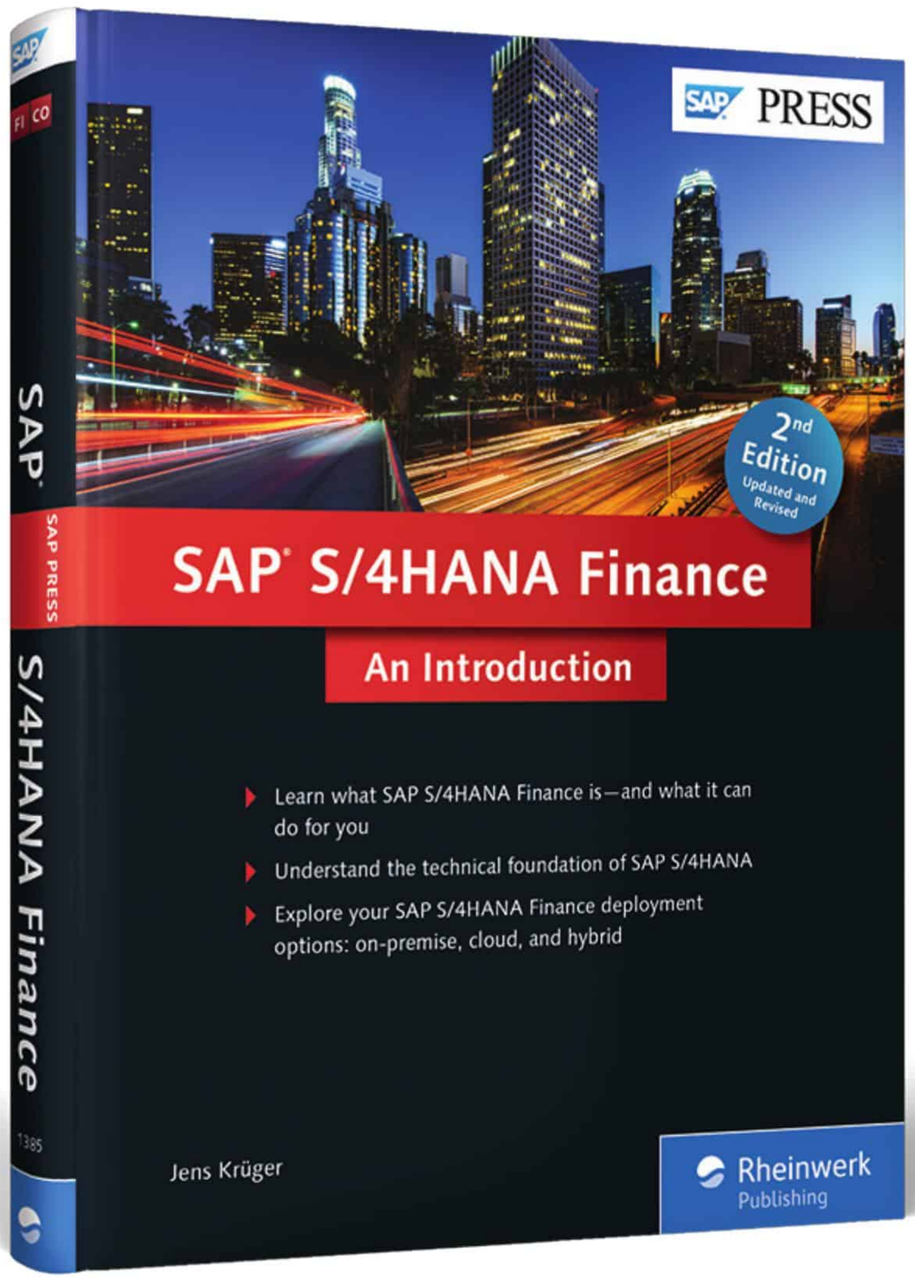


Reading Sample

Take a look at Chapter 3, which shows how a redundancy-free system is feasible thanks to SAP HANA. This chapter demonstrates how SAP S/4HANA Finance eliminates redundancy on different levels, and highlights the immediate benefits of the removal of redundant data for customers of SAP S/4HANA Finance.

"Removal of Redundancy"

Contents

Index

The Author

Jens Krüger

SAP S/4HANA Finance

An Introduction

411 Pages, 2015. $69.95/€69.95
ISBN 978-1-4932-1385-6

www.sap-press.com/4122

© 2016 by Rheinwerk Publishing Inc. This reading sample may be distributed free of charge. In no way must the file be altered, or individual pages be removed. The use for any commercial purpose other than promoting the book is strictly prohibited.

This chapter covers the benefits of a system without data redundancy, how such a system is now feasible thanks to SAP HANA, and how SAP S/4HANA Finance eliminates redundancy. We highlight the immediate benefits of the removal of redundant data for customers of SAP S/4HANA Finance.

