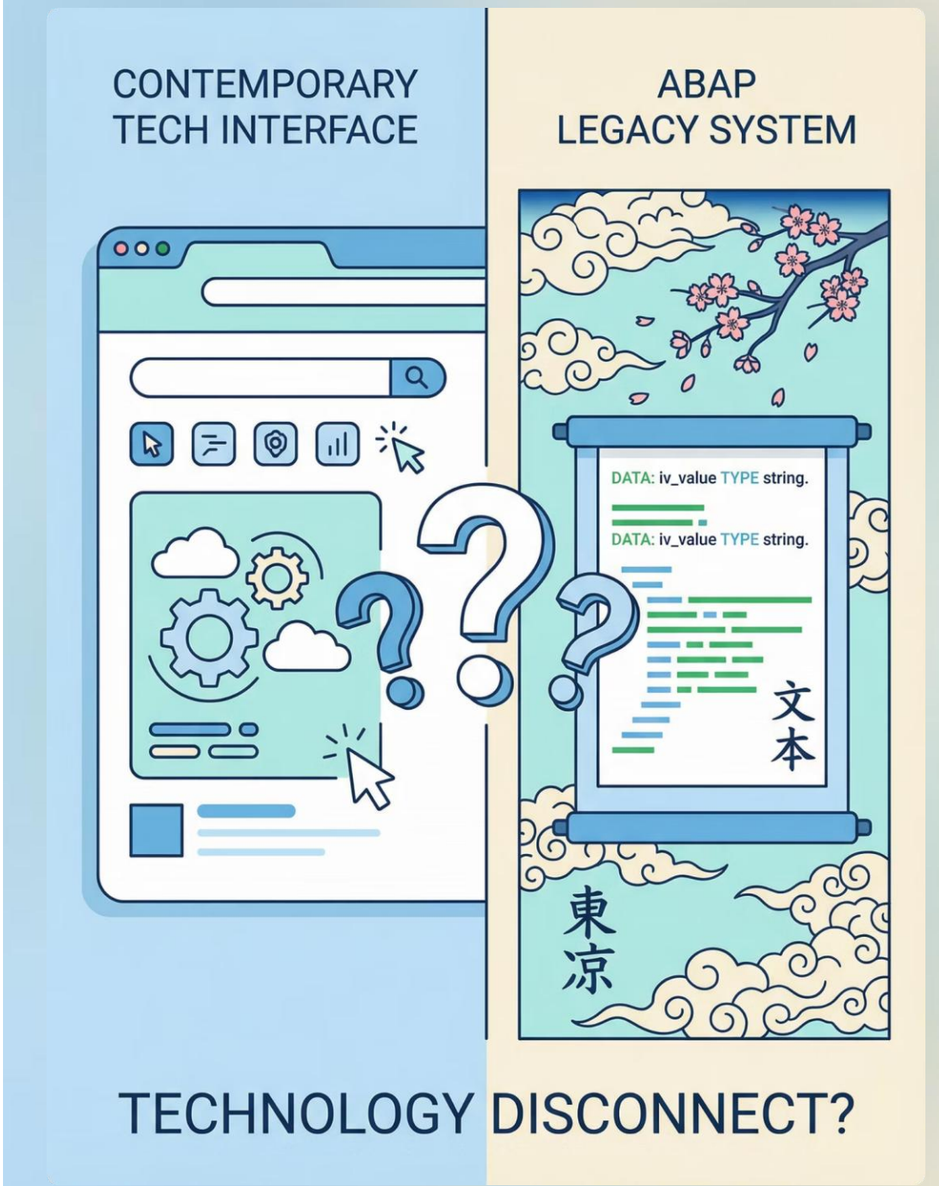


SAP Gateway Fundamentals: The Bridge for Your Data

What if Your Browser Could Speak ABAP?

🌐 Modern Web App ↔️ ❓ ↔️ SAP Backend

The mystery: How do two systems that don't understand each other communicate in milliseconds?



The Two Worlds Problem

SAP Backend

- Speaks ABAP
- Complex table structures
- BAPIs & function modules

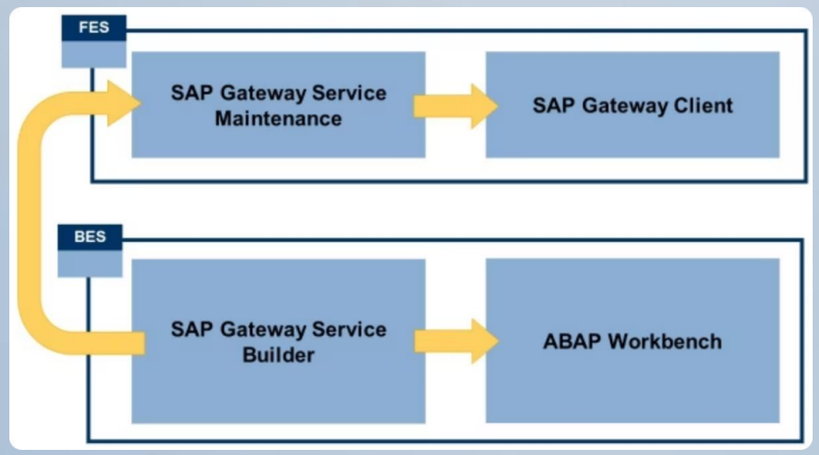
Modern Apps

- Expect HTTP requests
- Want JSON/XML
- Need simple web services

The gap: No common language

Gateway Sits in the Middle

Gateway = Translator converting ABAP logic into OData services



Gateway's Two-Part Architecture

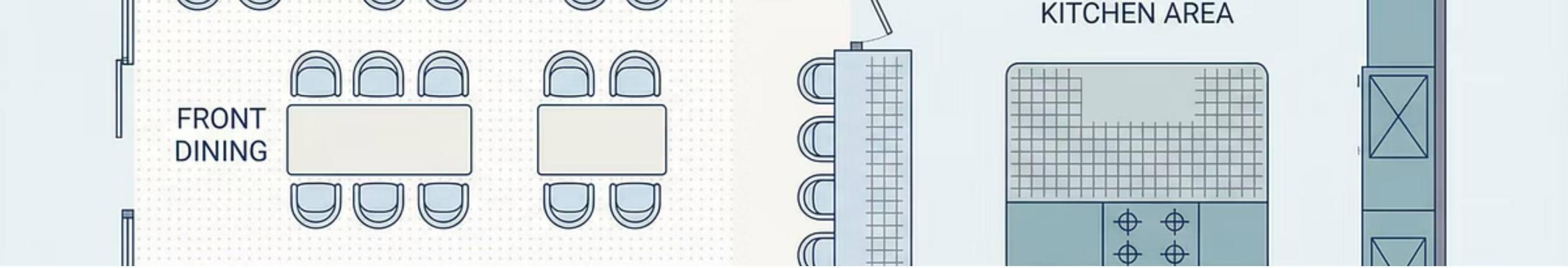
🏢 Gateway Server (Hub)

- Where services are registered
- Where apps consume services

🍳 Gateway Backend (Kitchen)

- Where services are implemented
- Connects to business logic

Communication: RFC protocol



Think of It Like a Restaurant

🍽️ Front of House

(Gateway Server)

- Customers place orders
- Menu published

👩‍🍳 Kitchen

(Gateway Backend)

- Orders prepared
- Ingredients = Backend data
- Recipes = Business logic

Service Artifacts: The Building Blocks

🧱 What Gateway Needs to Know:

Data Model

What to expose

Service Implementation

How to execute

Service Registration

Where to publish

Service Metadata

How apps discover it

The Four Key Artifacts

1

Data Model

- Entities: AirlineFlight, AirlineCarrier
- Properties: CarrierID, Price, SeatsMax
- Relationships between entities

2

Service Implementation

- ABAP code for CRUD operations
- Retrieves from /DMO/FLIGHT table

3

Service Registration

- Publishes to Gateway Server

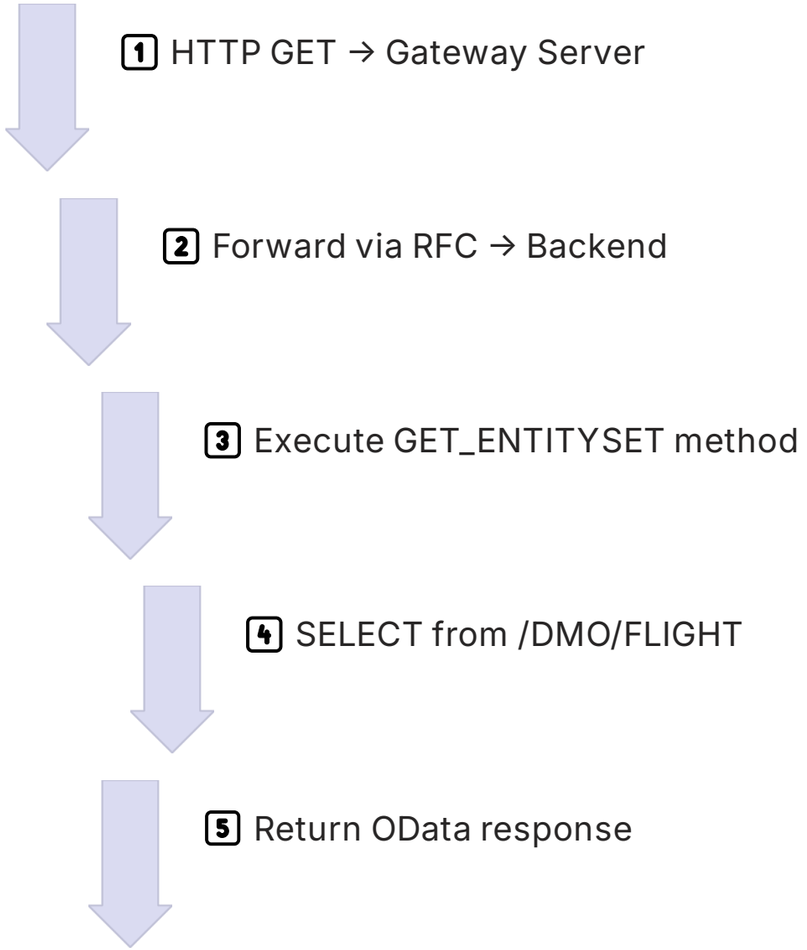
4

Service Metadata

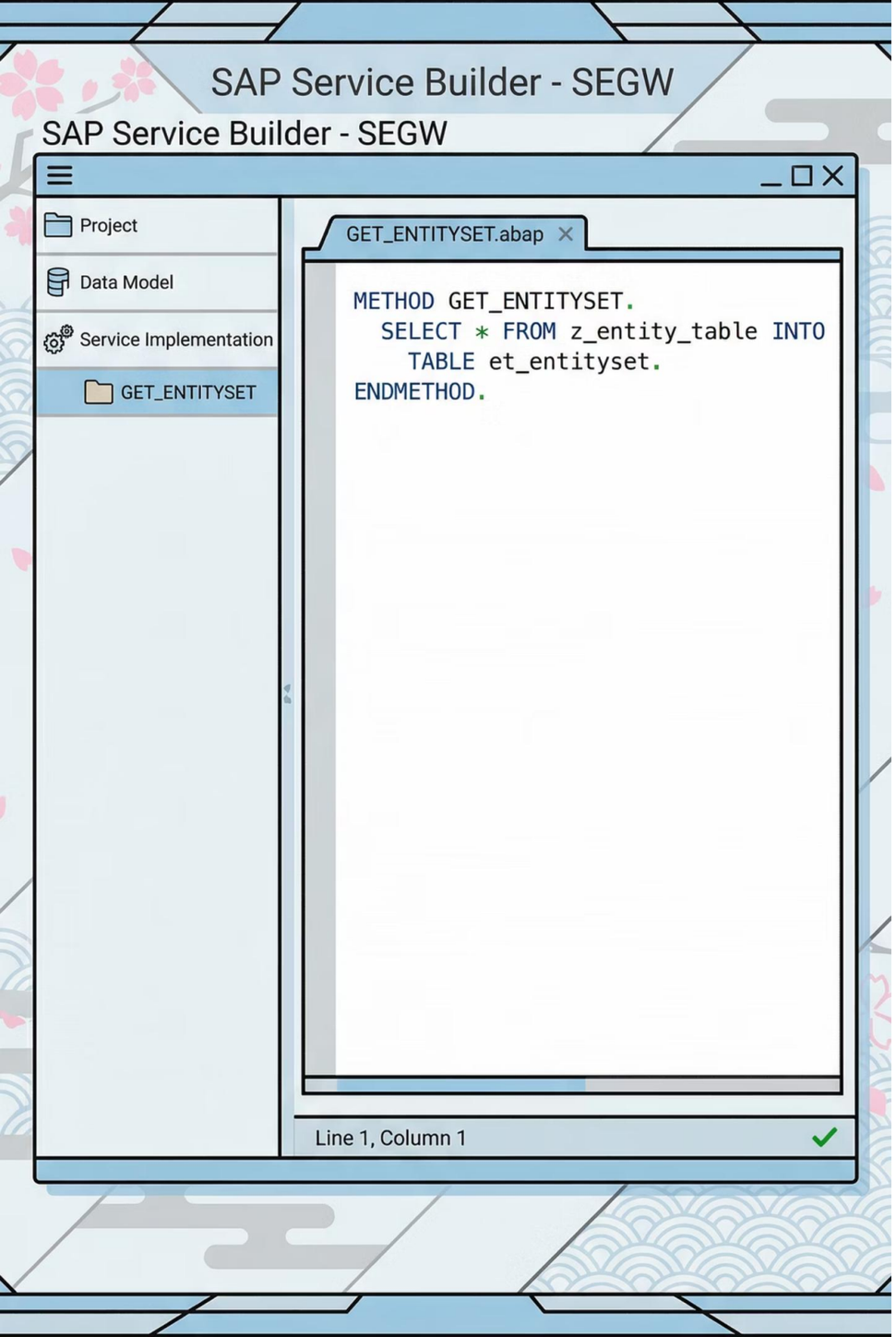
- Auto-generated service description

Real Request Flow in Action

User clicks "View Flights" in Fiori app



Total time: < 1 second



Behind the Scenes: Transaction SEGW

🔧 Gateway Backend Tools:

SEGW

Service Builder

/IWFND/GW_CLIENT

Test HTTP requests

GET_ENTITYSET

ABAP method implementation

Like Postman for SAP - test services internally

The Key Insight

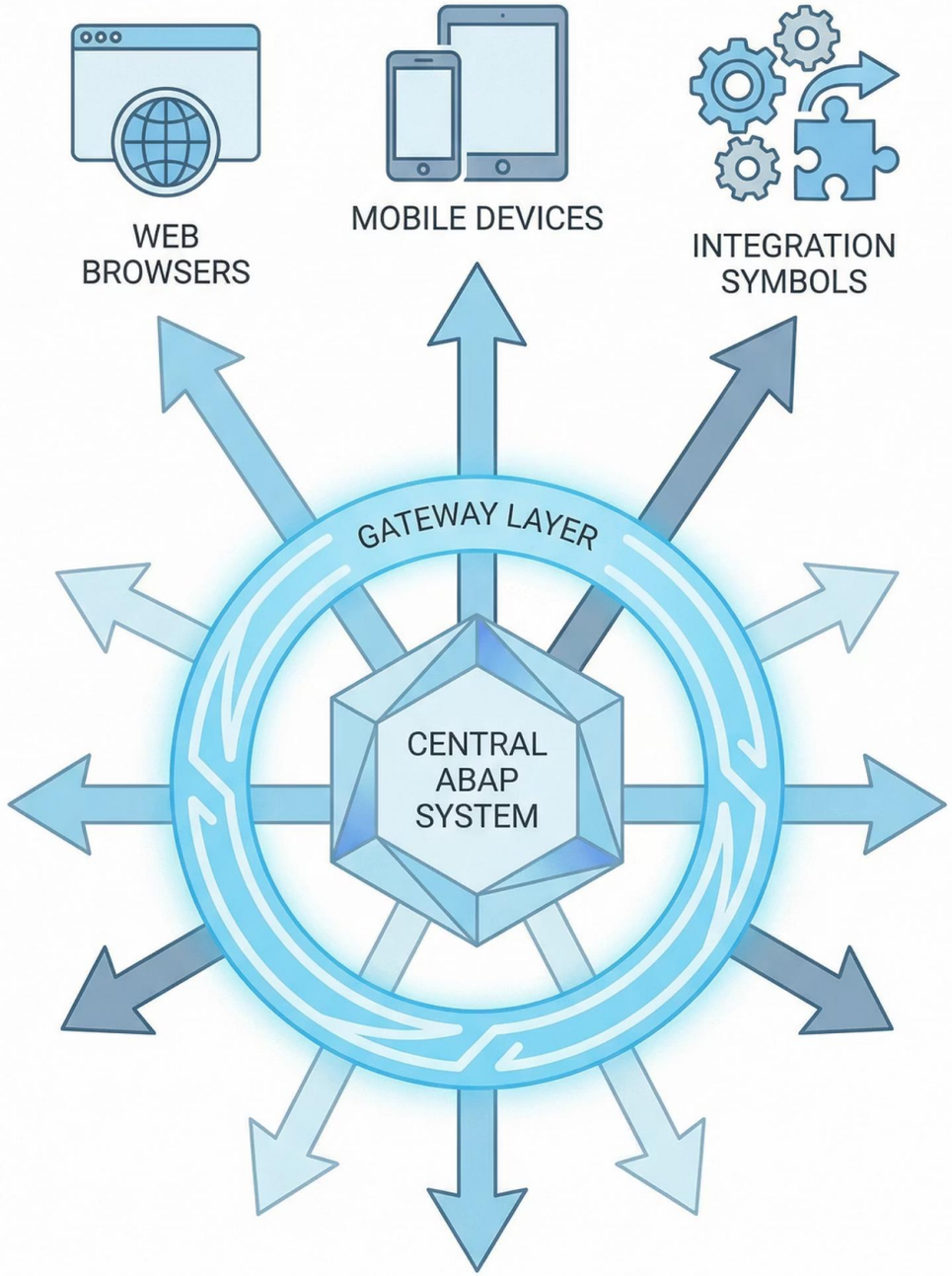
Gateway enables modern access

WITHOUT rewriting business logic

Core ABAP processes stay where they are

Now accessible via HTTP + OData

Web apps, mobile apps, third-party integrations - all can connect



That's the Power of Gateway

SAP Gateway = The Bridge

Two-Part Architecture

🏢 Server (register & consume) + 🔍 Backend (implement & connect)

