In-Memory Computing SUMMIT EUROPE 2019

Does the data model fit to in-memory computing? Tales from SAP Analytical Banking software.

Dirk Vollmer, Solution Architect
ADWEKO Consulting GmbH
4th June 2019

ADWEKO

Agenda

- Introduction
- Classical SAP Analytical Banking on any DB
- Bringing SAP HANA to the game: Finance and Risk Data Platform
- Next generation in-memory SAP Financial Analytics
- Conclusion

Who we are

Our Services

Consulting

Finance
—
Regulatory Reporting
—
Risk Management
—
Data Management

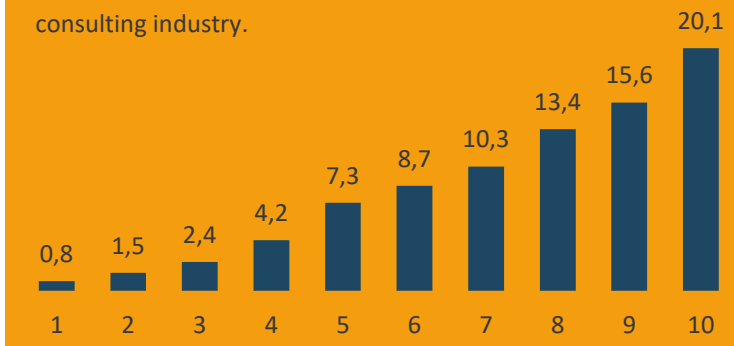
Managed Services

Application Management Services
—
Managed Services

Software Solutions

Smart Add-ons
—
Individual Contract Development
—
Configuration & Enhancements

Since its foundation, ADWEKO has become a renowned company in the financial services consulting industry.



Sales Trend in Mio. €

SAP HANA Reference

Project at a major German bank

Conceptual and operative implementation of SAP FSDP

SAP Silver Partner

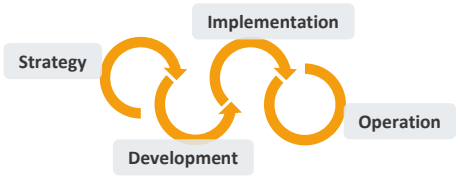
SAP FSDP Expert Partner

Release Upgrade

Release upgrade of financial service solution to SAP HANA

Expert Solutions

with a wide range of services for all stages of an IT project's life cycle and partnerships with the most important IT solutions providers for banks and insurance companies.



BearingPoint
pitney bowes
Qlik
SAP Silver Partner
Wolters Kluwer

Specialisation

with solutions to current challenges & trends and our unique attribute – the ADWEKO team.