3 Removal of Redundancy

Redundantly kept data—that is, data derived from other data available elsewhere in the database—is one of the big challenges of software systems.

You may wonder why you would store, for example, the sum of two invoice amounts separately if the same value can easily be derived by calculating it on the fly from the two invoices. In the past, such data redundancy was only introduced to increase performance, because traditional databases could not keep up with user expectations in light of billions of data entries. This came at the costs of significant effort to keep the redundant data consistent, increased database storage, and more complex systems.

Now that SAP HANA improves performance radically, as outlined in the previous chapter, the need for redundancy vanishes. Based on a *single source of truth*, derived data can be calculated on the fly instead of being physically stored in the database. Hence, in the spirit of simplification, getting rid of redundancy is a key paradigm of SAP S/4HANA Finance. We begin our exploration of the paradigms underlying the SAP S/4HANA Finance model by looking at the removal of redundancy in this chapter.

This chapter first introduces the conceptual benefits of a redundancy-free system by contrasting it with the disadvantages of redundant data (Section 3.1). Section 3.2 demonstrates how SAP HANA and in-memory technology enable us to overcome redundancy and avoid its pitfalls. In Section 3.3, we explore how SAP S/4HANA Finance simplifies the core financials data model by nondisruptively replacing materialized redundant data with on-the-fly calculation. Finally,

Section 3.4 highlights the immediate benefits of this simplification for companies switching to SAP S/4HANA Finance.

3.1 Benefits of a Redundancy-Free System

Data redundancy occurs if data is repeated in more than one location—for example, in two separate database tables—or can be computed from other data already kept in the database. In general, all data that you can also derive from other data sources of a system is redundant. Redundant data is distinguished by the fact that you need to maintain its consistency. If you modify the data at one location, then you need to apply corresponding modifications at all the other locations in the database in order to keep the relationship intact and avoid anomalies. Redundant data is high-maintenance data, and it won't take care of itself.

Software engineers spotted the fundamental problems of data redundancy early on. In 1970, Edgar Codd introduced the fundamental relational model that underlies many of today's database systems.[1] In order to reduce redundancy, normalization techniques such as normal forms are an integral part of the relational model.[2]

However, data redundancy remains. We can distinguish four different kinds of redundancy within a database system, depending on the relationship of redundant data to the original data:

- **Materialized view**
  A *materialized view* stores a subset of data from one or several tables as duplicate copies in a second, new, typically smaller table. It materializes the result of a *database view*, which is a stored database query, in order to provide direct access to the result. As such, the materialized view can be compared to an index in the base table. The query may also join several tables.
- **Materialized aggregate**
  If the query of a materialized view aggregates tuples from the base table(s), then we speak of materialized aggregates as a special case. A *materialized aggregate* does not contain one-to-one duplicates, but nevertheless the data is redundant,

1 Edgar Codd, "A Relational Model of Data for Large Shared Data Banks."
2 Edgar Codd, "Further Normalization of the Data Base Relational Model."

because it can be derived from the original data at any time by applying the same calculation on the fly.

- **Duplicated data due to overlap**
  Data may be duplicated if related information is stored in separate locations. Part of the data attributes overlap, whereas others are present only at one location. The overlap is then a source of redundancy. The different locations are often due to a separation of concerns, whereas the overlap again stems from performance reasons—for example, to avoid joins that are costly in traditional database systems.
- **Materialized result set**
  Long-running programs that arrive at a certain output after several steps of calculation often store the result of their work as a *materialized result set*. Due to the complexity of the program logic, the result is not described as a query but is created by running the program, which then stores its output for fast future reference.

None of these instances of redundancy is necessary from a functional point of view. They have been introduced in the past into enterprise systems to improve query response times in view of slow, disk-based database systems, which made an on-the-fly calculation prohibitively expensive. In-memory technology now breaks down the performance barrier so that SAP S/4HANA Finance can remove all kinds of redundancy. This chapter covers the removal of redundancy with regard to the first two categories: materialized views and materialized aggregates. We'll see these categories come up again later in the book. As described in Chapter 6, eliminating the need for long-running batch jobs makes materialized result sets obsolete in a real-time finance system. The Universal Journal covered in Chapter 8 describes a case in which SAP S/4HANA Finance eliminates duplicated data due to overlap between financial components.

If you look beyond a single database, redundancy also occurs when duplicating the same or derived data in several database systems. In contrast to redundancy within the same database, this cross-system duplication may be wanted in some cases—for example, for data security purposes. In other cases, it is truly redundant—for example, in the case of data warehouses, due to missing analytical capabilities of traditional database systems. As outlined in Chapter 2, this kind of data redundancy on a system level is inherently superfluous because SAP HANA combines OLTP and OLAP capabilities with superior performance.