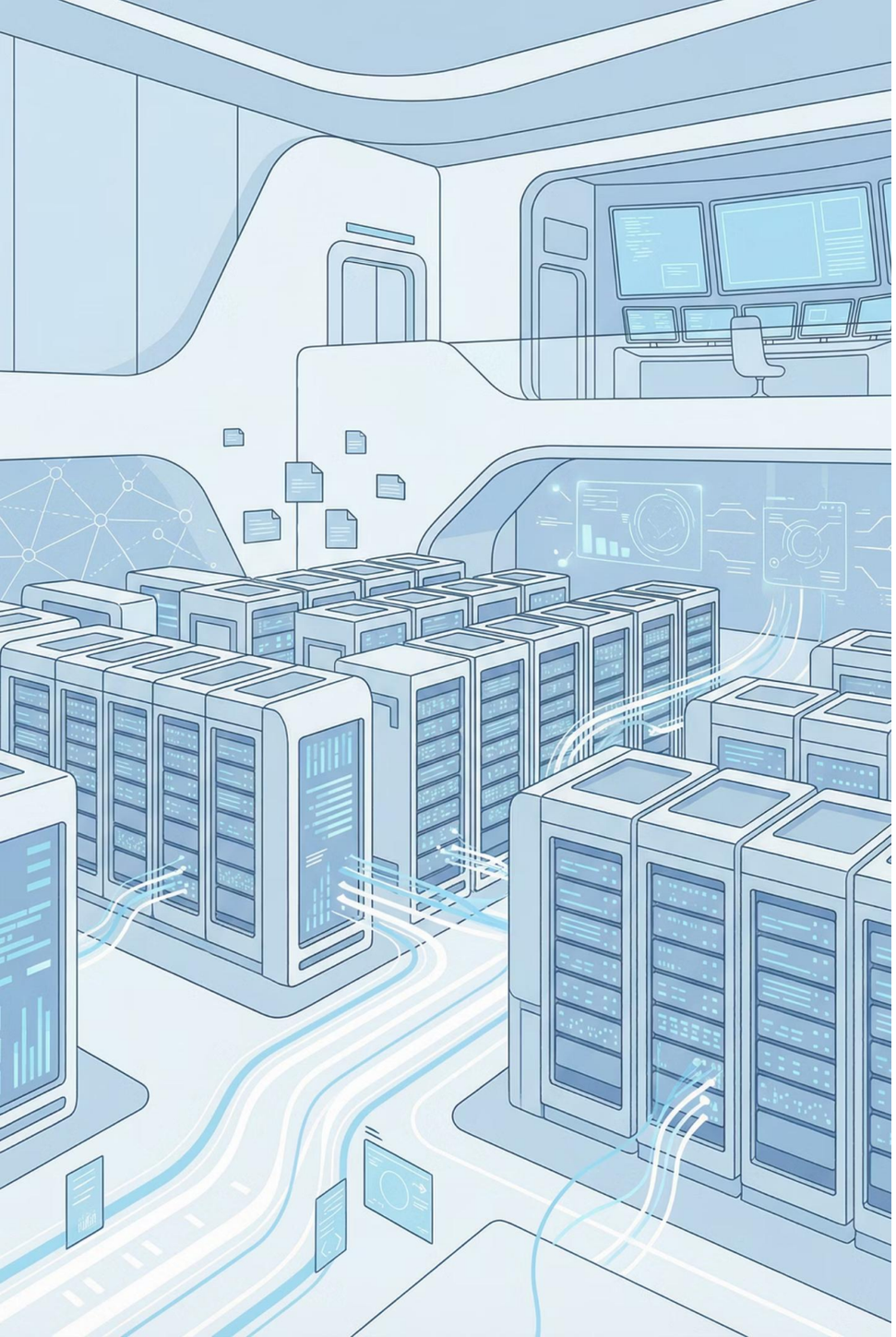
Four Artifacts

📊 Data Model | 💻 Implementation | 📝 Registration | 📖 Metadata

Universal Language

🌐 OData protocol for modern applications

Connects SAP's traditional strengths with the modern world



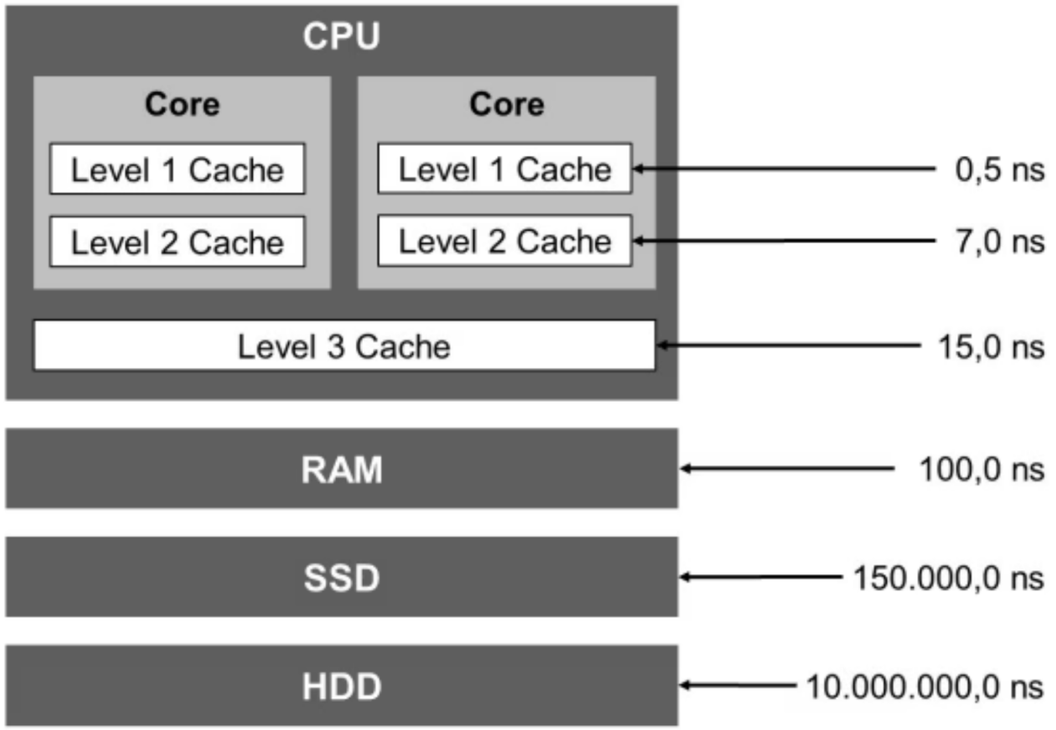
SAP HANA Fundamentals: The Platform That Changed Everything

What If the Database Stopped Being the Bottleneck?

🐌 Traditionally: Slow relational databases = performance problems

⚡ What if that changed completely?

What would happen if the bottleneck disappeared?



The Historical Database Problem

💿 Spinning Mechanical Disks

The slow cycle:

1. System needs data
2. Goes to fetch from disk
3. Disk spins and locates
4. Returns information

Result: Massive bottleneck limiting real-time

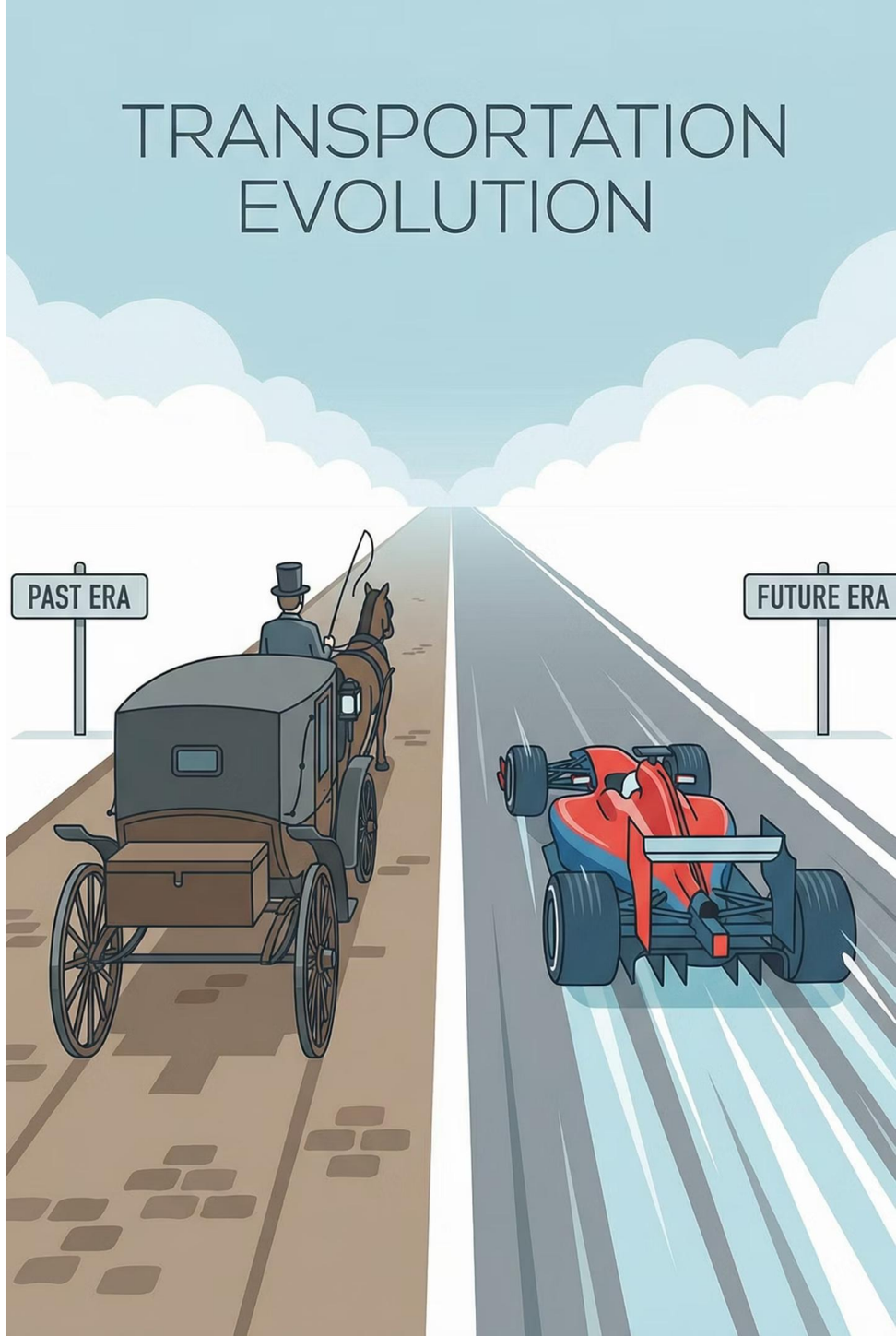
HANA Changes Everything

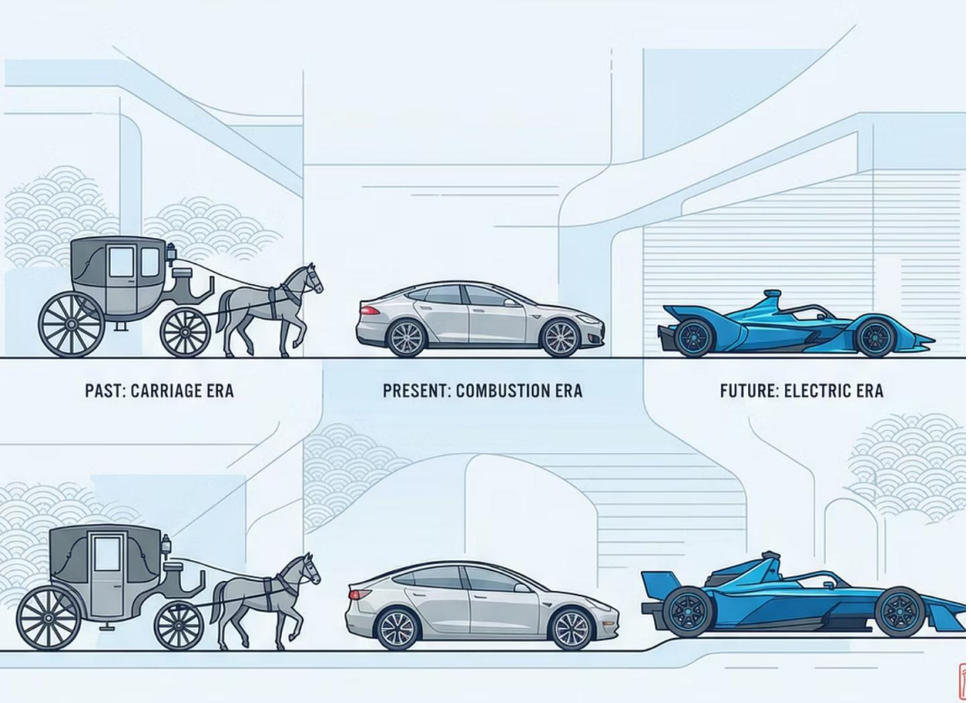
SAP HANA = Modern Hardware Advances

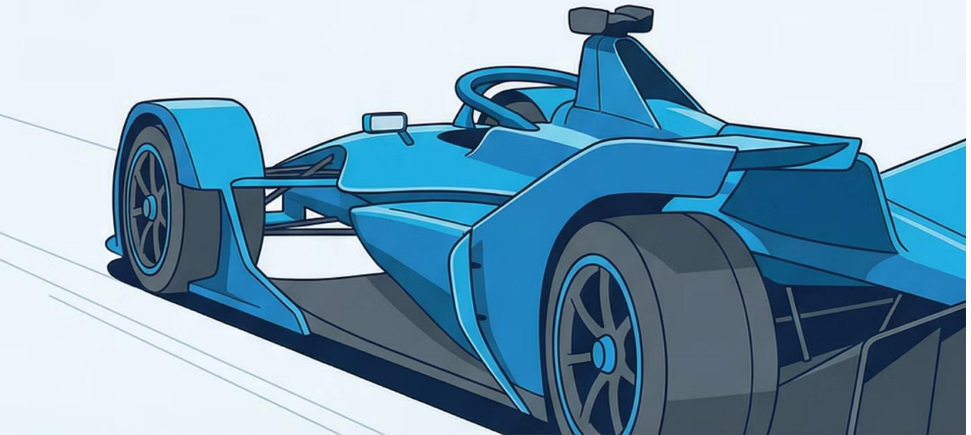
Built to exploit:

- Abundant and affordable memory
- Powerful multicore processors

Completely different performance







From Horse Carriage to Formula 1

🐴 Traditional Database → 🏎️ SAP HANA

Same destination

Revolutionary performance

First Breakthrough: Memory

📊 Before

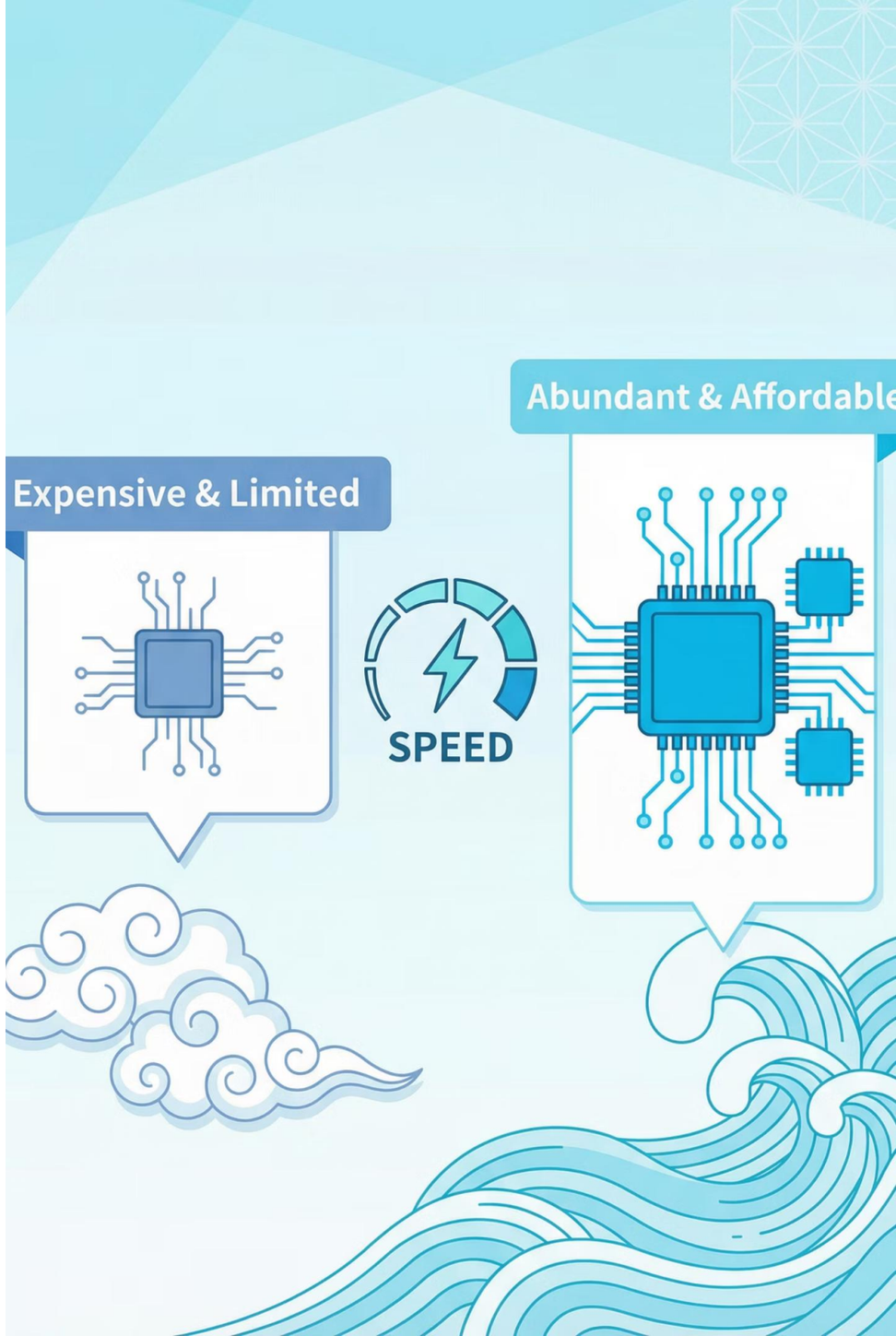
Expensive and limited memory

⚡ Today

Abundant and affordable memory

🚀 HANA = 100% In-Memory

Gigabytes/Terabytes in super-fast RAM, no waiting for mechanical disks



In-Memory: Data at Light Speed

⚡ Everything in RAM

1. Your entire company's data

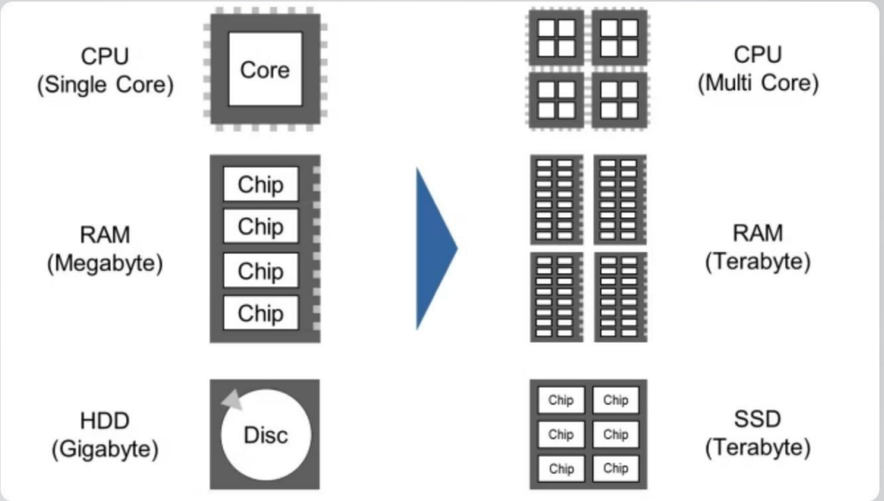
   Terabytes

2. Ultra-fast RAM Access

   Almost instantaneous

3. No mechanical disks

   No waiting



More Than Just a Fast Database

SAP HANA = Complete Platform

4 Core Services:

Database Services

Application Services

Processing Services

Integration Services

Column + Row: Best of Both Worlds

📊 Row Storage

→ OLTP (Transactions)

