
Understanding the Real Design Intent of AMDP and CDS Views

1. Introduction

In the world of ABAP on HANA, two major technologies define how developers interact with the database layer: **Core Data Services (CDS)** and **ABAP Managed Database Procedures (AMDP)**.

While both are designed to leverage HANA's in-memory power, they serve very different purposes.

CDS is about **data modelling and semantics**, whereas AMDP is about **complex computations and logic execution**.

At first glance, both seem capable of data retrieval and processing, but their **architectural intent** differs entirely.

This document explains why SAP separated the two, what it means when we say "*AMDP is deterministic and database-optimized*", and how both fit into the broader S/4HANA landscape.

2. Why SAP Introduced CDS and AMDP Separately

Before HANA, ABAP developers primarily relied on Open SQL and internal processing loops. Data was fetched from the database into the application server, where business logic was executed. This was fine for small datasets, but inefficient for large-scale or analytical operations.

With HANA's in-memory, column-oriented database, SAP reimaged how data should be processed — **push logic down to the database** instead of pulling data up to ABAP.

However, SAP recognized two fundamentally different needs:

Layer	Purpose	Expected Output	End Consumer
CDS Views	Define, model, and expose data semantically	Structured dataset with business meaning	OData, RAP, Fiori, BW, analytics
AMDP	Perform complex logic, transformations, or calculations directly in HANA	Computed result sets, not reusable semantics	ABAP reports, background jobs, frameworks

This separation ensures a **clean boundary**:

- CDS defines **what** data represents.
 - AMDP defines **how** data is processed when complexity exceeds CDS capabilities.
-

3. Deterministic and Database-Optimized — What It Really Means

When SAP says AMDP is designed for “**deterministic, database-optimized operations**,” it is referring to **predictability and optimization**.

Deterministic Behavior

Deterministic means the database can **predict the query's structure and logic at design or compile time**.

In other words, the query and its data flow are fixed — the database knows:

- Which tables will be read
- Which joins or filters will be applied
- What columns are needed
- How data will be aggregated

Because everything is known in advance, the **HANA optimizer** can:

- Pre-generate the most efficient execution plan
- Cache that plan for repeated execution
- Minimize runtime overhead

This is why deterministic AMDPs perform exceptionally well. The database knows the path it needs to follow — just like a pre-planned route that never changes.

Database-Optimized Operations

When the logic is deterministic, HANA can fully utilize its in-memory, columnar structure:

- Only required columns are read into memory.
- Parallel processing can be planned in advance.
- Intermediate results stay within memory — no disk I/O.
- Execution plans are reused efficiently.

The outcome is predictable performance and highly efficient data access — the core promise of SAP HANA.

4. Why Dynamic or Generic AMDPs Are Discouraged

Sometimes developers try to make AMDPs dynamic — building queries at runtime with variables like table names or field lists. This flexibility is tempting, but it removes the deterministic nature.

Dynamic AMDP Example

```
lv_sql = 'SELECT * FROM ' || :iv_table;
EXECUTE IMMEDIATE :lv_sql INTO rt_result;
```

Here, the database no longer knows:

- Which table will be used
- What columns exist
- What indexes or joins to apply

Each execution may be different, so HANA must **re-parse and re-optimize the query** every time.

This breaks optimization caching and adds overhead. In short, it becomes **non-deterministic** — the database cannot pre-plan the work.

Furthermore:

- Static code checks cannot validate syntax.
- The result structure is not predictable.
- There are potential SQL injection risks if parameters are not sanitized.
- CDS or OData exposure cannot consume such dynamic outputs, since metadata is unknown at design time.

That is why SAP restricts dynamic SQL in AMDP to framework-level components only (for example, data extraction utilities or generic loaders), not for standard business logic.

5. CDS Views — Static and Semantic by Design

CDS views were created to model data relationships and expose them consistently across systems.

They must always remain **deterministic and metadata-driven**.

In a CDS, everything is **known at activation time**:

- Tables and joins are fixed.
- Field names are predefined.
- Associations, annotations, and data types are clear.
- OData metadata can be generated directly.

This deterministic behavior allows:

- Seamless OData exposure.
- Compatibility with RAP, Fiori, and UI5.

- Simplified authorization and data governance.
- Predictable performance with HANA optimization.

In short, CDS views represent **semantic stability** — they form the foundation for reusable, readable, and cacheable data models in the SAP ecosystem.

6. AMDP — When CDS Is Not Enough

CDS covers 80–90% of business modeling scenarios. But when logic becomes more computationally heavy or requires procedural handling, AMDP takes over.

For example:

- Recursive calculations (like BOM explosion)
- Complex aggregations not expressible in SQL
- Statistical functions or simulations
- Multi-step transformations
- Dynamic runtime filters controlled by user input

AMDP runs SQLScript directly on HANA, allowing such advanced logic while still remaining database-optimized.

However, SAP expects these operations to stay **deterministic within their scope**, meaning the structure and logic are defined, even if the parameters change.

7. Why AMDP and CDS Work Best Together

In a typical S/4HANA or RAP design:

- CDS views handle data modeling and exposure (to OData or UI).
- AMDP handles performance-intensive calculations or transformations.
- CDS views can consume AMDP table functions when specialized logic is required.

This ensures:

- Data models remain reusable and analyzable.
- Performance bottlenecks are offloaded to AMDP without breaking semantics.
- The system retains predictable optimization and consistent metadata exposure.

SAP's separation of CDS and AMDP is therefore deliberate: it enforces **clear layering** and protects system integrity.

8. Real-World Analogy

Think of CDS and AMDP as two different transportation systems.

Analogy	Behavior	Optimization
CDS View	A train route	Fixed path, known schedule, pre-optimized, consistent
AMDP	A high-speed freight truck	Handles special loads, heavy tasks, optimized for speed but within defined routes
Dynamic AMDP	A taxi with no fixed route	Very flexible but hard to plan or optimize efficiently

SAP's intent is that business data flow (CDS) stays on the train — predictable and well-mapped — while heavy backend processing (AMDP) uses the freight truck. The free taxi approach is used only when absolutely necessary.

9. Summary Table

Aspect	CDS View	AMDP
Purpose	Data modeling and semantic exposure	Heavy computation and logic pushdown
Type	Declarative	Procedural
Behavior	Deterministic	Deterministic by design (dynamic only in rare use cases)
Optimized for	OData, RAP, analytics	Performance-intensive calculations
Output structure	Fixed, known at design time	Fixed unless explicitly made dynamic
Exposure	Can be published as OData	Not directly; used internally or via CDS Table Function
SAP's design goal	Semantic clarity and reusability	Database-level efficiency and precision

10. Conclusion

SAP designed **CDS** and **AMDP** to complement each other, not to replace one another. Both serve different architectural purposes, unified by one principle — **code pushdown**.

When SAP says AMDP is “*deterministic and database-optimized*,” it means AMDP should:

- Execute fixed, predictable logic.
- Allow HANA to optimize the query path in advance.
- Produce consistent, reusable performance outcomes.

By contrast, **dynamic or generic frameworks** belong only in the lowest technical layers — not in the core business or semantic models — because they compromise determinism and optimization.

In essence:

CDS defines what the business wants to see.

AMDP defines how complex data should be processed efficiently.

Together, they form the foundation of high-performance, maintainable, and future-ready ABAP on HANA development.