Coming back to redundancy within a database, what are the benefits of having a redundancy-free system? Why is having a single source of truth in SAP S/4HANA Finance preferable compared to the traditional world of materialized views that serve to improve the performance of read operations?

The fundamental problem with redundant data is the effort necessary to keep it consistent when the original data changes. If you materialize the total amount of all orders of a certain customer, then you need to update that figure every time a new order by that customer comes in. Materializing a figure that needs to be updated frequently—for example, the balance of an often-used account—may even lead to contention, because the balance entry needs to be locked on each debit or credit on the account.

Similarly, a materialized view containing a duplicated subset of accounting data—for example, all open invoices—needs to be reconciled whenever the original data changes. If a customer's payment clears an open invoice, then the payment also has to be removed from the corresponding materialized view.

Regardless of whether this reconciliation needs to be performed manually by each business transaction or is handled automatically by the database system, maintaining consistency leads to additional database operations on top of the actual modifications. Otherwise, your database has more than one "truth"—which means that it is inconsistent. Getting rid of such reconciliation activities is the source of several benefits:

- **Simplicity**
  Overall, the architecture and data model of a redundancy-free system are significantly simpler. There are fewer dependencies and constraints that transactions need to take into account when modifying or analyzing data. As a consequence, transactions are also simpler. The overall simplification also manifests in a few other upcoming benefits.
- **Consistency maintained by design**
  Business transactions that modify data do not need extra work in order to maintain consistency. This makes for simpler programs based on a less complex data model with fewer dependencies. Overall, the architecture of a redundancy-free system is more natural: Transactions record what happens in the database by adding corresponding database records. The system provides everything else as read-only algorithms on top of the data.

- **Increased throughput**
  Because it is no longer necessary to modify redundant data in parallel, the number of database operations per business transaction diminishes. Only the essential operations to record the transaction occur, and no additional modifying operations are necessary. Fewer database operations and thus shorter duration of each transaction increase the throughput of the whole system so that it can handle more business transactions. Furthermore, materialized aggregates no longer have to be locked for updating, thus removing any contention issues.
- **Smaller database footprint**
  Redundant data takes up memory and hard disk space that is no longer needed in a redundancy-free system, which shrinks the overall database footprint. A smaller footprint reduces not only the requirements on the main database server but also in turn the disk space needed for backups. A smaller database footprint and higher throughput make a system more cost-efficient. As a consequence, a redundancy-free system has a lower TCO.
- **Flexibility in real time**
  A redundancy-free system also implies that it is no longer necessary to prebuild aggregates for performance reasons. Fast responses to analytics questions no longer depend on the availability of a materialized answer. Instead of being restricted to questions that can be answered based on what was foreseen when designing the data model of a system, users are given the flexibility to ask any question that can be answered in real time based on the original data. An in-memory redundancy-free system opens up this new level of flexibility with the performance expected from today's fast web applications.

The obvious question that arises from this list of impressive benefits is this: Why haven't all systems always been free of redundancy? Also, why wasn't SAP ERP Financials historically an exception either?

In a traditional disk-based database system, the answer in the past was "performance reasons." Now, in an SAP HANA world, the answer is that there is no excuse left for redundancy, thanks to the speed of in-memory technology. In the past, query response times (especially involving aggregation) were too slow for productive use without materialization. The costs of a redundancy-free system were too high, even compared to the benefits. These days, the superior performance of in-memory database systems means that the fear of slow response times is a thing of the past, as demonstrated in the following section. Figure 3.1 illustrates that, in contrast to the traditional database world, the benefits of a

redundancy-free system clearly outweigh the costs, as we'll see applied to the case of SAP S/4HANA Finance in Section 3.3.