HANA combines both

- without moving data between systems

📈 Column Storage

→ OLAP (Analytics)

Real Example: Sarah in Retail

🤦‍♀️ Before HANA:

Daily report = hours

Data outdated when ready

⚡ With HANA:

Millions of transactions → Instant report

Real-time price adjustments

From obsolete hours to decisions in the moment

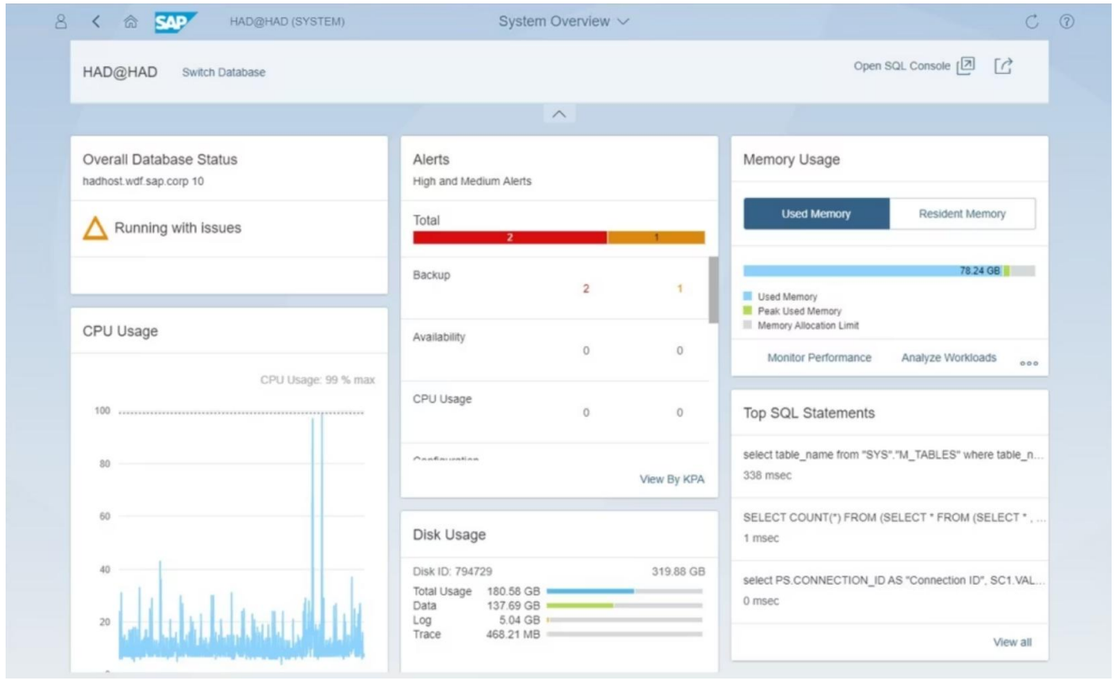
HANA Cockpit: Mission Control

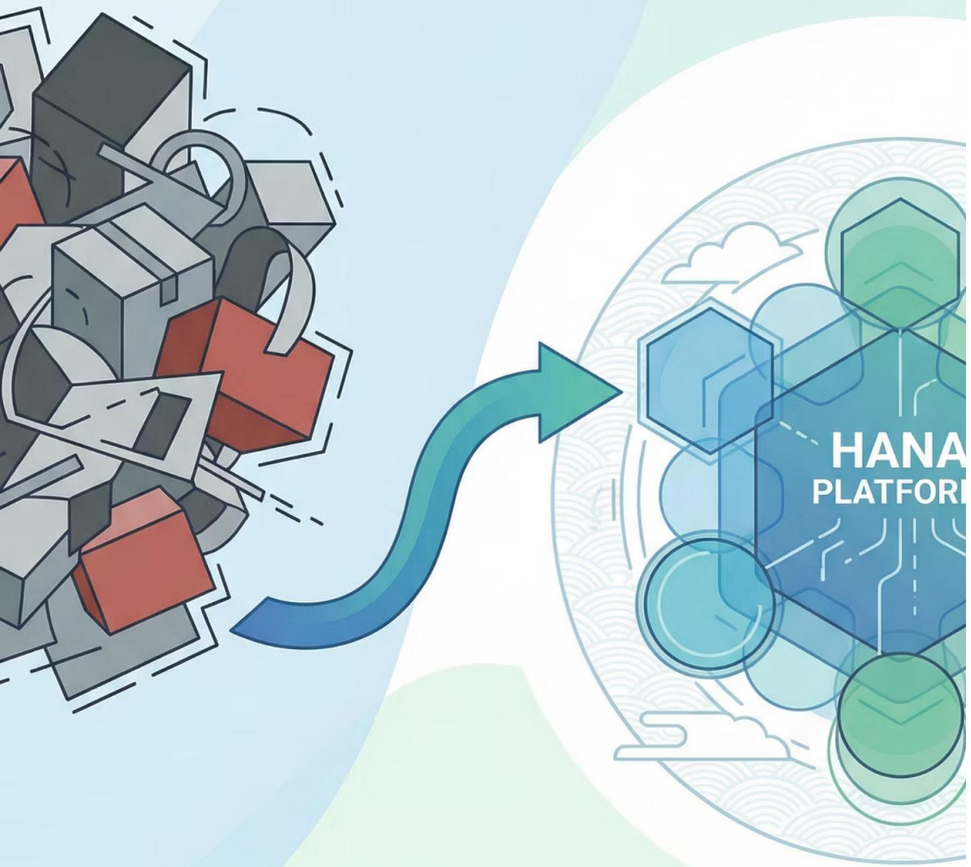
🎛️ HANA Cockpit = Central Hub

Administration:

- Start/stop services
- Monitor system health
- Manage users

🔍 + Database Explorer for advanced SQL





The Big Insight About HANA

⚡ HANA = Simplification Engine

Eliminates:

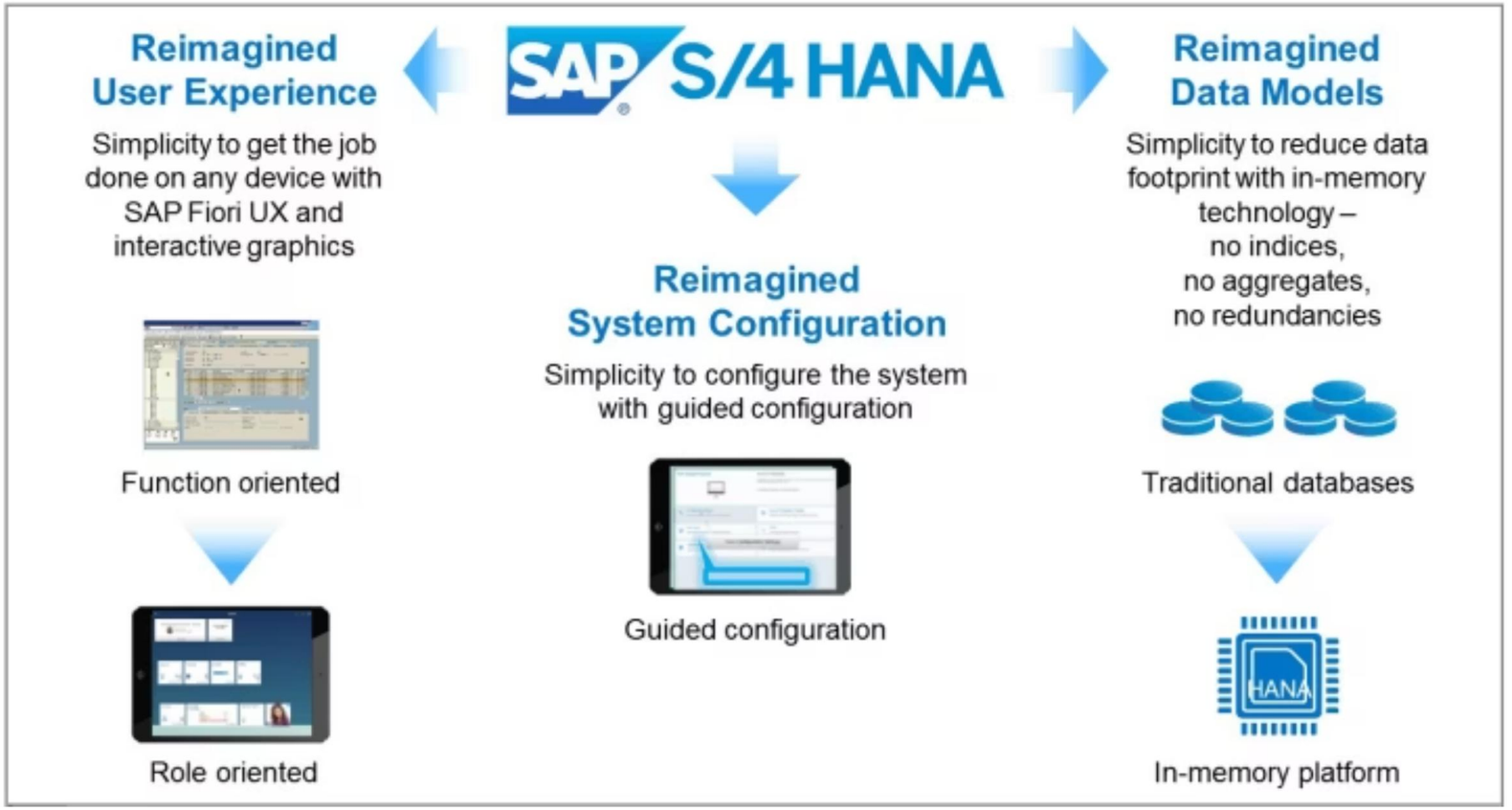
- ❌ Multi-system complexity
- ❌ Batch processing
- ❌ Outdated data

Enables:

- ✅ Unified OLTP + OLAP
- ✅ Real-time insights
- ✅ Instant informed decisions

Real-time GPS vs driving looking in the rearview mirror

S/4HANA: Architecture That Solves Real Problems

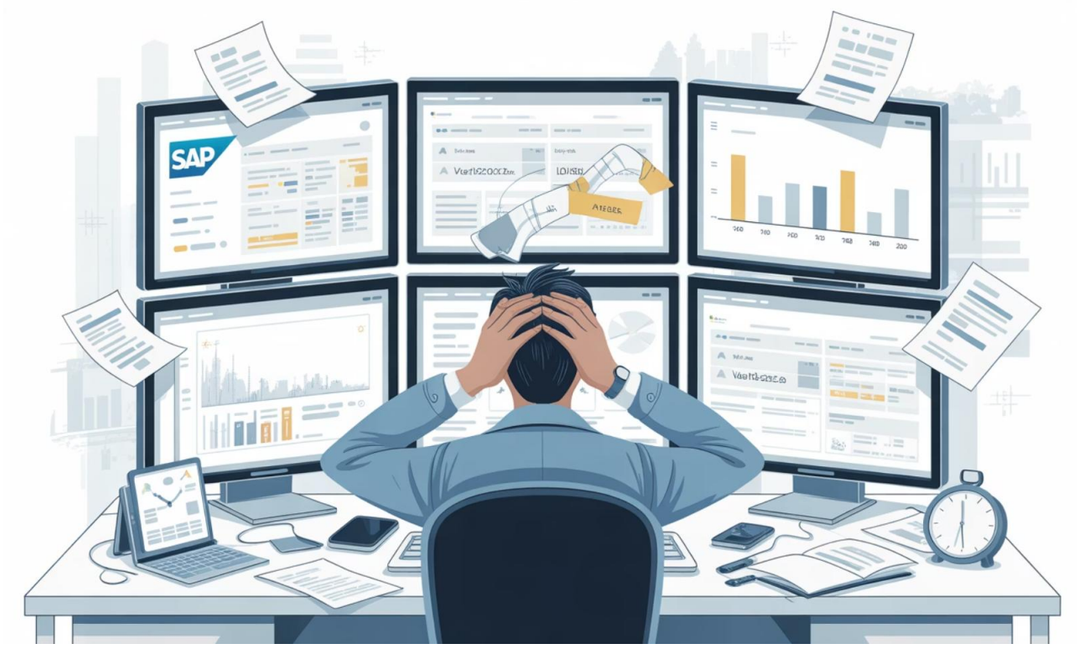


Month-End Close Nightmare

Alright, picture this. You're a finance manager at month-end close

The Traditional ERP Reality:

- 📊 Reconcile FI general ledger
- 💰 Then CO controlling ledger
- 📦 Then materials management
- 💼 Then sales data
- 🔍 Three days hunting $50,000 variance (timing difference!)



Four separate data sources, discrepancies everywhere

The Architecture Problem

Meanwhile, your CEO wants real-time profitability by customer. Impossible

Why Real-Time is Impossible:

- 📱 CEO wants real-time profitability by customer
- 🗄️ Data scattered across 30,000 database tables
- 🌙 Aggregates refresh overnight only
- 🚫 This isn't workflow problem - it's architecture problem

S/4HANA was built to solve this fundamental architecture challenge

Three Simplicity Pillars Transform Everything

By the end of this lesson, you'll understand S4 HANA's three simplicity pillars

Universal Journal

Single source of truth

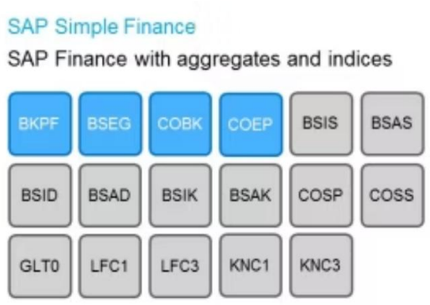
Embedded Deployment

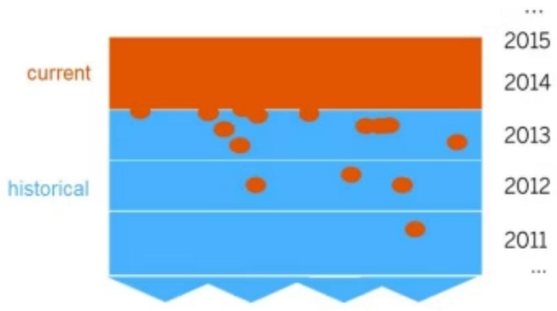
Integrated analytics

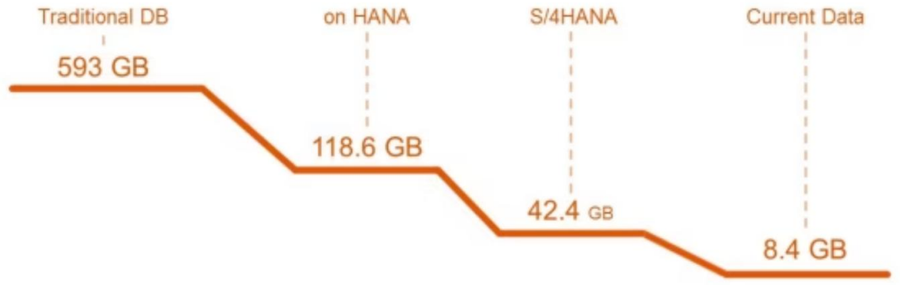
Data Aging

Intelligent storage management

These pillars enable measurable ROI and business process transformation







The Business Impact

Legacy ECC systems require five to 10-day month-end closes with manual reconciliations

ECC vs. S/4HANA Performance:

- ❌ ECC: 5-10 day month-end closes, manual reconciliations
- ✅ S/4HANA: Real-time with single source of truth
- ⚡ Close cycles drop to hours
- 💰 Compliance costs drop by millions

S/4HANA does month-end close in real-time with single source of truth

HANA: The Speed That Makes It Possible

S4 HANA is what happens when you redesign the entire SAP business suite

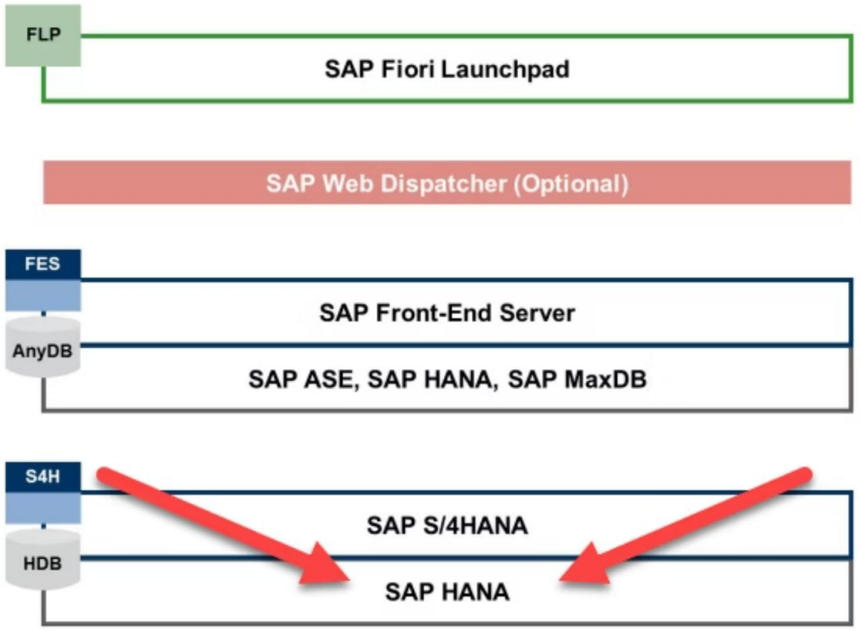
HANA's in-memory platform

Transactions and analytics simultaneously

Live data processing

S/4HANA exploits this power completely

Without HANA speed, these simplifications wouldn't be possible

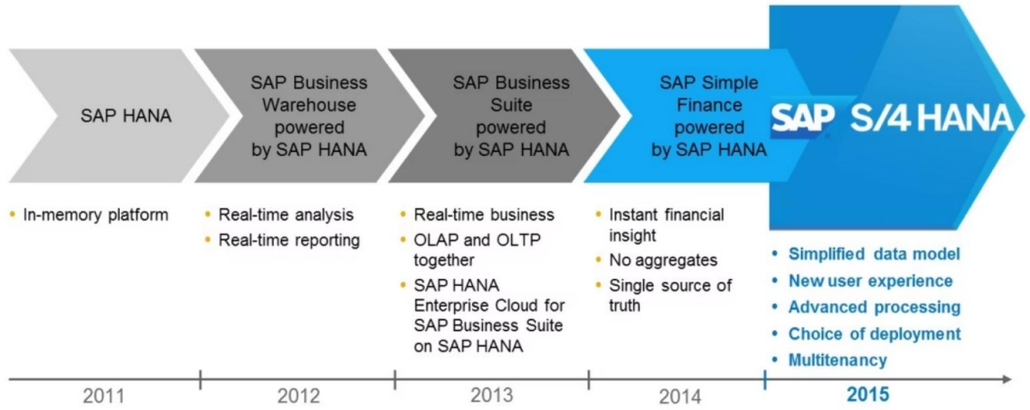


The Digital Core for Intelligent Enterprise

S4 HANA launched in 2015 as SAP's next generation ERP

S/4HANA Positioning:

- 🚀 Launched 2015 as next-generation ERP
- 🏢 Digital core for intelligent enterprise
- 🎯 HANA exclusive - only runs on SAP HANA
- 🚫 No Oracle, no DB2, no exceptions



The digital core for what SAP calls the intelligent enterprise

ECC: Decades of Accumulated Complexity

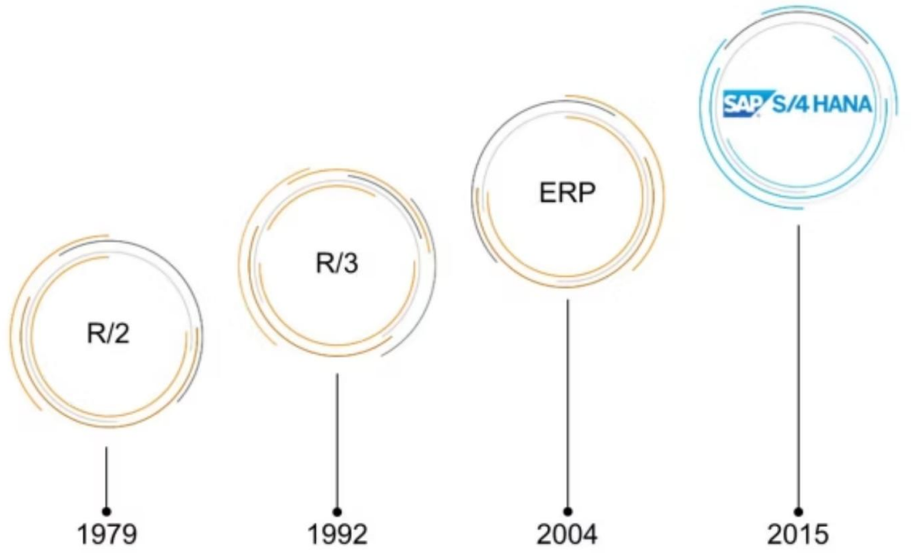
The problem is decades of accumulated complexity in SAP ECC

01

40+ Years of Evolution Problems:

- 📈 ECC evolved over 40+ years
- 🗄️ Added modules, tables, redundant structures
- 💰 Finance: separate FI and CO ledgers
- 📊 Sales data duplicated across SD and billing
- 🐌 Pre-calculated aggregates (raw data too slow)

02



Result: 30,000+ database tables, reconciliation nightmares

IT Budget Reality: 30-40% Just Keeping Lights On

The result, over 30,000 database tables, reconciliation nightmares, and 30 to 40% of IT budgets

The Maintenance Trap:

- 🗄️ 30,000+ database tables
- 😤 Reconciliation nightmares
- 💸 30-40% IT budget on maintenance
- 🚫 Not innovating - just keeping lights on

30-40% of IT budgets spent keeping lights on instead of innovating

What This Means for Everyone

Here's what this means for you

Universal Benefits:

Consultants

Pitch migrations with concrete ROI

Developers

Unified data models, no spaghetti relationships

Executives

Live KPIs on phones, not yesterday's reports

Architects

50% smaller landscapes

S/4HANA reduces table count by half, hardware footprint by 60%

Pillar 1: Universal Journal (ACDOCA)

Pillar one, Universal Journal. This is the single source of truth

Single Source of Financial Truth:

- 📖 Table name: ACDOCA
- ✅ Single source for all financial and management accounting
- ❌ ECC: Separate FI, CO, AA, ML tables
- ⚡ S/4HANA: One posting, multiple dimensions

Every financial transaction posts here once with all dimensions

Why Universal Journal Works Now

Why is this possible now? HANA can handle billions of line items in memory

HANA Makes It Possible:

- 🧠 HANA handles billions of line items in memory
- 🐌 Traditional databases would choke on single wide table
- 🚫 No more reconciliation needed
- ⚡ FI and CO always in sync (same data)

Month-end close drops from days to hours

Pillar 2: Embedded Deployment

Pillar 2. Embedded Deployment In ECC landscapes, you had separate systems for everything

From Separate Systems to Embedded:

- ❌ ECC: Separate ECC, BW, PI, Gateway, Solution Manager
- ✅ S/4HANA: Analytics, planning, integration embedded in core
- 🔄 Uses HANA's platform services
- 📊 Analytics on live transactional data

One system instead of five, eliminating data replication

Embedded Benefits and Trade-offs

Why embedded? It eliminates data replication

The Embedded Advantage:

- 🚫 Eliminates data replication
- 💰 Reduces costs (one system vs. five)
- 🔒 Simplifies security (one user base, one authorization)
- ⚖️ Trade-off: More powerful hardware needed for core
- 📉 Total landscape cost drops 40-60%

One user base, one authorization model, 40-60% cost reduction

Pillar 3: Data Aging

Pillar 3. Data aging. HANA RAM is expensive

HANA RAM expensive

Don't want 20 years of closed POs in active memory

Automatically moves old data to cheaper storage

Queries span hot and cold data transparently

Keeps active dataset lean while maintaining compliance access

Warehouse vs. Distribution Center Analogy

Think of it like this. ECC is a cluttered warehouse

Perfect Analogy:

- 🏭 ECC: Cluttered warehouse, inventory scattered across buildings
- 🚚 Need forklift to move between them (reconciliation)
- 💸 Paying rent on 10-year-old boxes
- 🏬 S/4HANA: Organized distribution center, everything in one place

Automated conveyor belts, smart off-site storage for old items

Custom Code Compatibility

Here's what confuses people. Does S4 HANA break my custom code?

Migration Reality:

- ✅ Mostly no - SAP provides compatibility views
- 🔄 Virtual tables mimic old ECC structures
- 💻 Existing ABAP code keeps working
- 🔧 Rework needed: Custom aggregates, batch reconciliation jobs

Not rewriting everything, but adapting to the new paradigm

Innovation Platform, Not Just Speed

But simplicity in S4 HANA doesn't mean fewer features

Better architecture, not fewer features

HANA speed eliminates redundancy

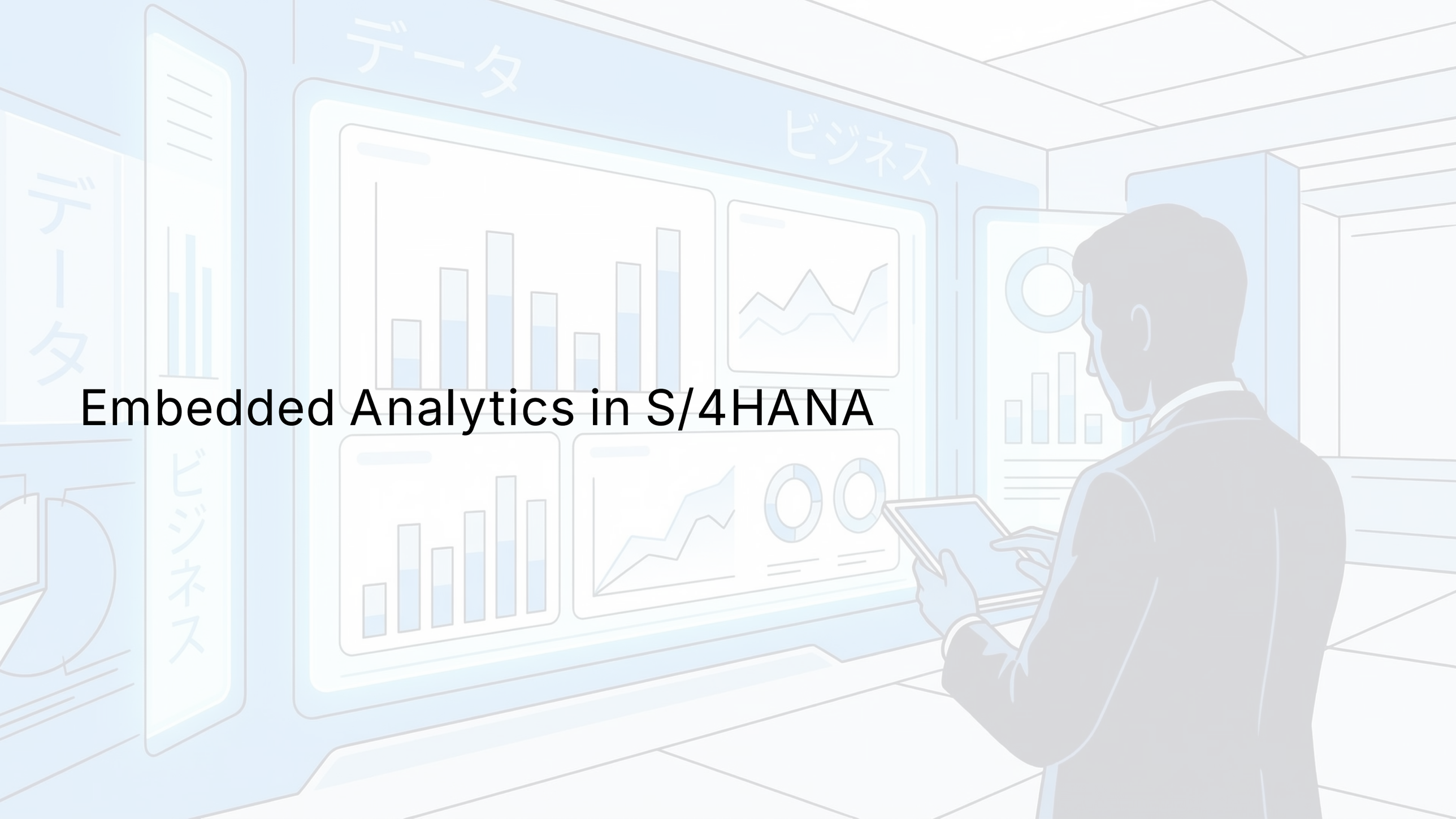
One journal = unified truth for decisions

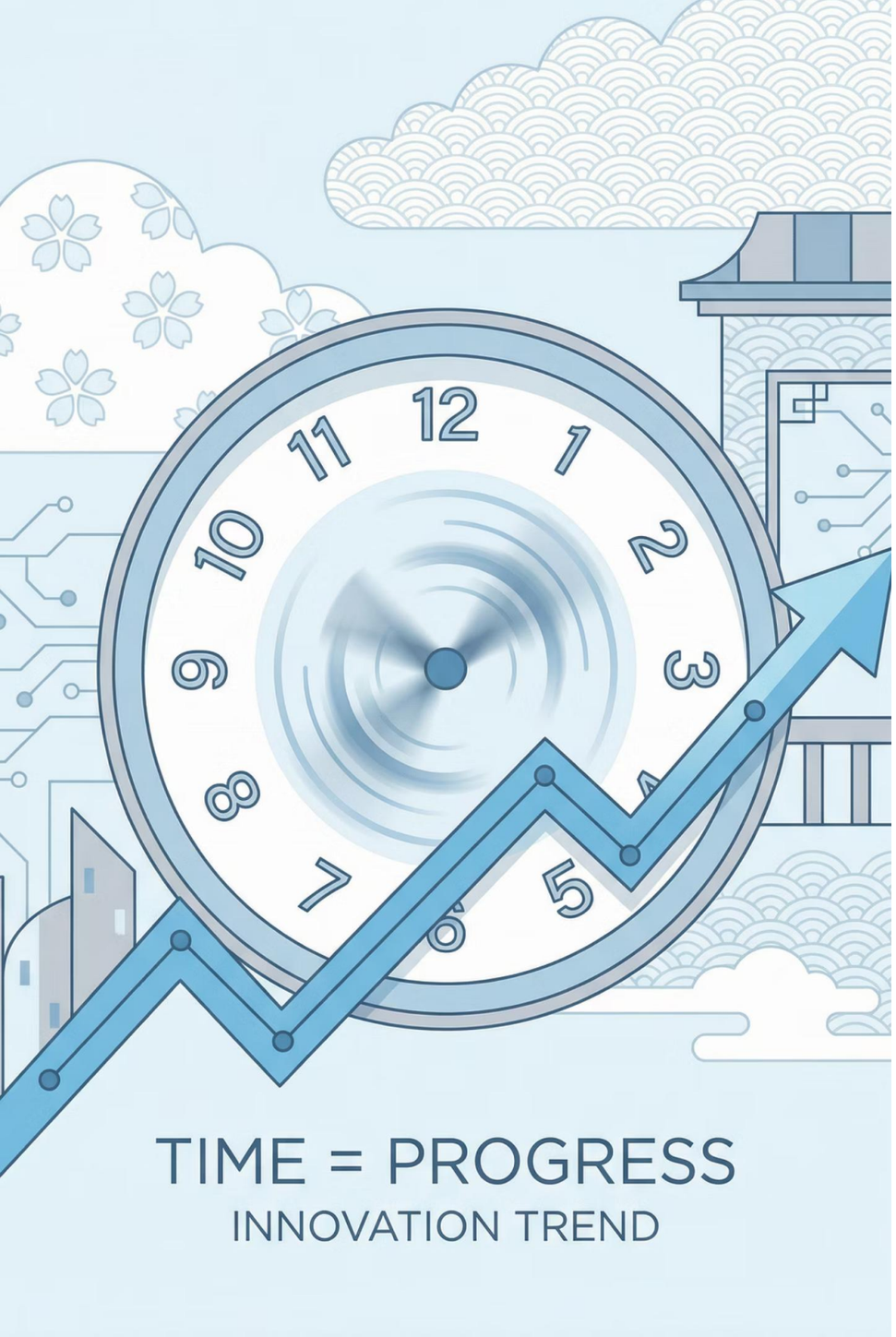
Embedded deployment = agility without silos

Aging = sustainable growth

S/4HANA becomes innovation platform you extend

Embedded Analytics in S/4HANA





Data from Now, Not Yesterday?

What if your analytics were live, reflecting business reality second by second?

This is the core promise of S/4HANA Embedded Analytics.

The Old Way: Separate Systems

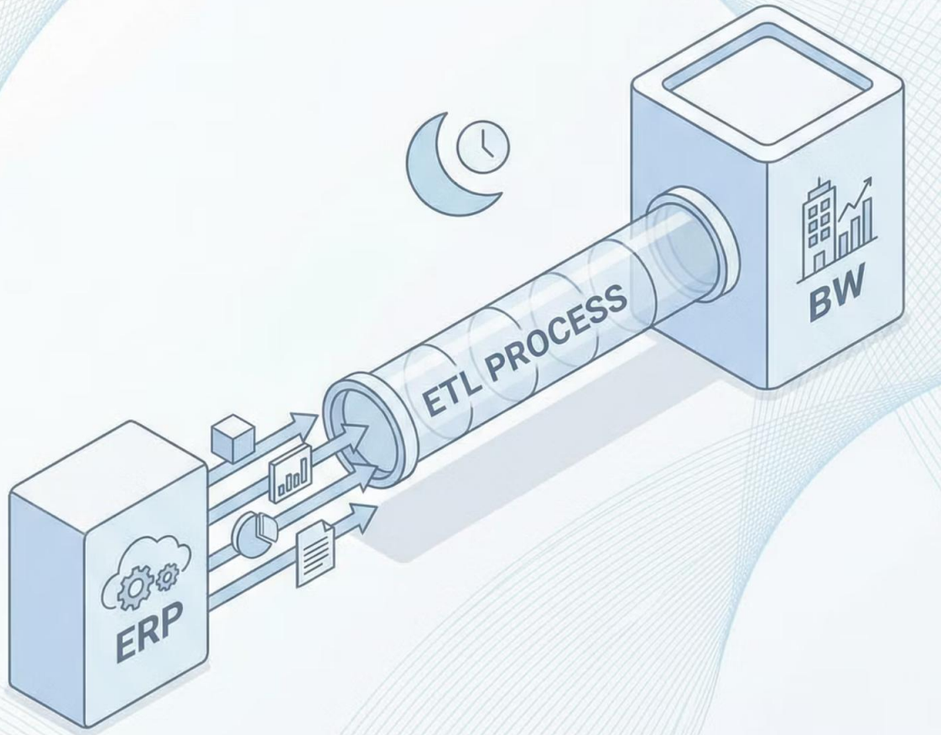
Transactional System (ERP)

- Optimized for day-to-day operations
- Running the business

Analytical System (BW)

- Optimized for reporting & queries
- Understanding the business

These systems were separated because traditional databases couldn't do both jobs well.



The Nightly Data Shuffle (ETL)

1 Extract

Pull data from the transactional system.

2 Transform

Re-structure the data for analysis.

3 Load

Push the transformed data into the warehouse.

This entire process created a mandatory delay between operations and insights.

The Problems with Separation

❌ Data Latency

Reports were always at least one day old.

❌ High Complexity

Two systems to maintain, patch, and secure.

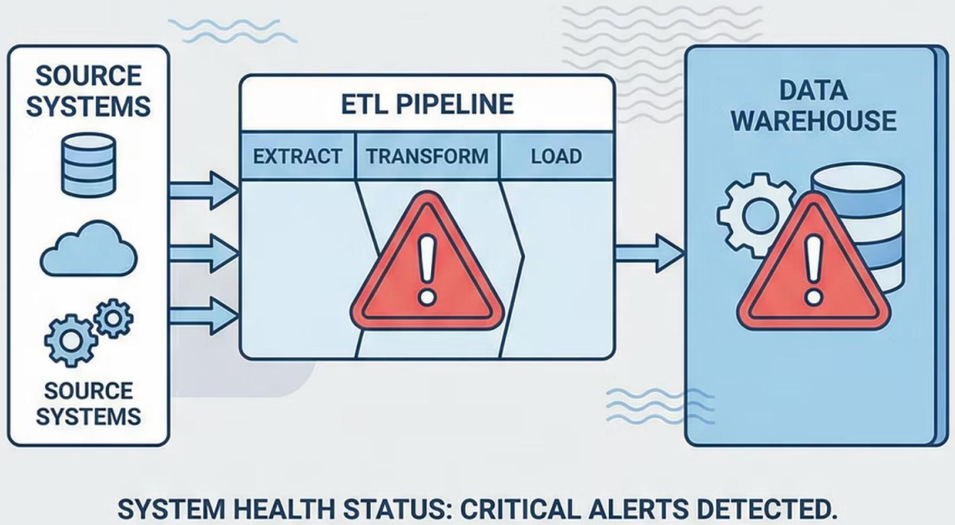
❌ Synchronization Issues

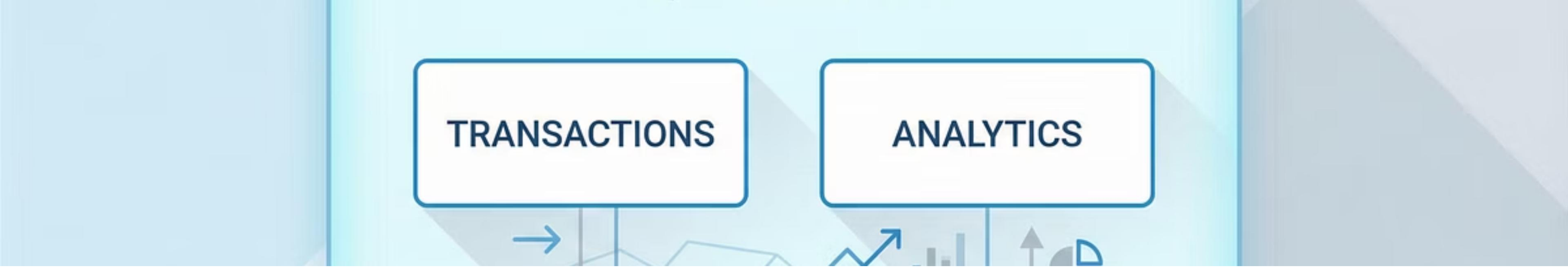
ETL processes could fail or be inconsistent.