Factory Model
HANA Best Practices
Kafka-to-SAP
SEM-PA to FPSL
Test Automation

Technical Experts
Experienced Industry Insiders
Pleasant Team Players

29%
Sales growth

150
Employees are part of the ADWEKO family in 2019

45
Customers in the Financial Services field

>330
Projects since ADWEKO's foundation in 2008

Classical SAP Analytical Banking

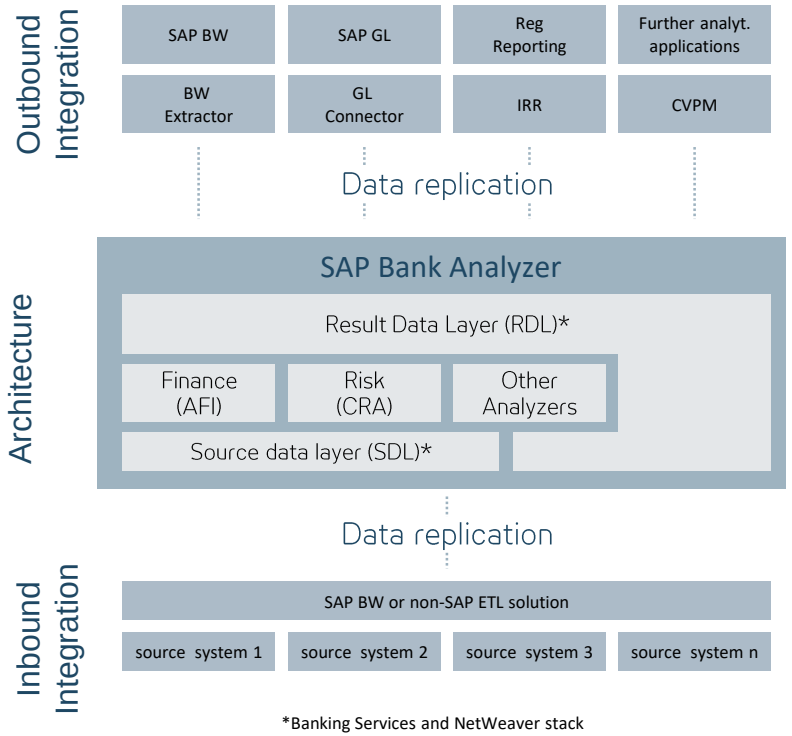
SAP Bank Analyzer on any DB

First generation SAP Analytical Banking Software

SAP Analytical Banking Software

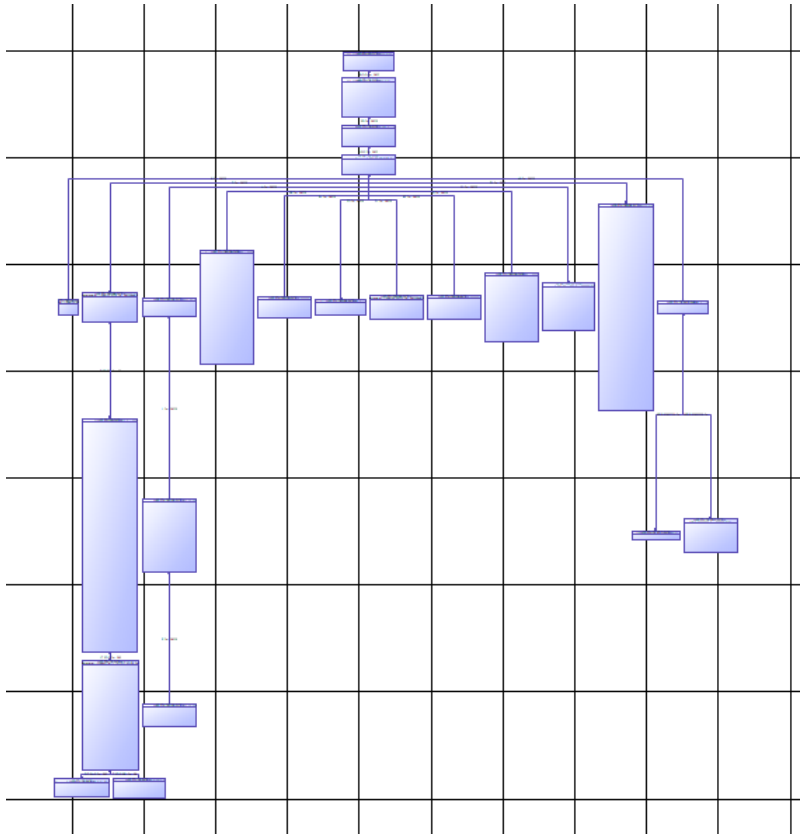
- Save Data: Source Data Layer
- Process Data: Process and Methods Layer
- Save Data: Result Data Layer

- Bi-temporal versioning
- NetWeaver stack and any DB concept
- Relies heavily on ABAP application logic



The traditional Bank Analyzer Data model

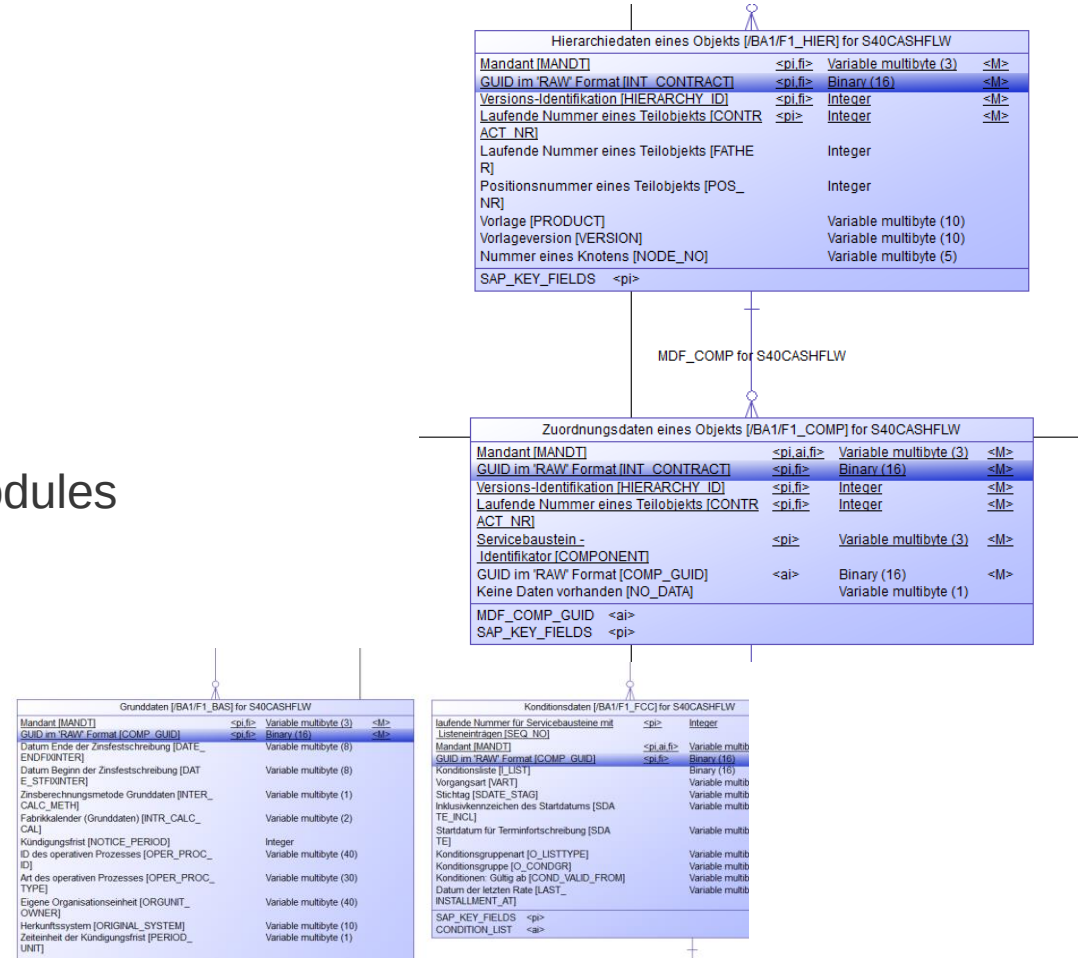
Master Data Framework – Financial Transactions



The traditional Bank Analyzer Data model

Traditional data model

- /BA1/F1_CONTRACT: Header
- /BA1/F1_VERSION; Version
- /BA1/F1_HIER(_N): Hierarchy
- /BA1/F1_COMP: Link to Service Modules
- /BA1/F1_xyz: Service Module data