❌ High Cost

Separate licenses, hardware, and operational staff.

The architecture solved a technical limitation but created major business challenges.





The New Way: A Single System

S/4HANA combines both workloads in one database.

Transactions (OLTP)

Analytics (OLAP)

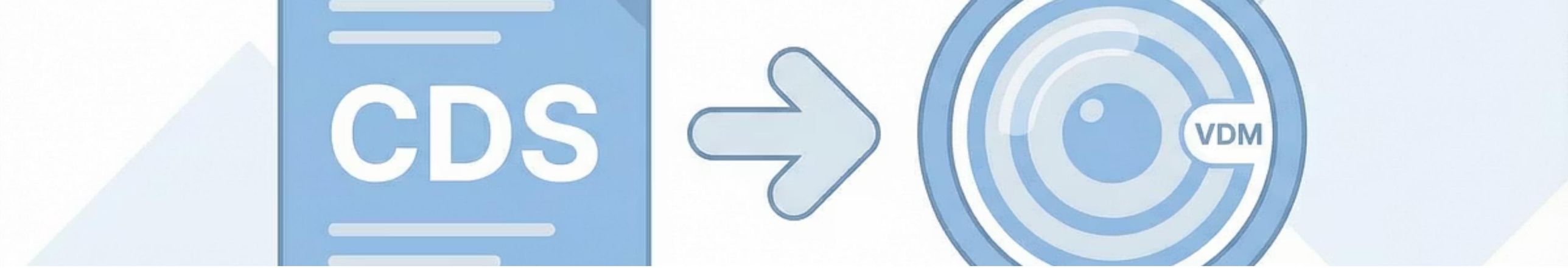
HANA's in-memory, columnar architecture allows live queries without slowing down the business.

The Secret: Virtual Data Models

VDMs are analytical "lenses" on live data.

- They are just definitions, not stored data.
- They calculate results "on-the-fly".
- They turn raw tables into meaningful insights.

VDMs make analytics possible without physically copying data.



The Building Block: CDS Views

Core Data Services (CDS) are the code behind VDMs.

Powerful Language

A powerful SQL-like language to define data models.

Maximum Speed

They execute directly in HANA for maximum speed.

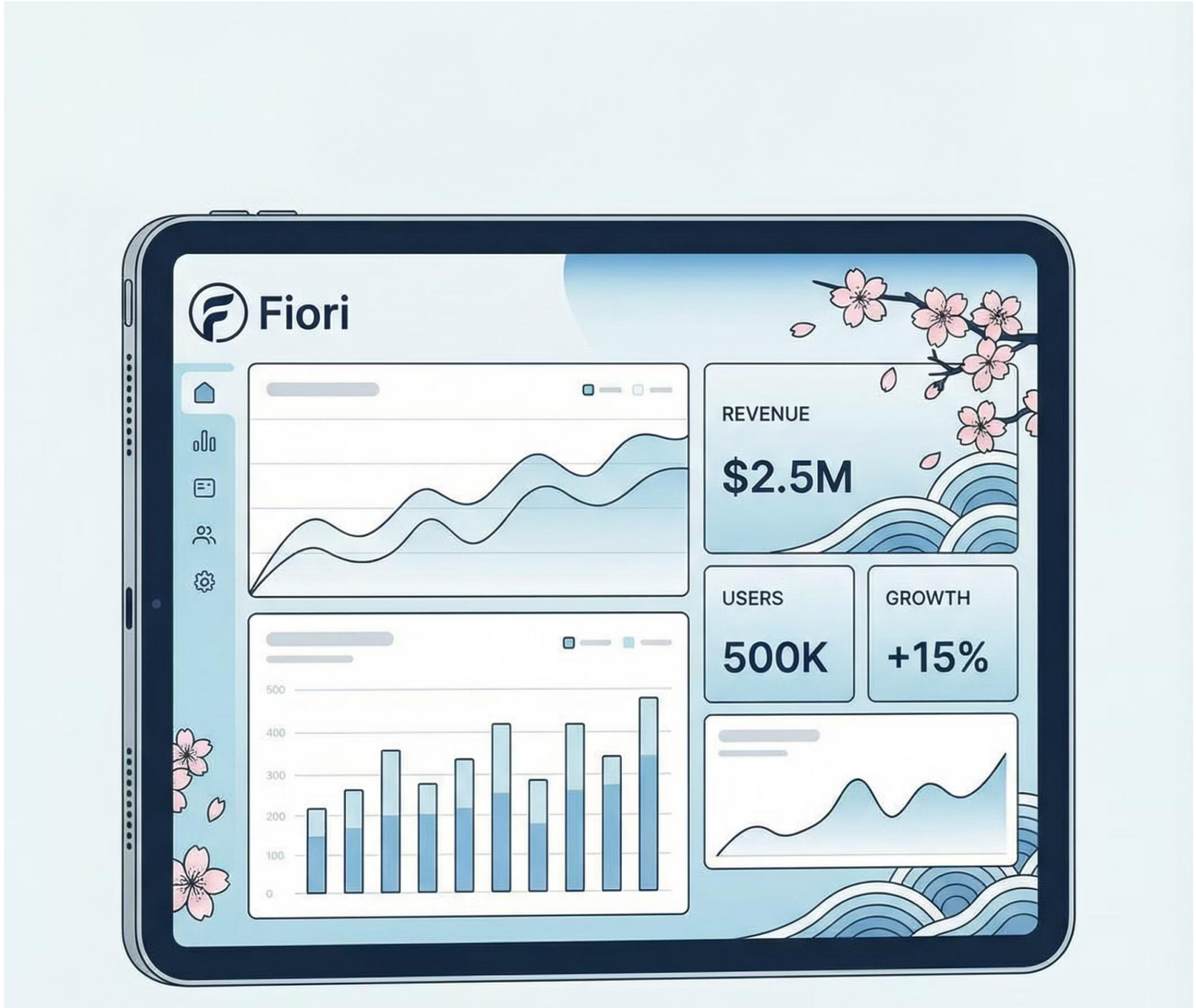
Pre-built Views

SAP delivers thousands of pre-built views.

CDS Views are the developer's tool to implement the "Code Pushdown" paradigm.

The User Experience: Fiori Apps

Users consume real-time data through role-based Fiori apps.



- Dashboards, reports, and interactive charts.
- Tailored to specific jobs (e.g., Sales, Finance).
- Zero latency between transaction and insight.

Analytics become part of the daily workflow, not a separate task.

When is a Warehouse Still Needed?

Embedded analytics is for *operational* reporting, not everything.

Keep a Warehouse for:

✅ Multiple Systems

Combining data from multiple SAP & non-SAP systems.

✅ Historical Data

Long-term historical data archiving (many years).

✅ Complex Models

Extremely complex, specialized analytical models.

Embedded analytics reduces the need for a warehouse; it doesn't eliminate it entirely.

The Business Impact

⚡ Faster Decisions

Respond to opportunities as they happen.

⚙️ Reduced Complexity

Fewer systems to manage and maintain.

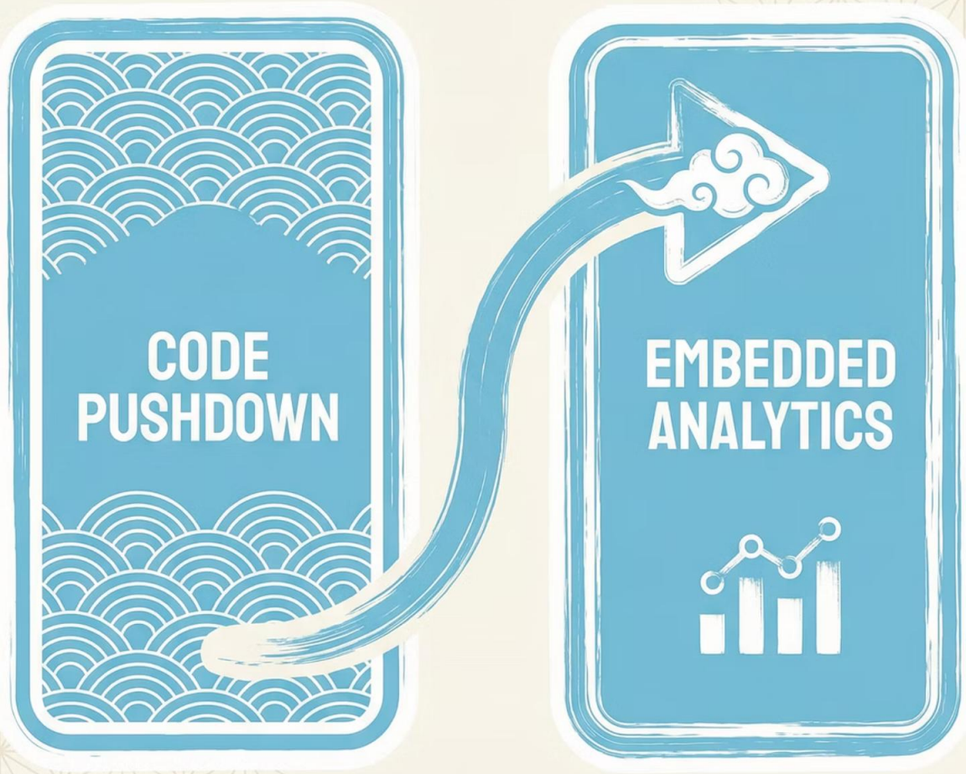
💰 Lower TCO

No separate BI infrastructure, licenses, or storage.

👍 Better Adoption

Analytics are integrated directly into user workflows.

Real-time insight is a competitive advantage.



From Analytics to Architecture

Embedded Analytics is one result of a bigger paradigm: Code Pushdown.

We've seen *what* it does. Now we'll explore *how* it's possible.

The Code Pushdown Paradigm

Where Should Logic Live?

A fundamental question for all modern SAP development.

The choice architects make here defines the performance of the entire system.

The Traditional Model

Data

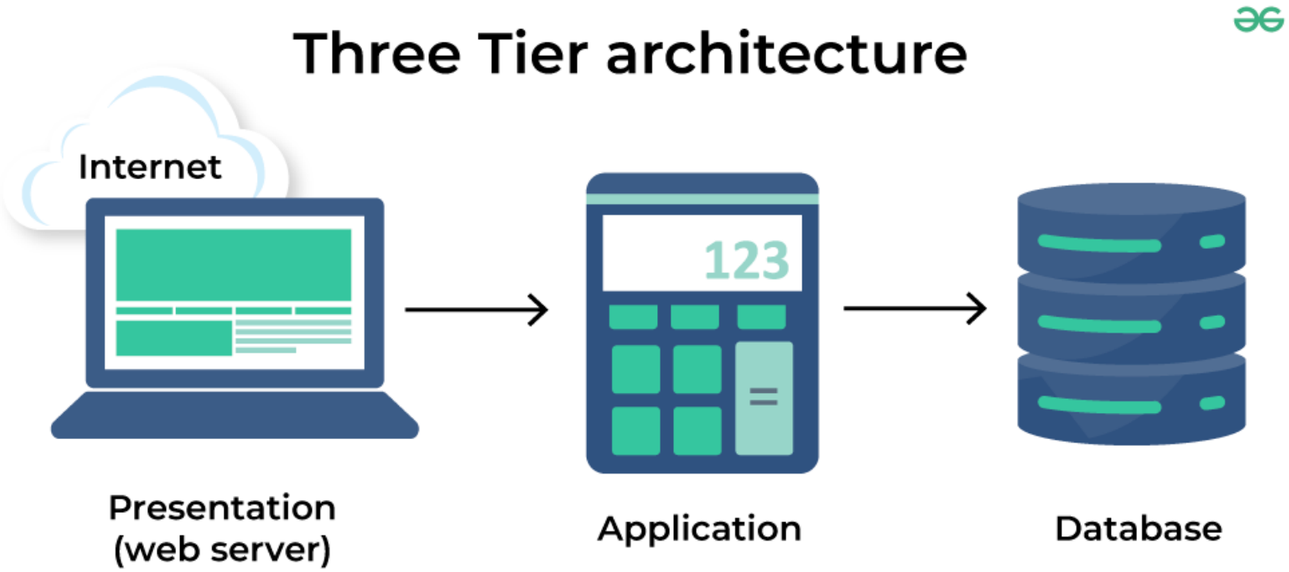
is pulled from the Database

Logic

is processed on the Application Server

Results

are sent back to the Database



This created a constant back-and-forth flow across the network.

Moving Data to Code

01

Query

App server requests data.

02

Transfer

Massive data volume travels over the network.

03

Process

App server calculates, filters, and aggregates.

04

Return

Results (and sometimes data) sent back.

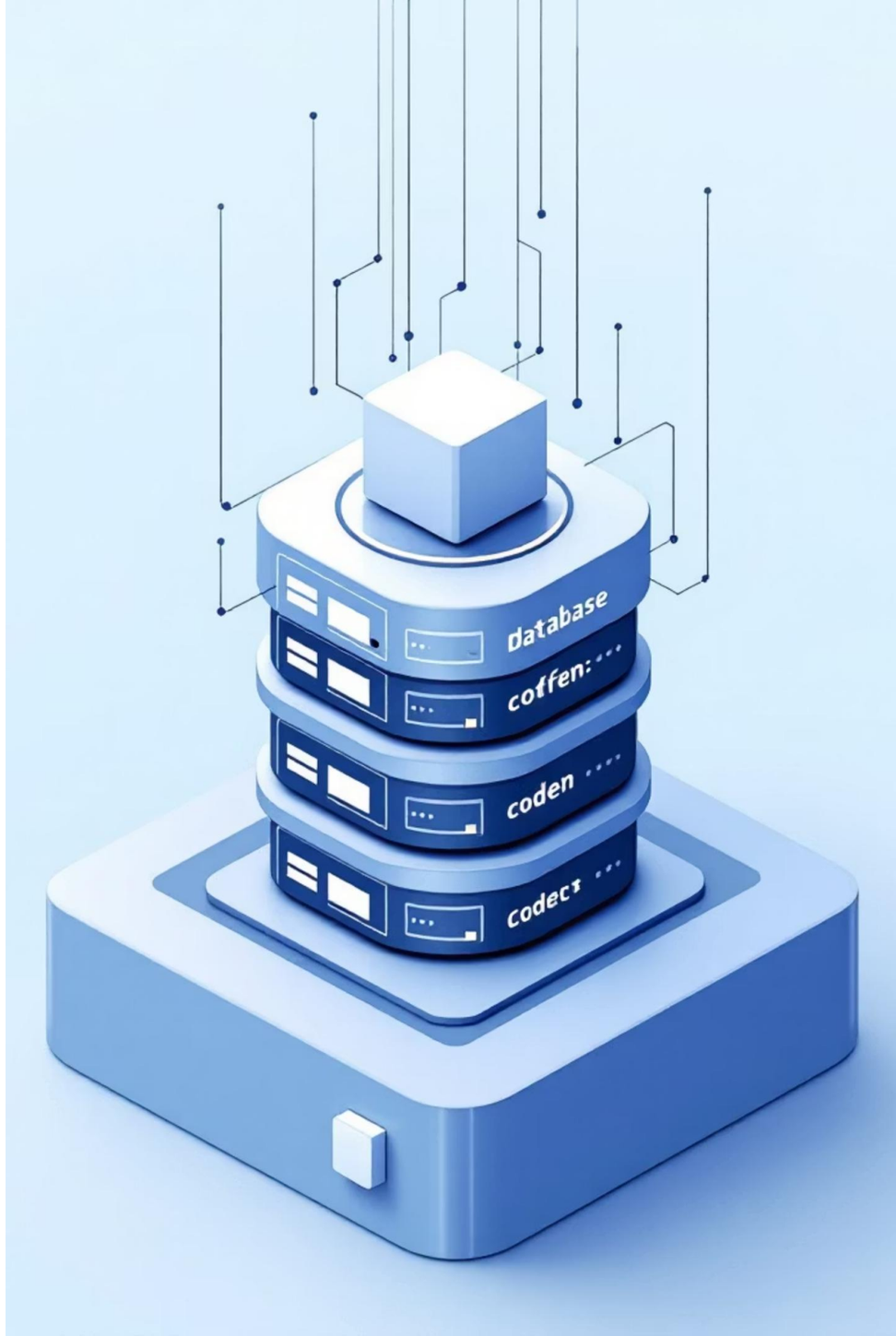
The bottleneck is the network transfer, not the processing.

The New Paradigm: Code Pushdown

Instead of moving data to the code...

Move the code to the data.

Logic executes directly inside the database, where the data already lives.



Why HANA Enables This

🚀 Processing Power

Advanced calculation engines.

⚡ In-Memory Speed

Eliminates network latency.

⛓️ Parallel Processing

Splits operations across multiple cores.

📊 Columnar Storage

Only touches the columns it needs.

HANA is not just a database; it's a powerful computing platform.

What Logic to Push Down

- Aggregations

  Sums, averages, counts.

- Filtering

  Applying WHERE clauses at the source.

- Joins

  Combining multiple tables inside HANA.

- Analytics

  Ranking, windowing, and statistical functions.

If it processes large volumes of data to produce a small result, push it down.

How Developers Implement This

CDS Views

Define data models and calculations.

AMDP

Write SQLScript within the ABAP environment.

SQLScript

Native HANA procedures for maximum control.

OpenSQL

Modern ABAP syntax that leverages HANA features.

These are the primary tools for building modern, performant S/4HANA applications.

The Benefits of Code Pushdown

Performance

Drastically faster applications.

Network Traffic

Only small result sets are moved.

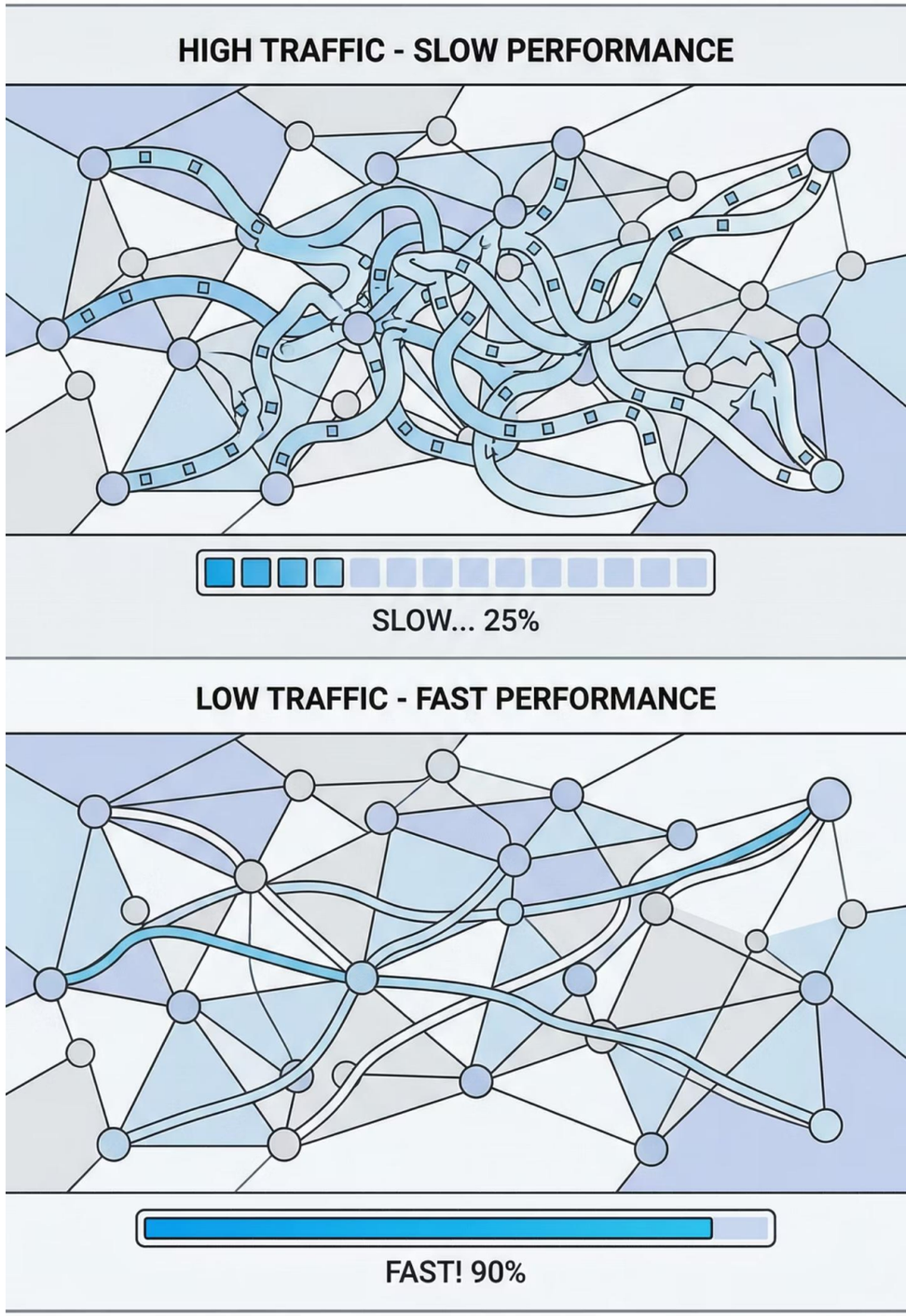
Scalability

Utilizes HANA's parallel processing.

Resources

Frees up Application Servers for UI and workflow tasks.

This results in a simpler, faster, and more efficient architecture.



A New Way of Thinking

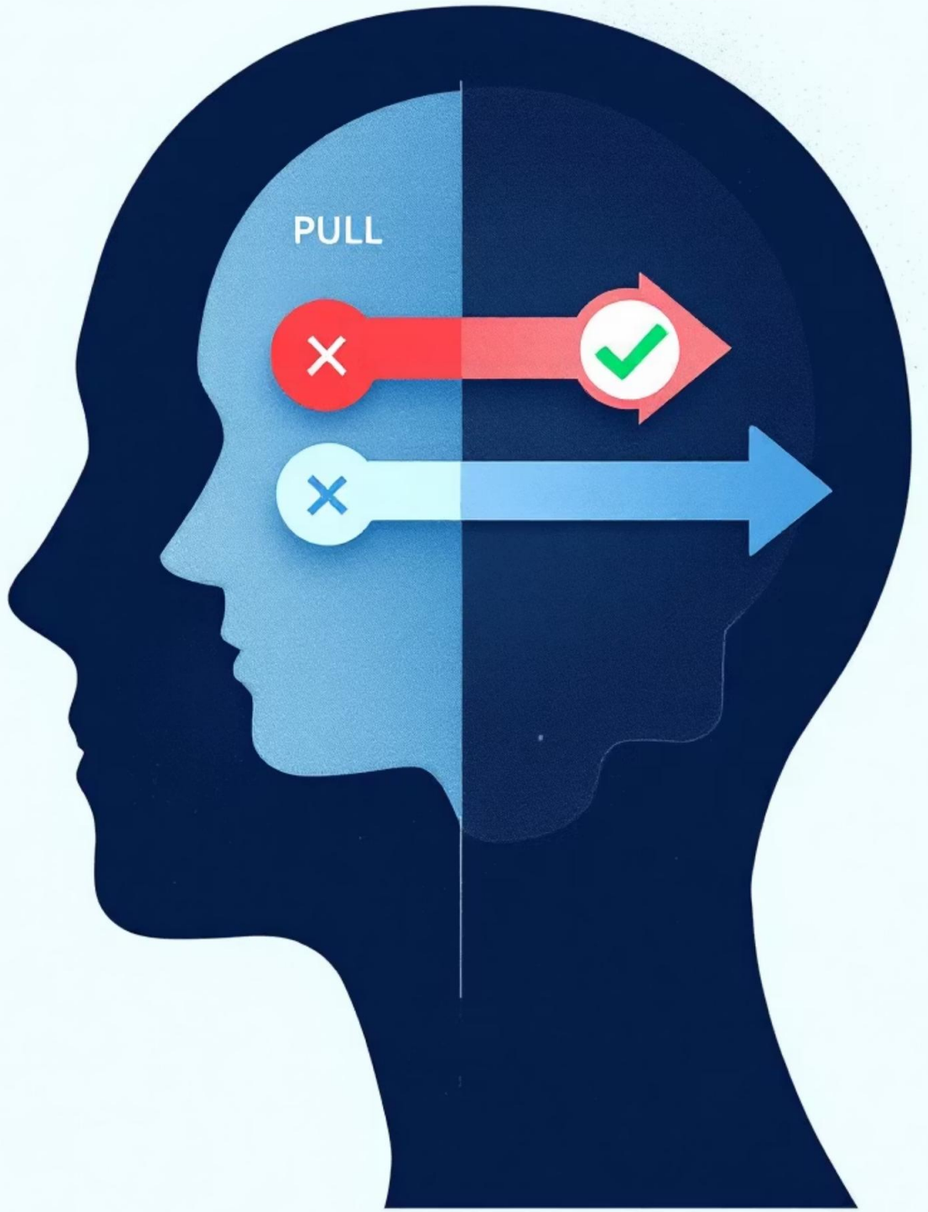
❌ Old Way

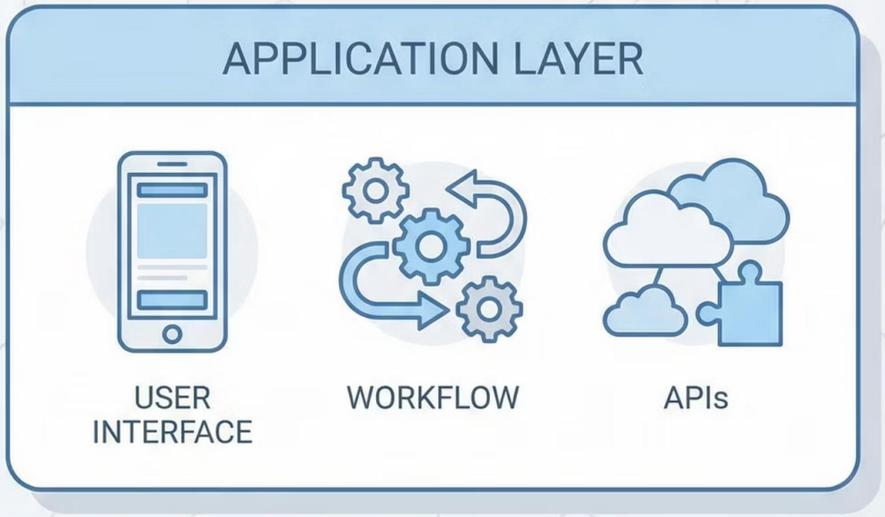
"Pull all the data I need, then process it in ABAP."

✅ New Way

"What logic can I push down to the database to run first?"

The developer's primary task shifts from data manipulation to data modeling.





What *Not* to Push Down

User Interface Logic

Managing screens and user interaction.

Workflow Orchestration

Multi-step business processes.

External Integrations

Calling outside systems via APIs.

The art is knowing what to push down and what to keep in the application layer.

The Fundamental Shift

From:

Moving DATA to the code.

❌

To:

Moving CODE to the data.

✅

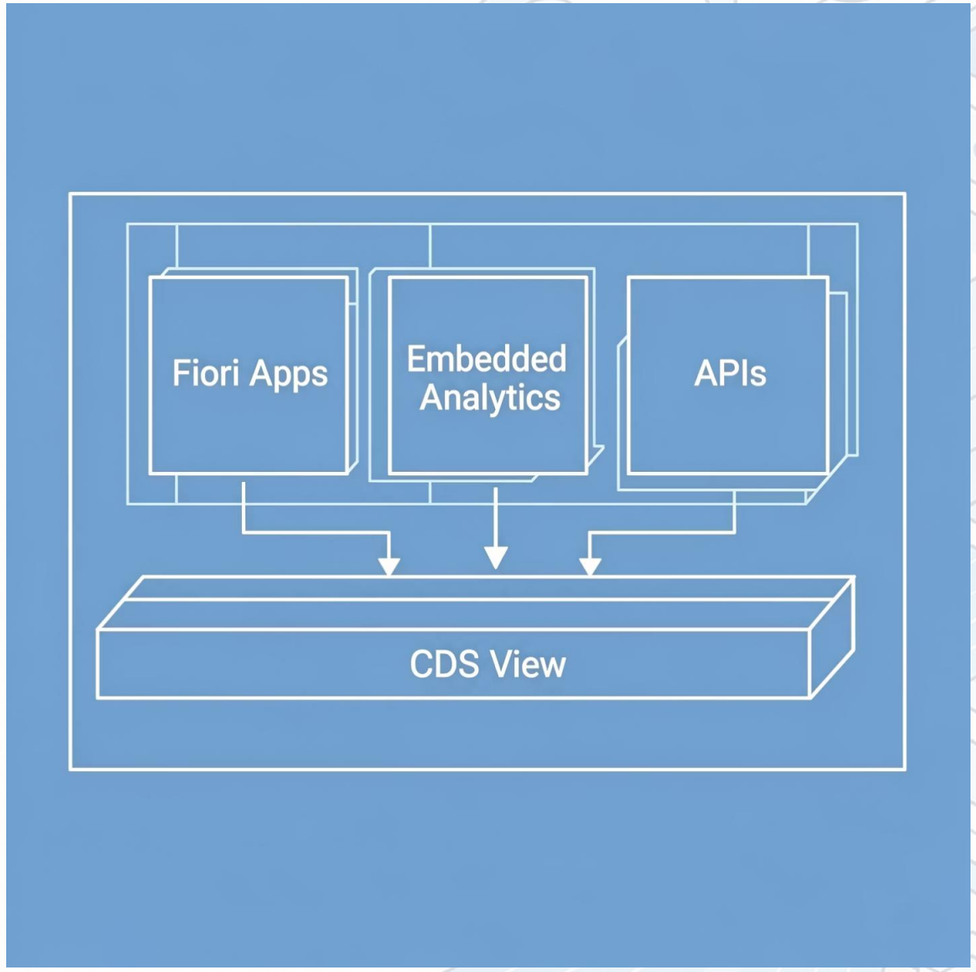
This is the core principle of modern SAP development on HANA.

What's Next?

A deep dive into the most important code pushdown technology.

CDS views are the foundation of modern data modeling and embedded analytics in S/4HANA.

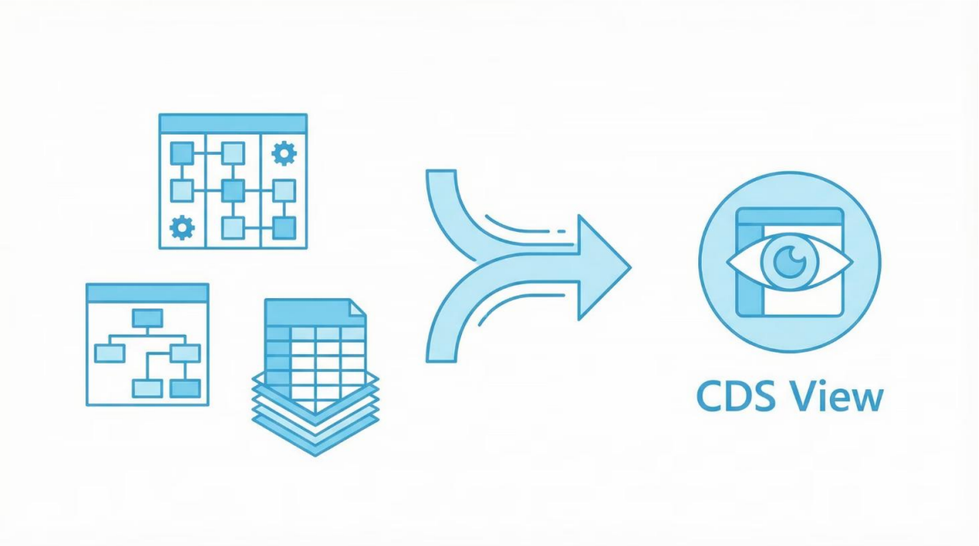
ABAP CDS Views



The Foundation of Modern SAP

The single most important data modeling technology in S/4HANA.

They are the bridge between your raw database tables and your applications.

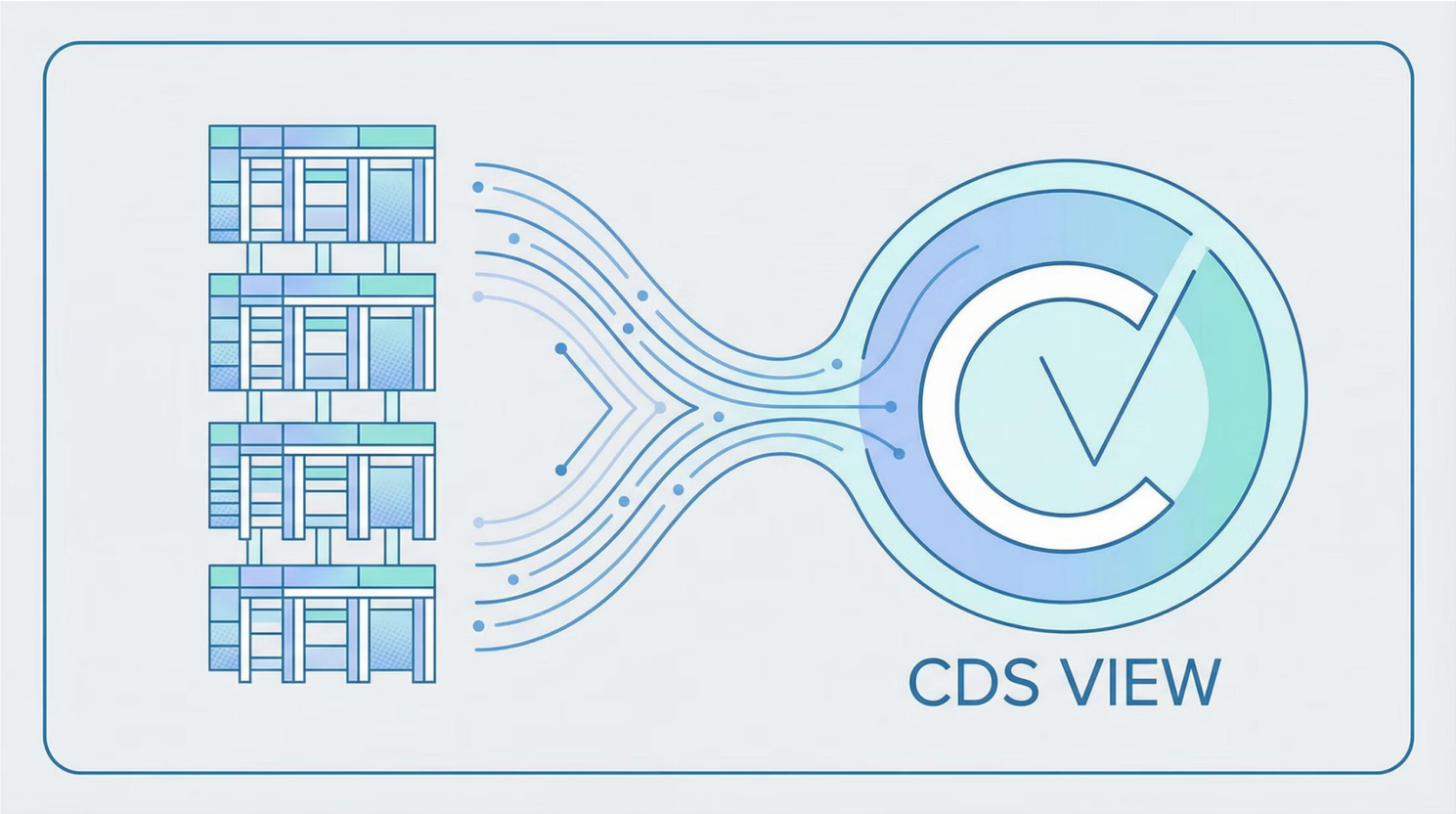


What is a CDS View?

A virtual layer over raw data tables.

- ✅ Joins tables together
- ✅ Calculates new fields
- ✅ Filters for relevant data

You define a meaningful "view" of your data that lives in the database.



Code Push-Down in Action

Logic moves to the data, not the other way around.

1

Instead Of:

Pulling data to the ABAP layer to process.

2

With CDS:

Processing happens directly in the HANA database.

This minimizes data movement and leverages HANA's massive processing power.

More Than Just SQL

CDS Views use Data Definition Language (DDL) to add more power.

Associations:

Defines relationships between views.

Annotations:

Adds semantic meaning (UI, security).

Parameters:

Creates reusable, flexible views.

Calculations:

Embeds business logic directly.

It's a declarative language: you describe *what* you want, and HANA figures out the *how*.

A View for Every Purpose

Basic Views:

Select & clean data from raw tables.

Composite Views:

Combine other CDS Views to build up logic.

Analytical Views:

Designed for aggregations & BI scenarios.

Transactional Views:

Optimized for fast single-record access.

This enables a clean, layered architecture for your data models.

The Power of Annotations

Annotations are metadata tags that give data meaning.

🏷️ @UI:

How should this field look in a Fiori app?