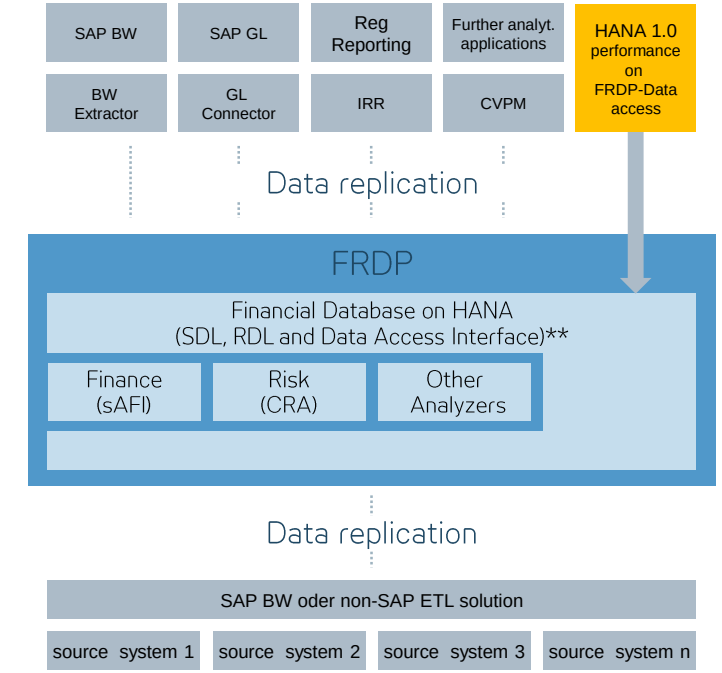
Finance and Risk Data Platform

Analytical Banking on HANA

Bringing SAP HANA to the game

SAP Finance and Risk Data Platform

- HANA views to access data on database level rather through application stack
- Same data model for legacy applications
- No major adjustments in application logic, however, new process smartAFI only available on HANA



**Banking Services, NetWeaver and HANA stack

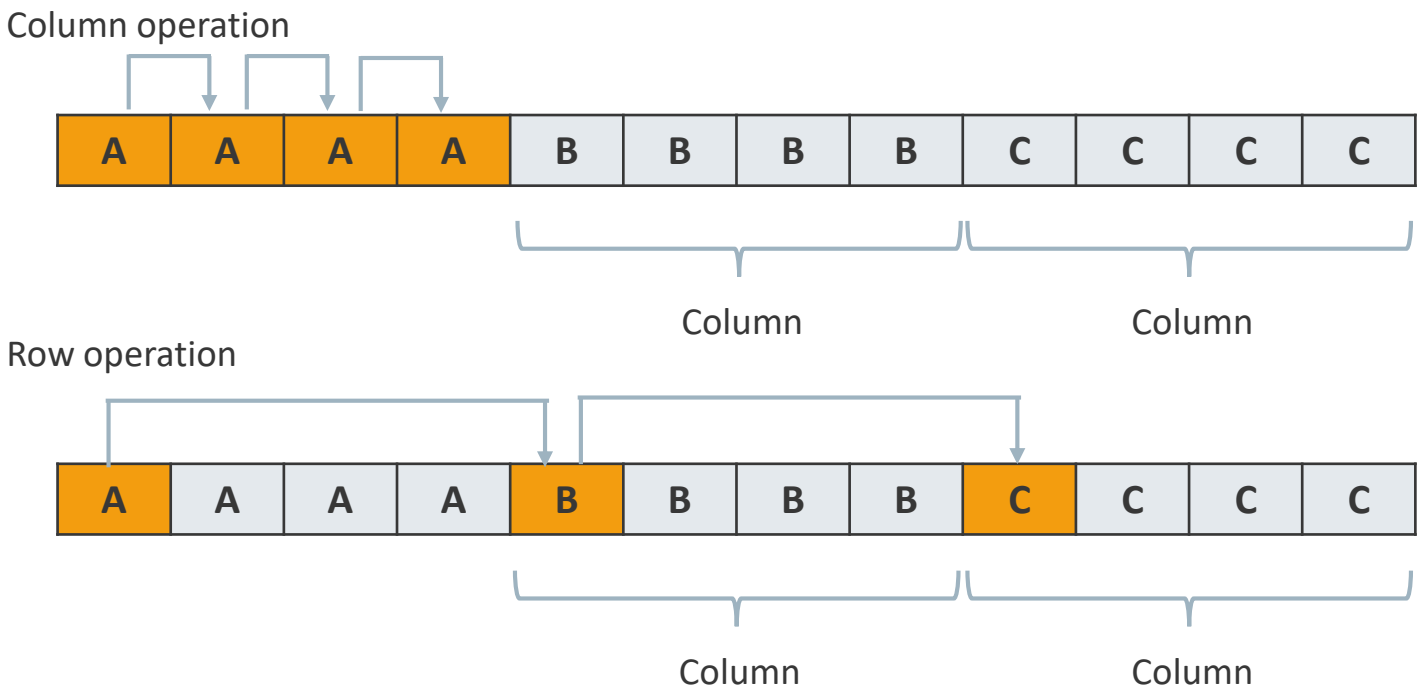
Access pattern of data

Reporting

- Aggregate Results per Unit, Region, Profit Center, …

Analyzers

- Select single contract to evaluate
- Join with Business partner or market data information



Bank Analyzer on HANA 1.x

- Legacy Applications can show bad performance out of the box after HANA migration (For example 4h -> 90h)
- Bank Analyzer applications and process frameworks are single object based and causing a lot of Single Selects on the Database
- HANA DB shows bad performance on Single Select (for example 1000 DB calls for 1 data object instead of 1 DB Select for 1000 datasets)
- Due to historical reasons no “Code to Data” Paradigm – a lot of ABAP application logic

Loop over Single Contracts and do calculations and joins to other objects

Bank Analyzer on HANA 1.x

Optimization process

Bank Analyzer on HANA 1.x

What can be done?

- Optimization of application server and HANA database server settings
- Optimization of HANA database on a per table basis (indices, partitioning, …)
- Addition of DB hints
- Use of buffering mechanisms to avoid single selects
- Adjust application server process parameters (parallelization, package size) to optimally utilize HANA resources

Bank Analyzer on HANA 1.x

Database tweaks

- SAP HANA very young compared to classical Databases
  - (Was) not mature and stable in some areas
  - Service packs and patches can have significant run time effects
  - New features in every release especially with HANA 2.0

- Tweak the database parameters
  - Example: Fast Data Access (FDA) parameters

Fast Data Access

What is fast data access?

- Fast data access (FDA) is a protocol available in SAP ABAP environments. It allows to submit open SQL SELECTs against SAP HANA using the data format of SAP ABAP, i.e. a SAP ABAP table is transferred to the database and back. This approach eliminates the need for field-wise copying and data conversion.