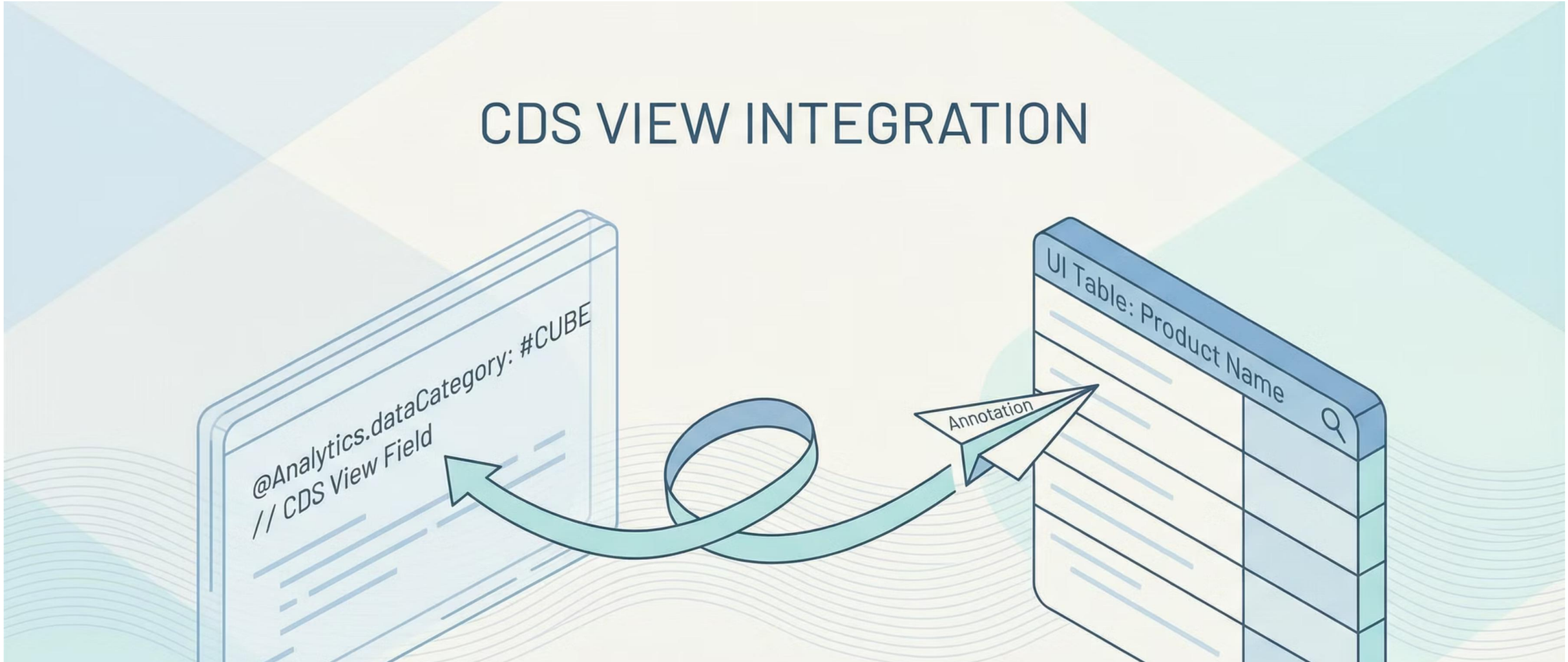
🏷️ @OData:

How should this view be exposed as an API?

🏷️ @AccessControl:

Who is allowed to see this data?

The data model itself carries instructions for how it should be used.



Driving Fiori Development

CDS Views are the primary data source for modern Fiori apps.

01

CDS View:

Defines the data and business logic.

02

OData Service:

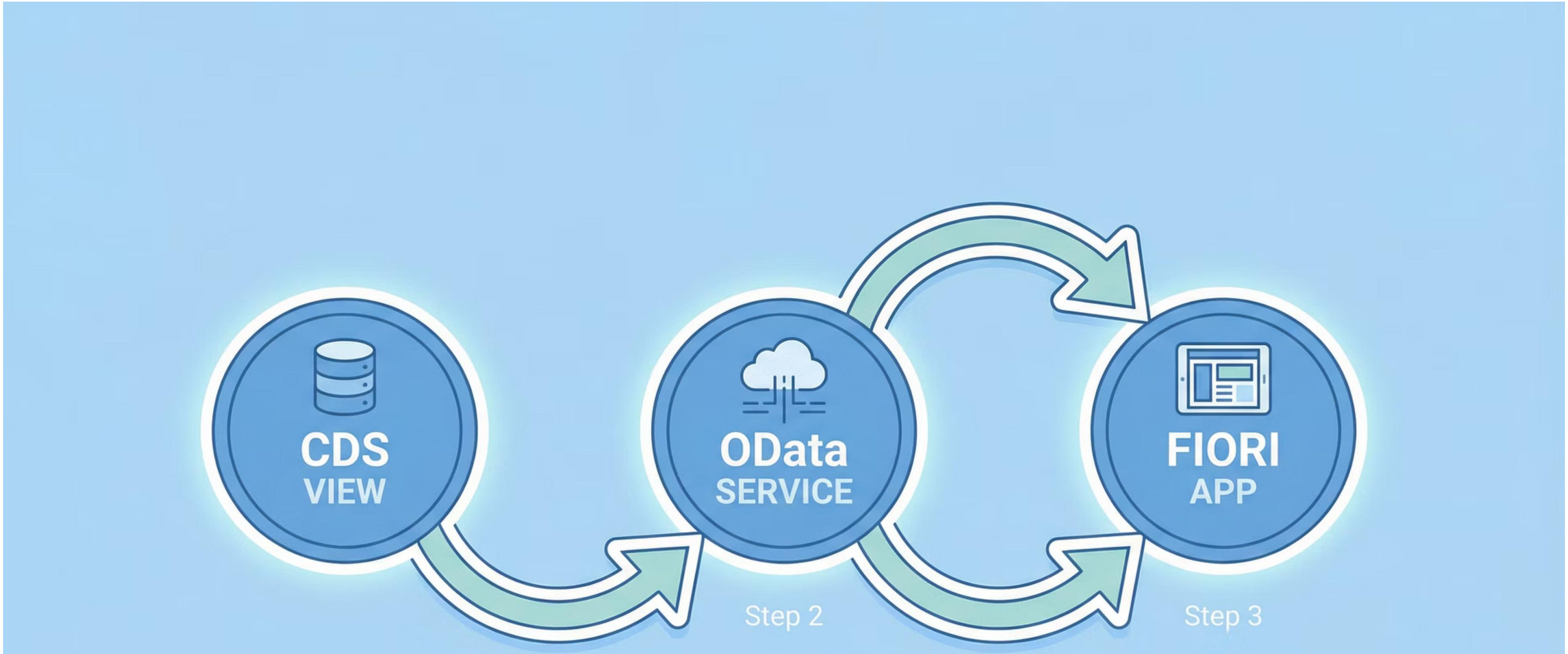
Exposes the CDS View as a standard API.

03

Fiori App:

Consumes the OData service to display data.

With Fiori Elements, the framework can build the entire UI automatically from the CDS annotations.



Designed for Performance

Leveraging the full power of the HANA database.

In-Memory:

Calculations happen at memory speed.

Columnar Store:

Only reads the columns needed for the query.

Parallel Execution:

Splits complex queries across multiple cores.

Processing happens where the data lives, minimizing data transfer.

Layered, Reusable Models

Build once, reuse everywhere.

Layer 3 (Consumption):

Tailor data for a specific app or API.

Layer 2 (Composite):

Add reusable business logic.

Layer 1 (Basic):

Clean access to physical tables.

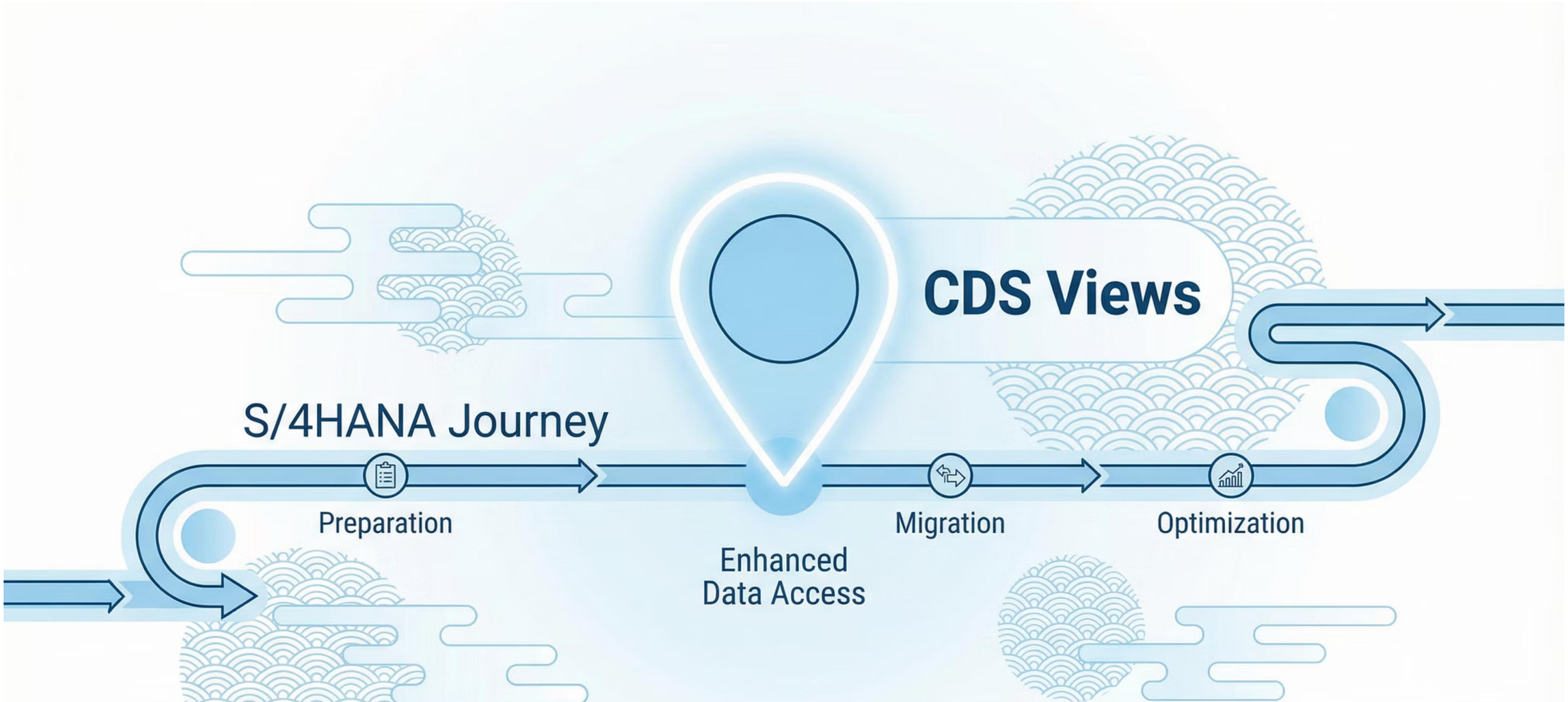
This separation of concerns creates a clean, consistent, and maintainable architecture.

The Strategic Direction

CDS is the future of data modeling in the SAP ecosystem.

- All new S/4HANA apps are built on CDS Views.
- It's the standard for Embedded Analytics.
- It's essential knowledge for all modern SAP developers.

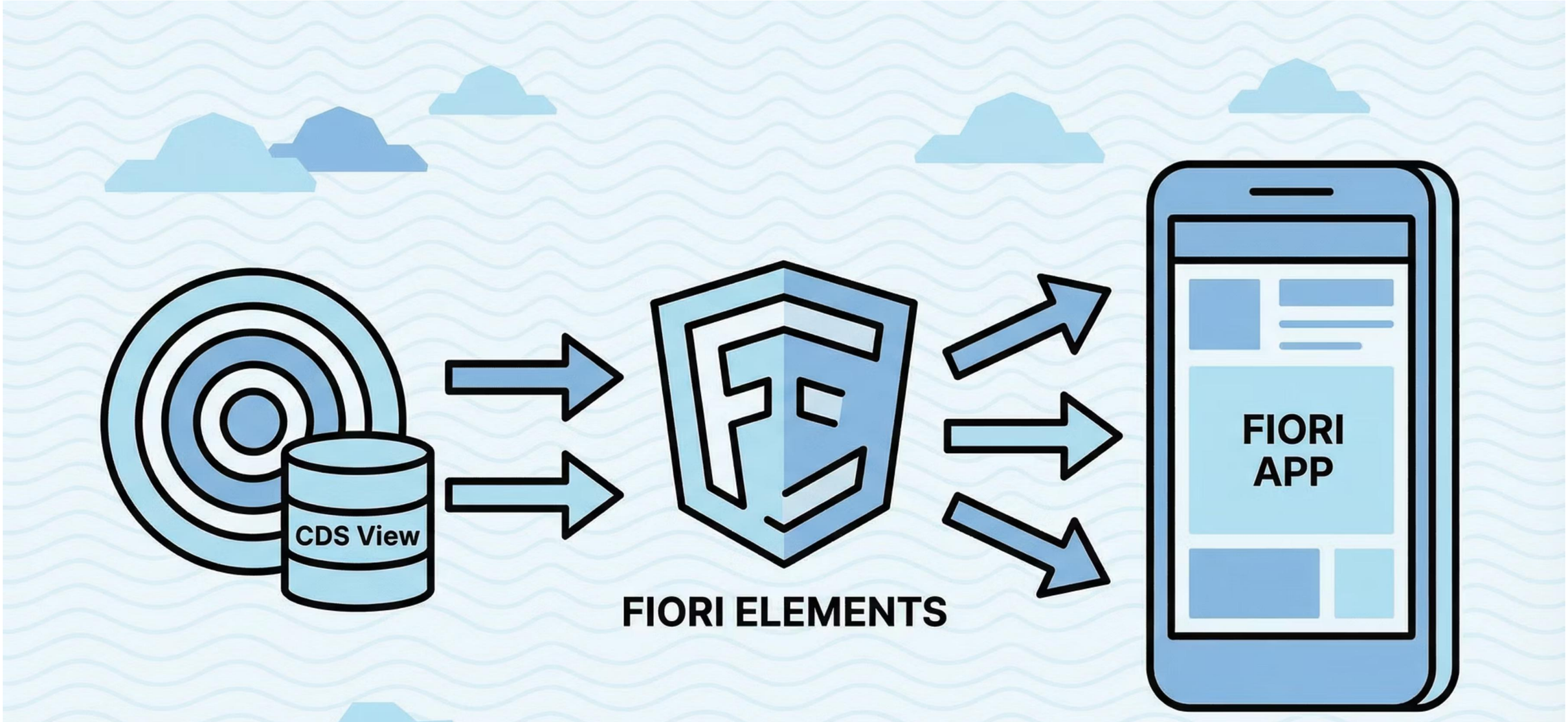
Understanding CDS is understanding the future of SAP development.



What's Next?

From automated data models to automated user interfaces.

How can we use CDS View annotations to generate entire applications with almost no code?



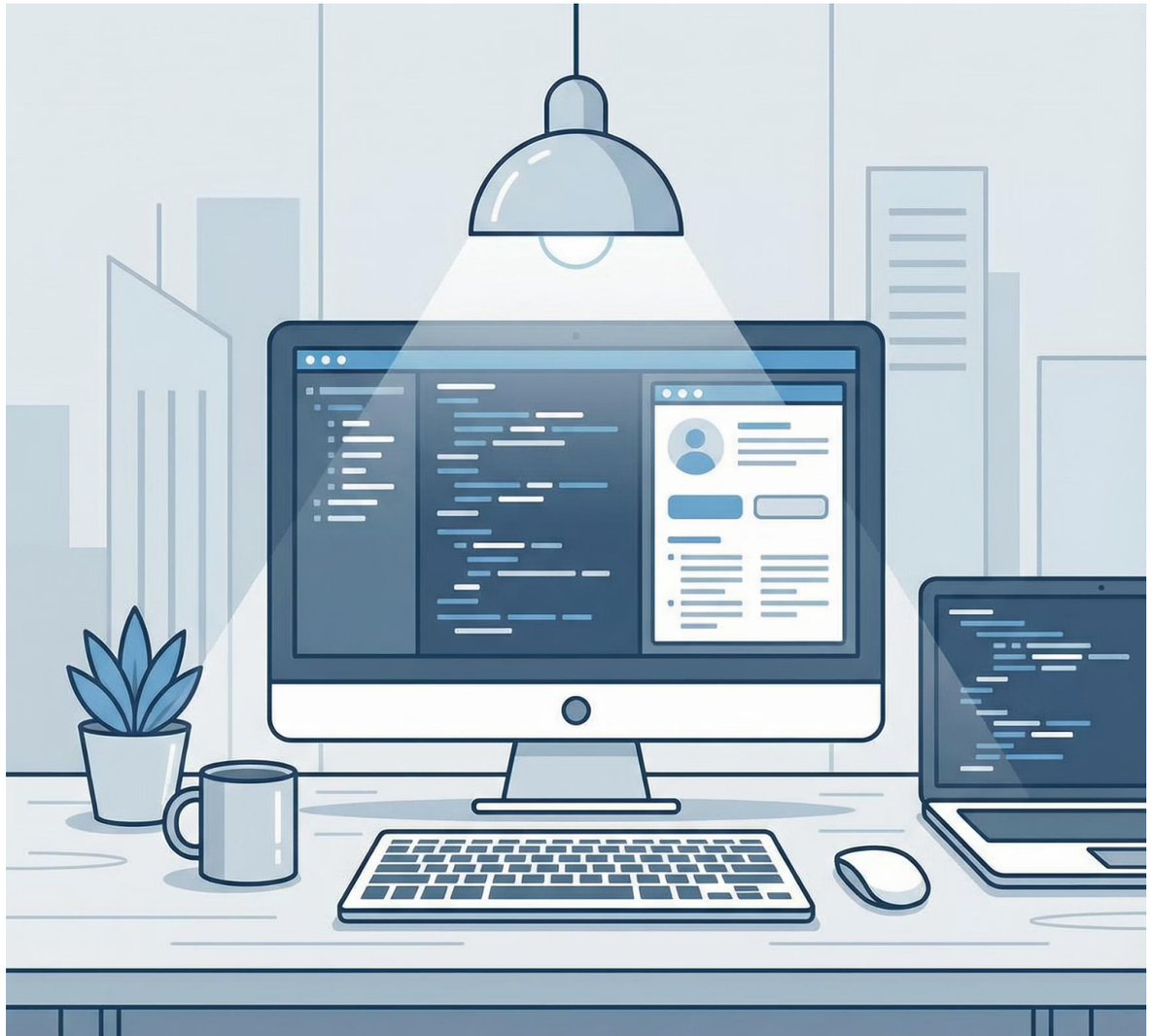
SAP Fiori Elements

Build Apps Without Coding UIs?