What are the advantages of fast data access?

- The elimination of conversion overhead can result in reduced resource utilization and improved performance.

Which problems exist in relation to fast data access?

- At least 15

Parameter

- rsdb/prefer_join_with_fda = 0 (deactivated) or = 1 (activated) ??
- &prefer_join_with_fda <val>&: Hint to control for specific SQL statements

Buffering framework in SAP Bank Analyzer

Buffering

- SAP Bank Analyzer provides buffering mechanism in application stack
  - Before DB is accessed buffer is checked. Only if data is not found in buffer, call DB
  - Bulk Select all objects in working package before processing
  - Standard Buffering framework for some SAP utilities (if used correctly) – but not necessarily for customer implementations

- Implement buffering framework

Buffering framework in SAP Bank Analyzer

Calling standard SAP buffering methods in customer class with all external object keys

```
buffering per position class
CALL FUNCTION '/BA1/F2_API_POS_PREP_BUFFER_2'
  EXPORTING
    i_tab_all_def_chara = l_tab_all_def_chara
    i_keydate           = m_business_day
    i_timestamp         = l_var_timestamp
    i_tab_state         = l_tab_pos_state
    I_SCENARIO_ID       =
    I_VERS_CAT          = 'R'
    I_FLG_EXTCON_ONLY   = ABAP_FALSE
  IMPORTING
    E_CNT_BUFFER        =
    e_tab_return        = l_tab_bapiret2
  EXCEPTIONS
    failed              = 1
    OTHERS              = 2.
```

```
CALL FUNCTION '/BA1/F1_API_CF_READ_CON'
  EXPORTING
    i_tab_contract_sel  = m_tab_contract_con_sel
    i_tab_component_sel = l_tab_component_sel
  IMPORTING
    e_tab_contract      = l_tab_contract.
```

```
"fill global buffer
CALL FUNCTION '/BA1/F1_OBJ_FRAMEWORK_FILL_DB'
  EXPORTING
    i_tab_contract_sel  = l_tab_contract_sel
    i_status_sel        = i_status_sel
    i_rng_component_sel = l_rng_component_sel
    i_skip              = i_skip "rmr1746933
  EXCEPTIONS
    failed              = 1.
```

Partitioning

Hard cut: ~2 Billion rows

General rules:

- Partition tables > 100 Million entries
- Partitions should be not bigger than 400 Million entries
- Not using “OTHERS” as residual partition
- Mediocre performance if partition key not in ‘most’ WHERE conditions as EQ
- Number of partitions should be multiple of 2

Partitioning and Scale-up/out

Typ	Details	Benefits	Disadvantages
HASH	Partitioning key based on HASH algorithm on the primary key.	Simple setup, no maintenance evenly distributed partition keys likely if primary key selective.	All partitions must be scanned if key fields are not defined in WHERE condition. No logical separation (hot-cold) possible.
ROUND ROBIN	Uniform distribution procedure without primary key.	Simple setup, uniform distribution.	All partitions must be scanned, no "partition pruning." No logical separation (hot-cold) possible.
RANGE	Definition of non-overlapping partition key (e.g. year).	Application-driven separation in "hot" vs. "cold." Best results for "partition pruning" and delta-merge optimization.	Application knowledge (access routes) and maintenance (new periods) necessary. Distribution of data can be uneven.

Flat data structures

SAP approach to optimize HANA capabilities

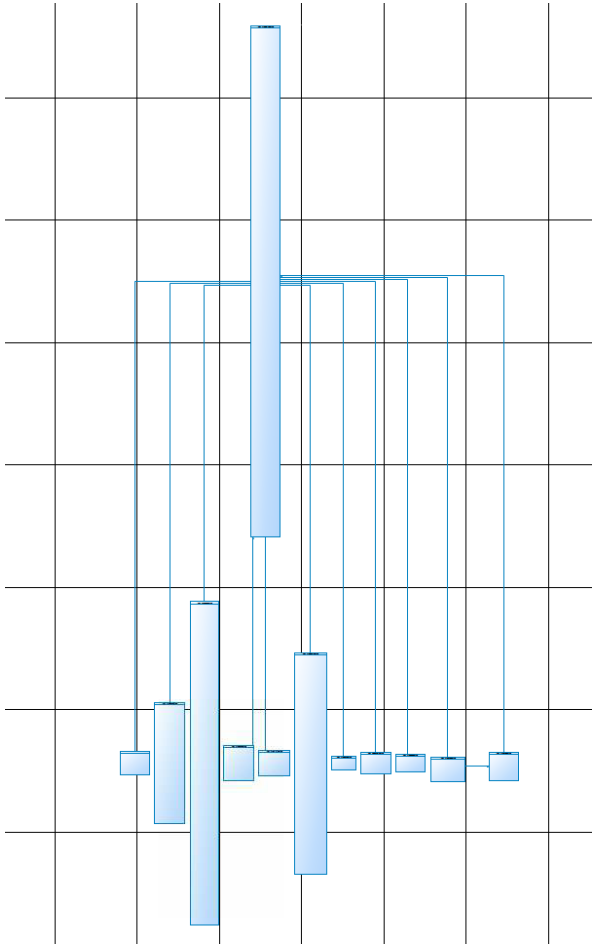
- For new applications (smartAFI) SAP introduced a simplified data model
  - “Flat” source and result data tables
  - Fewer but wider tables to avoid extensive joins and use column store more efficiently
- Partitioning key introduced in data model

- However: Business Partner data model not changed (part of other SAP solutions)

Bank Analyzer with Flat Structures

De-Normalized (Flat) master data for financial contracts

- /BA1/F1_CON_FLAT: Header, Version, Hierarchy, inline Service Modules and Link to not inline Service Modules (one GUID for all Service Modules)
- /BA1/F1_xyz_FLAT: Service Module data of not inline Service Modules (mainly 1:n)
- Fields PARTITION_VALID_TO and SYS_CURRENT: Fast access to most recent version

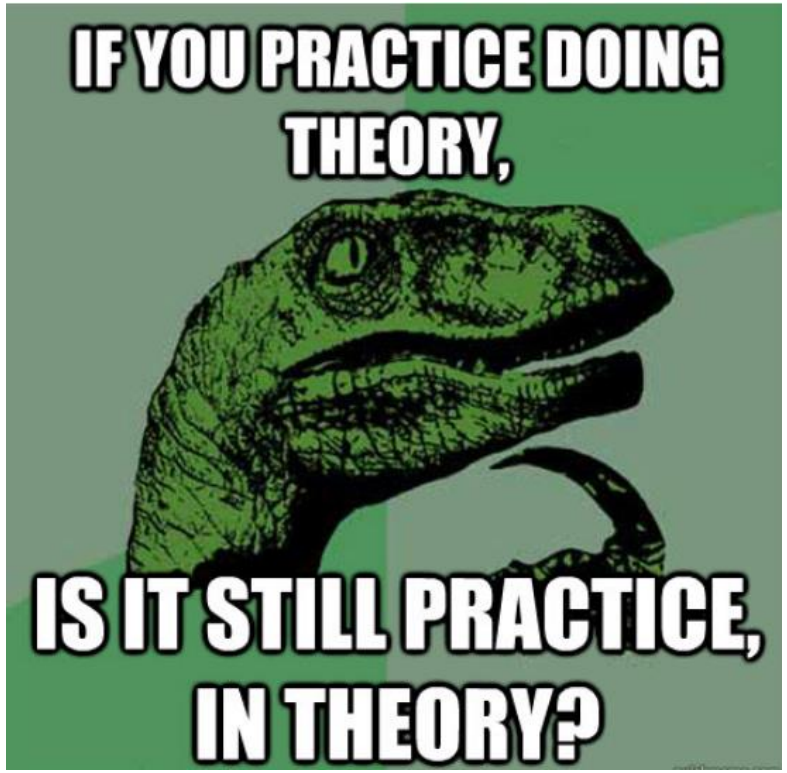


Bank Analyzer with Flat Structures

It works … in theory

- Standard SAP APIs not optimized or adapted for data model
  - SAP Notes (Patches)

- Buffering framework explodes!
  - First standard implementation put all versions and whole tables to memory!
  - Restrict to relevant versions and fields!



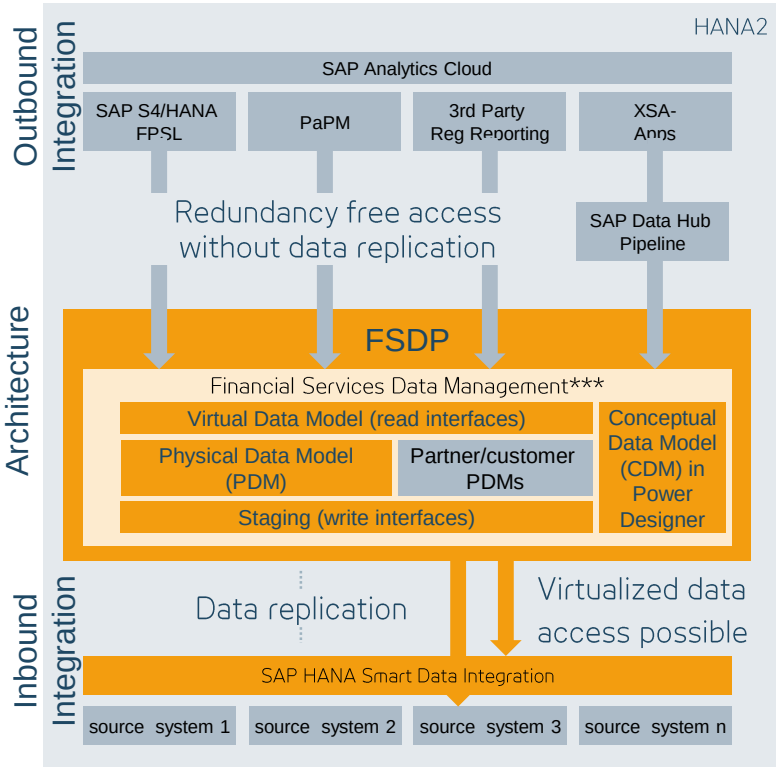
Next generation SAP Financial Analytics

Native Application on HANA2

Next generation data platform

SAP Financial Service Data Platform

- FSDP is a Data Platform for Banking and Insurance, which is implemented as a SQL Datawarehouse on SAP HANA with specific Financial Service content
- Hana 2.0 XSA native – No NetWeaver stack
- Standardized Data model for financial industry
- No “analyzers” – Data model serves as foundation for dedicated SAP and 3rd party software