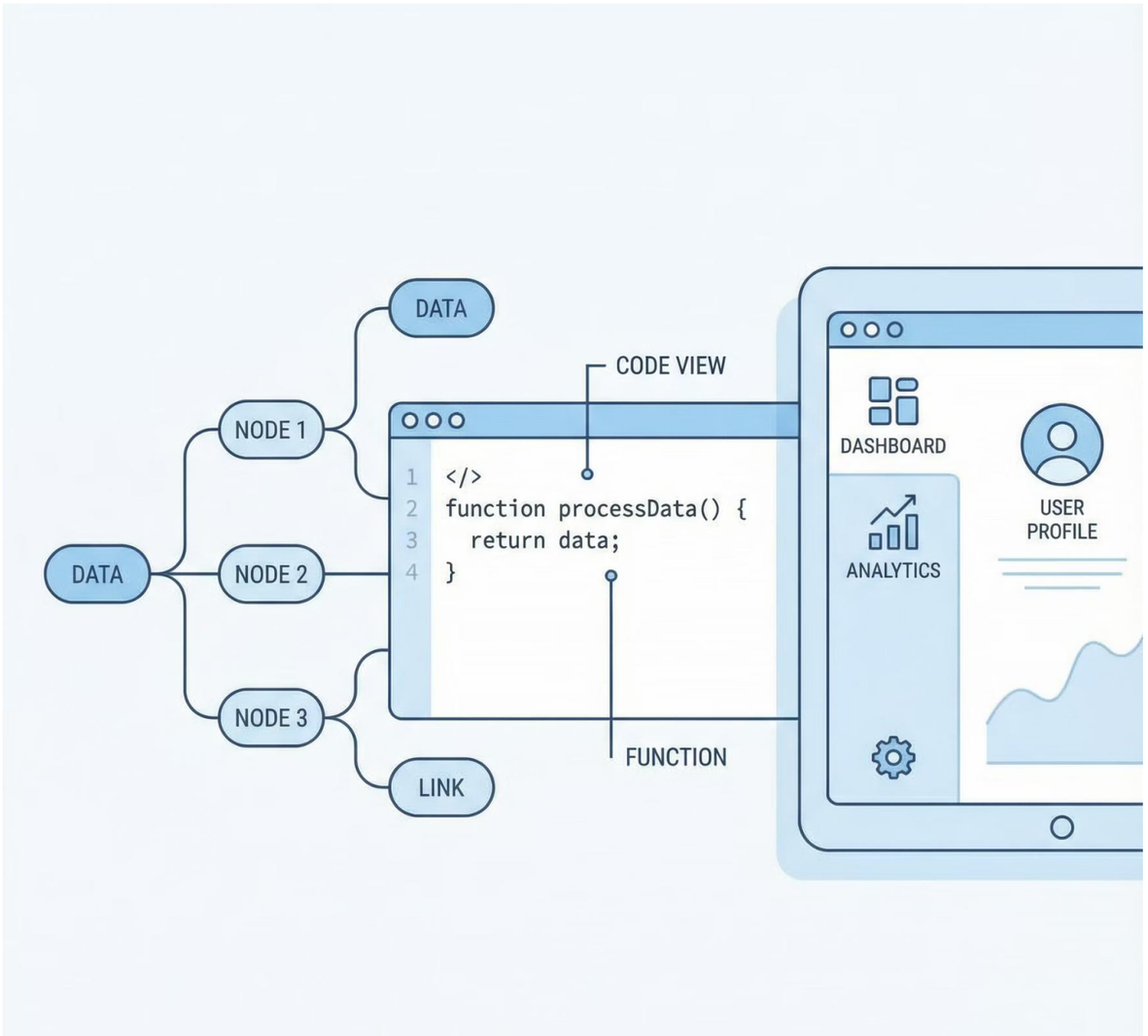
What if you could build a Fiori app by simply *describing* it?

This is metadata-driven development, the core idea behind Fiori Elements.

Before



After



The Fiori Elements Framework

You define the "what," and the framework builds the "how."

1

You Provide:

A CDS View with annotations.

2

Framework Does:

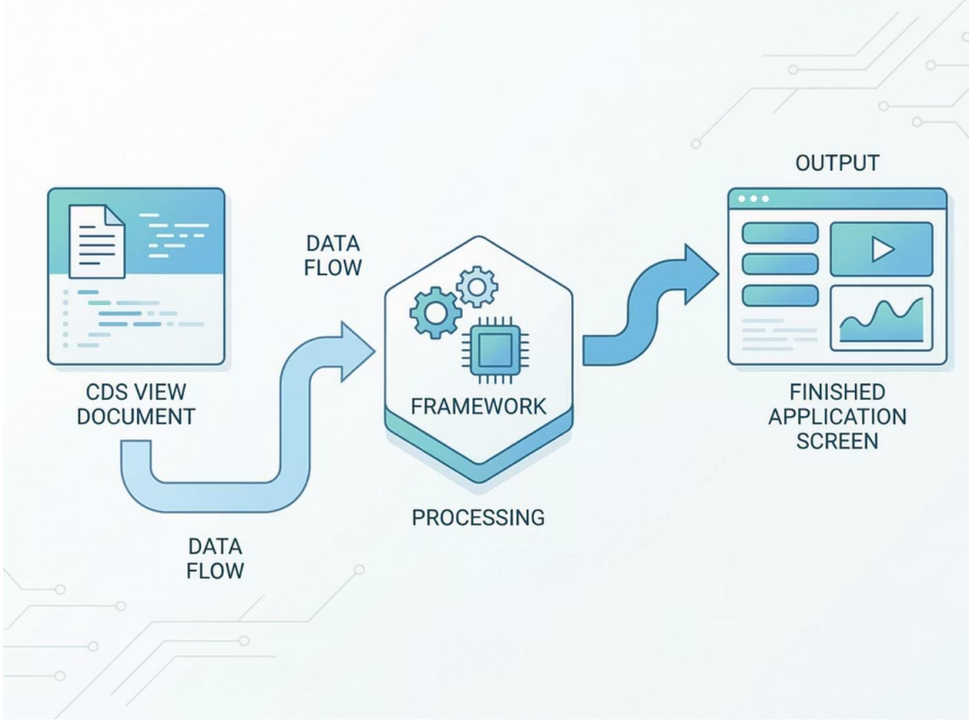
Interprets the metadata.

3

Result:

A complete, production-ready Fiori application.

Your code becomes the blueprint, not the building material.



Annotations as Instructions

The @UI annotations are the commands for the framework.

@UI.lineItem

"Put this field in a table column."

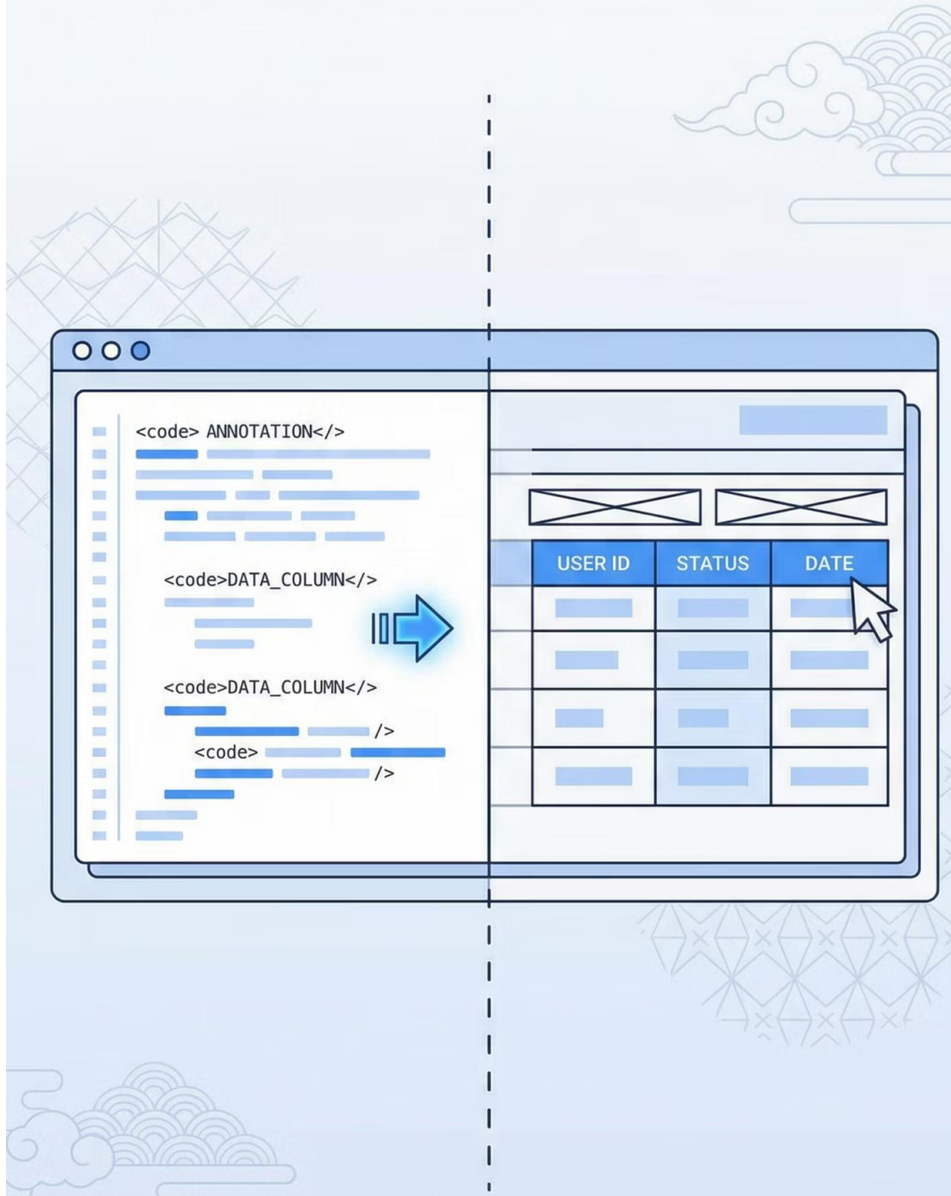
@UI.selectionField

"Make this a filter on the screen."

@UI.identification

"Show this field in the page header."

You're not writing code; you're providing structured directions.



Standard Floorplan Templates

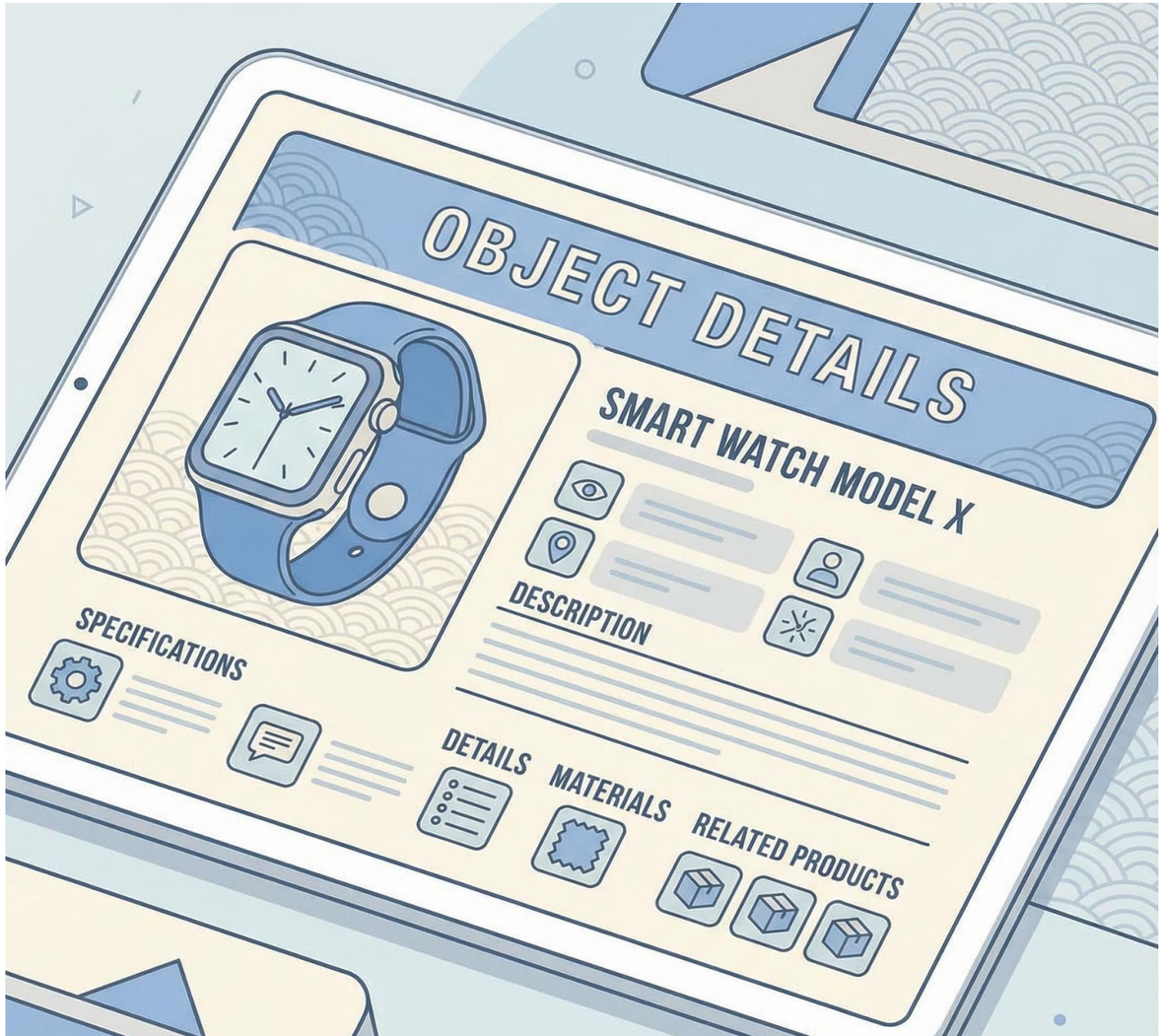
Consistent layouts for common business scenarios.

List Report

A starting page to search, filter, and view a list of items.

Object Page

A detail page showing all information for a single item.



What You Get

The framework automatically builds essential, complex features.

Navigation

Flow between pages is handled automatically.

Search/Filter

Powerful, responsive filtering is built-in.

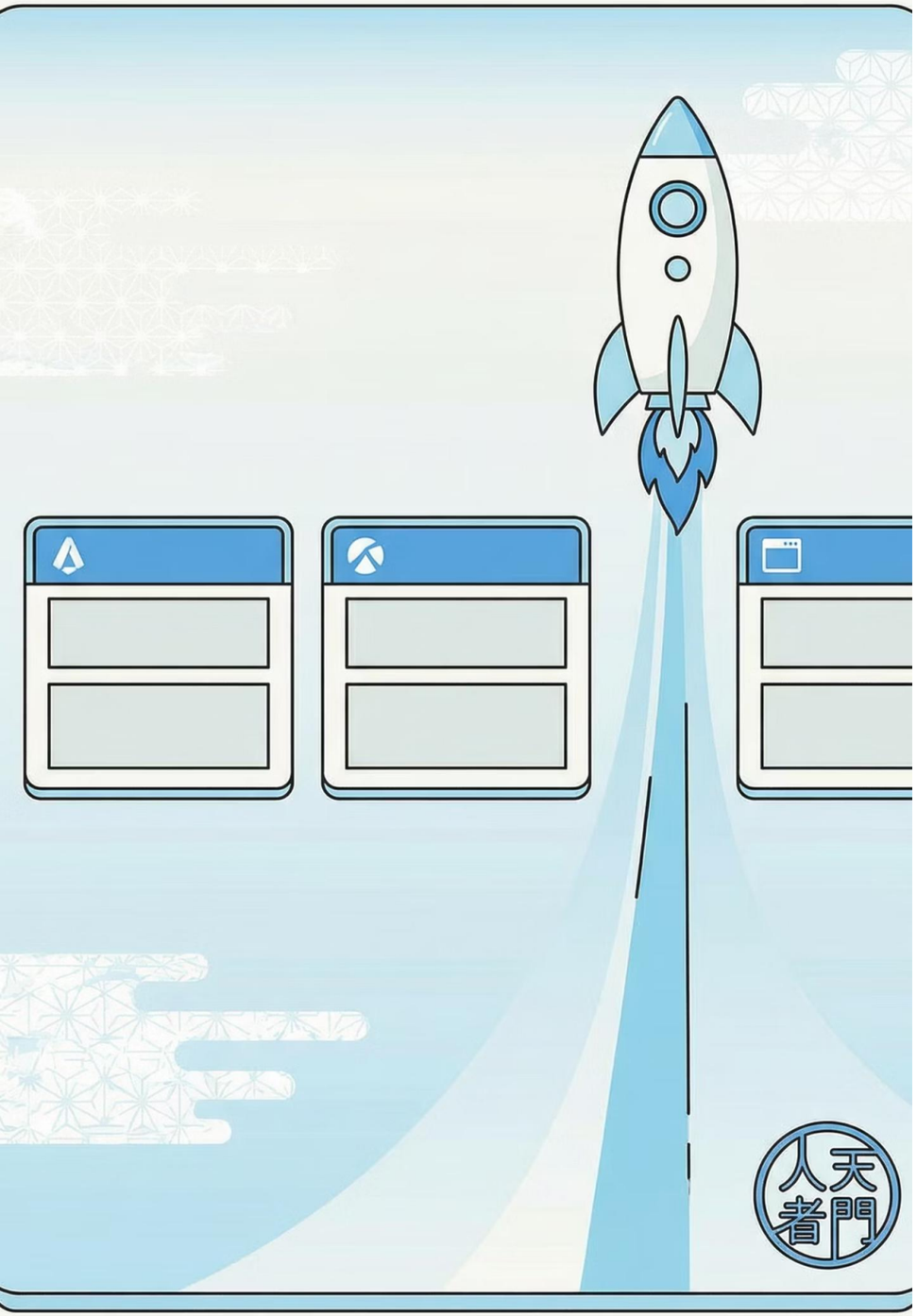
Responsive Design

Works on desktop, tablet, and mobile.

Accessibility

Follows SAP's accessibility standards out of the box.

This saves weeks of custom development effort.



Why Organizations Use It

Speed and consistency are the game-changers.

🚀 Development Speed

- Build apps in days, not months.
- Focus on the data model, not UI code.

✨ Standardization

- Consistent Fiori look and feel across all apps.
- Easier maintenance and upgrades.

Build better, more consistent apps, faster.

When to Use It (and When Not To)

Choose the right tool for the job.

✅ Fiori Elements

- Standard business apps (e.g., List/Object).
- When speed and consistency are priorities.

❌ Freestyle UI5

- Highly unique user experiences.
- Complex UIs that don't fit a pattern.

You can even mix them: build 80% with Fiori Elements, and add a 20% custom freestyle section.

SAP's Strategic Direction

Fiori Elements is SAP's move toward low-code development.

It's the recommended approach for standard S/4HANA apps.

It receives continuous investment and new features from SAP.

Understanding Fiori Elements is key to building modern SAP applications.

The Full Picture

You now understand the complete modern SAP development stack.

1

HANA

(The Engine)

2

CDS Views

(The Blueprint)

3

Fiori Elements

(The Factory)

4

Fiori App

(The Final Product)

This architecture enables rapid, consistent, and powerful application development.