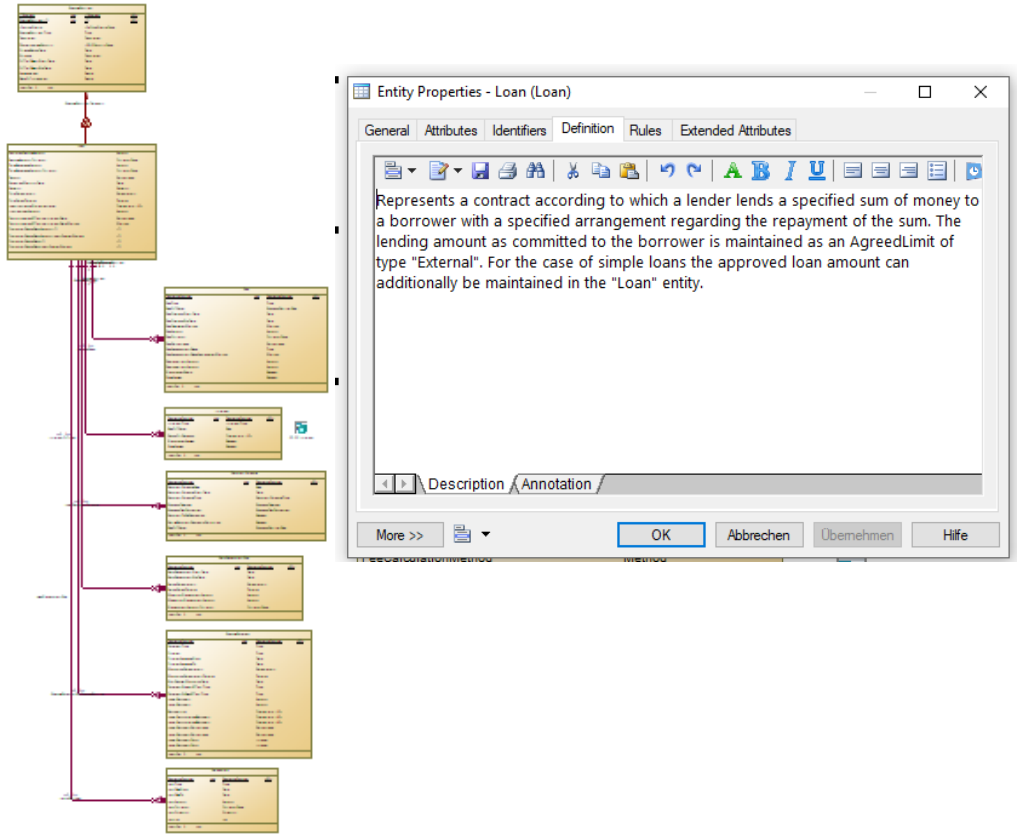


***FSDM, XSA and HANA stack

Financial Service Data Platform

The conceptual data model

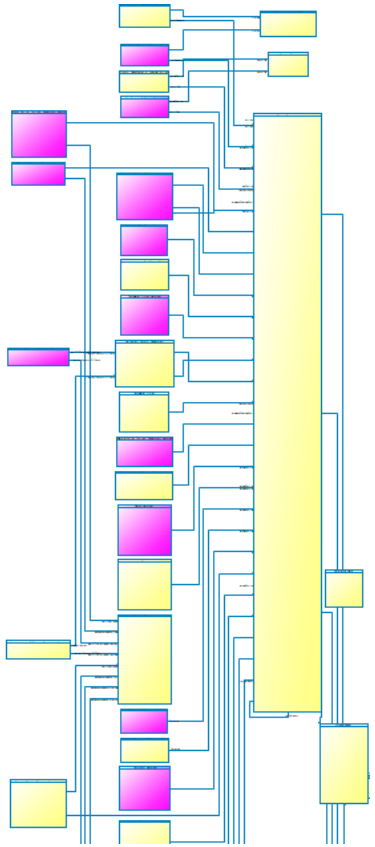
- Provides business semantics of the data model
- Normalized and semantic view on the banking and insurance world
- Helps to understand relations between entities
- For the data mapping and to talk to the “Business User”



Financial Service Data Platform

The physical data model

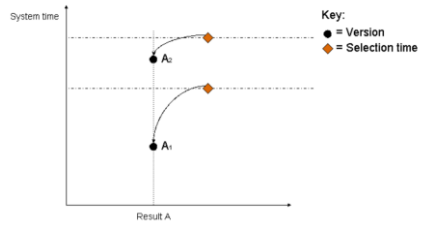
- Keep the tables de-normalization
- History tables and system versioning
- Bulk inserts and reads
- Ready for partitioning (but has to be implemented by customer)
- No “application logic” – Implement on database level with “code to data” paradigm

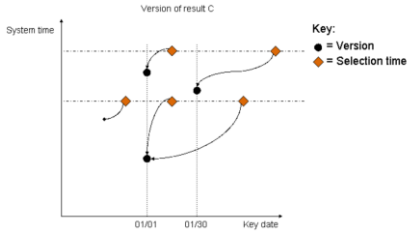


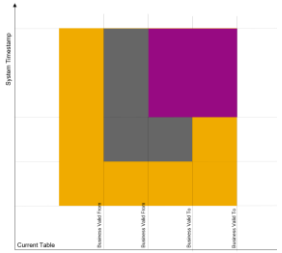
Bi-temporal versioning

Key Date Value can be differentiated into following type:

- Valid-on
  Key Date Valid is valid in one particular date -> combined with system time: One-Dimensional versioning
- Valid-from
  Key Date Value is valid from a particular date onwards until it get invalid by the same Key Date Value at a later validity date -> combined with system time: Two-Dimensional versioning
- Valid-from Valid-to
  Key Date Value is valid from a particular date onwards with a concrete validity end date

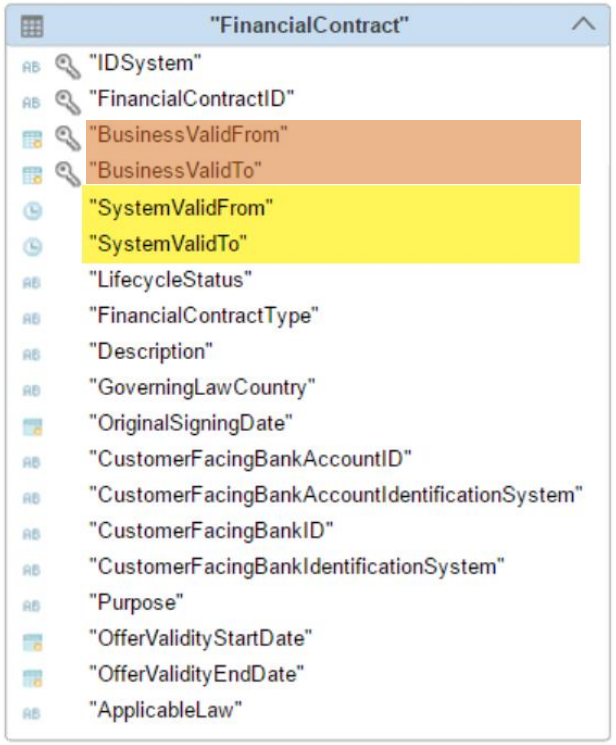






Versioning - Temporal Tables

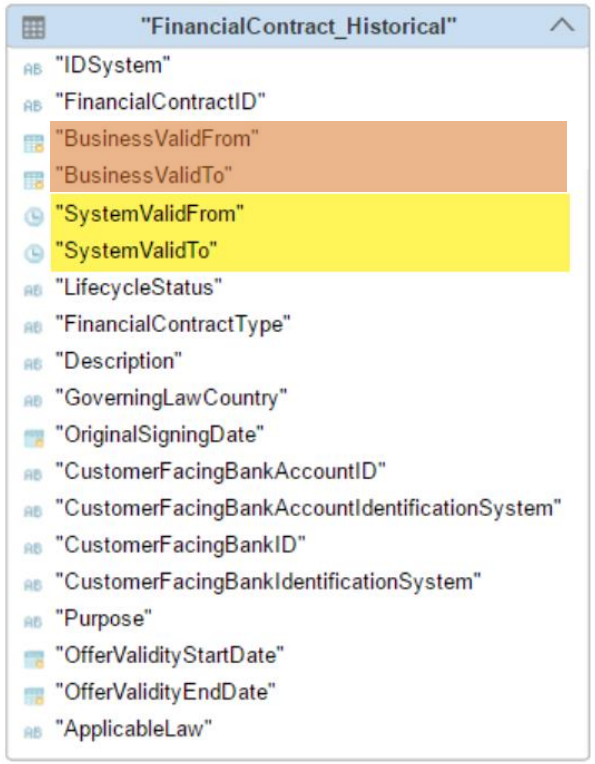
Current table



Business versioning fields

System versioning fields

History table



Write functions

- Generated write interfaces
- HANA SQL procedures
  - Handling business time stamp in SQL logic
  - Handling system timestamps with database features

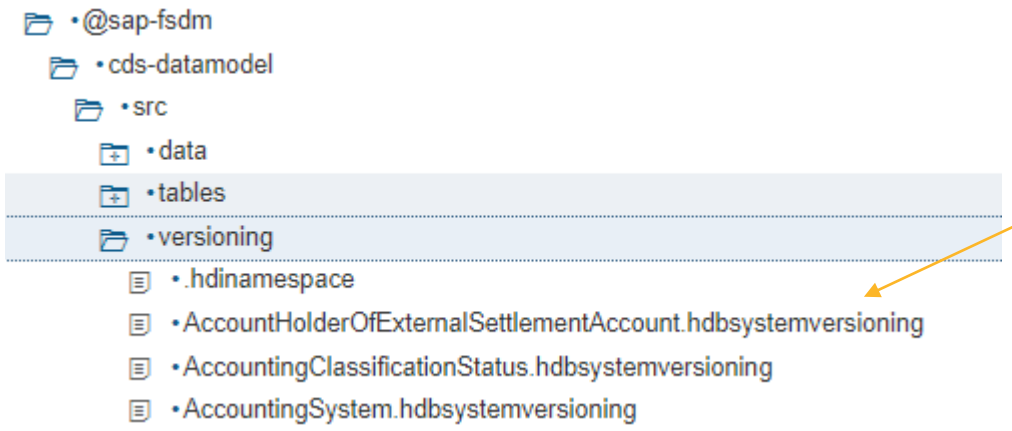
cds × | BusinessPartnerLoad.hdbproc... × | mta.yaml × | AddressDelete.hdbprocedure × | Addre

```
50    --select data matching in current table
51    var_matching =
52    select
53  LAG("IN"."BusinessValidTo",1,"OLD"."BusinessValidFrom") OVER (PARTITION BY   "IN"."BusinessPartnerID"
54  "OLD"."BusinessValidFrom"  ORDER BY "IN"."BusinessValidFrom") AS "NX_",
55  LEAD("IN"."BusinessValidFrom",1,"OLD"."BusinessValidTo") OVER (PARTITION BY   "IN"."BusinessPartnerID
56  "OLD"."BusinessValidFrom"  ORDER BY "IN"."BusinessValidFrom") AS "NY_",
57  ROW_NUMBER() OVER (PARTITION BY   "IN"."BusinessPartnerID",
58  "OLD"."BusinessValidFrom"  ORDER BY "IN"."BusinessValidFrom") AS "NR_",
59      "OLD"."BusinessPartnerID" AS "OLD_BusinessPartnerID" ,
60      "IN"."BusinessPartnerID"  ,
61      "OLD"."BusinessValidFrom" AS "OLD_BusinessValidFrom" ,
62      "IN"."BusinessValidFrom"  ,
63      "OLD"."BusinessValidTo" AS "OLD_BusinessValidTo" ,
64      "IN"."BusinessValidTo"  ,
65      "OLD"."SystemValidFrom" AS "OLD_SystemValidFrom" ,
66      "OLD"."SystemValidTo" AS "OLD_SystemValidTo" ,
```

Versioning - Temporal Tables

Using HANA native SQL:2011 features for automatic versioning

.HDBSYSTEMVERSIONING



System versioning file for table

```
SYSTEM VERSIONING
"sap.fsdm::AccountHolderOfExternalSettlementAccou
nt" ("SystemValidFrom", "SystemValidTo")
HISTORY TABLE
"sap.fsdm::AccountHolderOfExternalSettlementAccou
nt_Historical" NOT VALIDATED
```

Outlook: Versioning in HANA – SPS 04

Versioning for application time since SPS 04 possible with DB methods:

```
CREATE TABLE Partner(
    ID INTEGER,
    FirstName VARCHAR(64),
    LastName VARCHAR(64),
    City VARCHAR(64),
    BusinessValidFrom DATE CS_DAYDATE NOT NULL,
    BusinessValidTo DATE CS_DAYDATE NOT NULL,
    SystemValidFrom LONGDATE CS_LONGDATE NOT NULL GENERATED ALWAYS AS ROW START,
    SystemValidTo LONGDATE CS_LONGDATE NOT Null GENERATED ALWAYS AS ROW END,
    PERIOD FOR SYSTEM_TIME ( SystemValidFrom, SystemValidTo ),
    PERIOD FOR APPLICATION_TIME ( BusinessValidFrom, BusinessValidTo ),
    PRIMARY KEY( ID, BusinessValidFrom, BusinessValidTo )
)
    WITH SYSTEM VERSIONING HISTORY TABLE "PARTNERHISTORY"
;
```

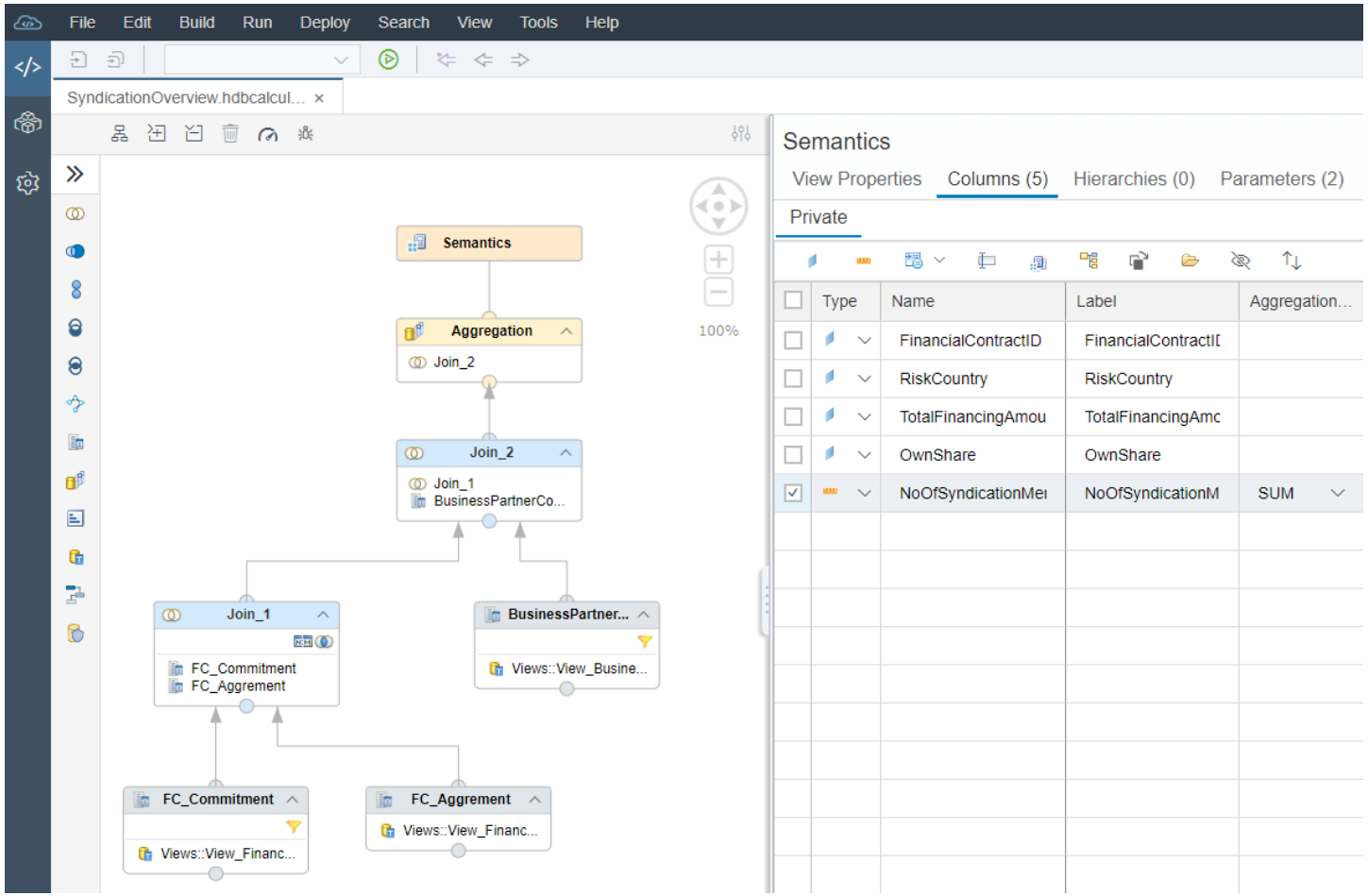
Read access to data

- Generated read interfaces that reflect customer enhancements to the data model
  - Direct access to tables without intermediary “index table”
- HANA Table functions
  - Input parameters for selection of bi-temporal time stamps
  - HANA engine for parallel execution
- CDS views for joined views on data
  - Regulatory reporting views
  - Accounting views (tbd)

Reporting and calculation logic

Calculation views

- Using (multiple) read interfaces as input
- Can have multiple layers of calculation logic
- You can create joins, unions, projections, complex functions and aggregation on the data flow
- Optimization of the execution plan (filter pushdown) handled in the background
- Calculation views can be stacked (cubes) to and called by other solutions



Conclusion

What have we learned

- Data model and access APIs are crucial for efficient in-memory computing
  - Software provider has to make sure that APIs and SQL statements for legacy applications that run on in-memory databases are optimized
  - Customer applications have to be checked and re-designed
  - Acceleration has its limit on legacy applications

- Native in-memory applications
  - Overcome limitations of legacy applications
  - Migration, stability, maturity, functional range?

- Customer developments on in-memory databases
  - More freedom and optimization potential
  - Think about access pattern and possible use cases

Is FSDP the answer for the financial industry DWH?

- Built on latest SAP technology, code to data paradigm can be achieved
- Data model to start with – Standardization benefits
- Can benefit from seamless and virtual integration to SAP and 3rd party systems

- More content has to be provided to set FSDP apart from native SQL warehouse implementations
- TCO compared to other options?
- Roadmap of SAP ?

Contact

Dirk Vollmer

Solution Architect

+49 176 1933 3529

dirk.vollmer@adweko.com

ADWEKO Consulting GmbH

Altrottstraße 31

69190 Walldorf

+49 6227 7337 90

info@adweko.com

ADWEKO