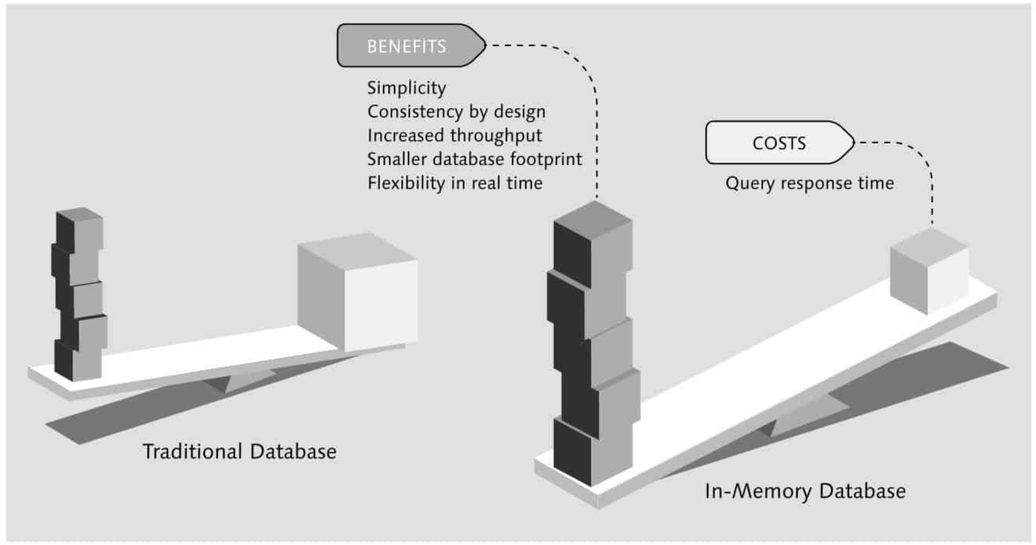


Figure 3.1 Replacing Materialization with On-the-Fly Calculation

3.2 In-Memory Technology Removes Redundancy

For decades, data redundancy has been reluctantly accepted in order to reach sufficient performance. The slow response times of disk-based database systems made a redundancy-free data model impossible. Because latencies and bandwidth of disk access are orders of magnitudes slower than in-memory access, calculating totals on the fly was prohibitively expensive.

Now, a new era is upon us. As outlined in Chapter 2, in-memory technology fundamentally challenges long-standing assumptions. In this section, we demonstrate that SAP HANA indeed enables you to get rid of redundancy without compromising performance.

These days, on-the-fly aggregation is feasible. In order to get rid of redundancy without disrupting businesses processes, on-the-fly calculations replacing materialized aggregates in an in-memory system have to perform at least as fast as users are accustomed to, based on materialization in a disk-based database system. The

comparison is thus between *disk-based with materialization* on the one hand and *in-memory with on-the-fly calculation* on the other.

Let's take a closer look. For our demonstration, we'll examine a disk-based system with a typical disk latency of 10 ms (i.e., accessing a random location on disk takes ten milliseconds, or ten million nanoseconds). In comparison, in-memory latency is only 100 ns (0.1 ms). In other words, the database can make one hundred thousand random accesses in the same time as one disk access. The memory bandwidth in case of sequential reads is 4 MB per millisecond per core. For the following example, we consider a simple server that has a single CPU with eight cores. Modern server systems often connect several CPUs, each with 10 cores or even more, further increasing the processing power. For our example, we assume a company with one hundred million accounting document line items in its database and fifty thousand customers.

Total sales in the current month to a particular customer is a typical aggregate value of interest in financial operations. In the disk-based scenario with materialization, a materialized aggregate for this kind of analysis will contain one tuple per customer per month that gives direct access to the answer to such an analysis query. Hence, a user can expect to receive an answer from a disk-based database with materialization after 10 ms, assuming the database system needs a single disk access to retrieve the value and ignoring any further processing for comparability.

In an in-memory system without materialization, the aggregate value is not precomputed as part of every business transaction. As a consequence, calculating the answer to the query takes more steps. Nevertheless, response times are faster, as we'll show. Because the calculation happens transparently for applications accessing the database, the calculation steps will not be noticed by users. To keep the explanations simple, the following sequence of steps assumes a sequential execution of the query. Another advantage of an in-memory system is the in-built parallelization, which further speeds up the execution of more complex queries. The steps are as follows:

1. **Select all accounting document line items for a particular customer.**
   In a column table, the column representing the customer associated with each line item is stored in one continuous block of memory. To identify all items of a particular customer, the database system scans the whole column and keeps track of the positions where the customer number in question appeared.

Thanks to the columnar layout, one such full column scan operates with the full bandwidth of 4 MB per millisecond and core. SAP HANA's built-in compression means that only the compressed integer representations of customer numbers need to be compared. Because there are fifty thousand different numbers, the compressed representation of each customer number only needs two bytes (216 = 65,536 different values). Hence, the customer column for 100 million line items takes up 200 million bytes, or roughly 190 MB. With eight cores, scanning this attribute vector takes 6 ms (190 MB divided by 8 times 4 MB per second).

2. **Apply further selections.**
   For each of the items from Step 1, the database next applies further selection conditions, such as the month. For each additional condition, this mandates a lookup in the corresponding attribute vector at every position identified so far. With two thousand line items per customer on average, this leads to two thousand random accesses in a worst-case scenario with entirely noncontiguous positions. Even in that case, this takes only 0.02 ms on a single core (2,000 accesses times 10 ns per access = 20,000 ns).
3. **Add up the sales amount of all line items.**
   After all relevant items have been identified in the previous steps, the database retrieves the sales amount for each item and adds it to the result. Again, the database makes one random access per position in the attribute vector for the sales amount and resolves the dictionary-encoded value in one additional access to the dictionary. Even if Step 2 didn't exclude further positions, this requires four thousand accesses and 0.04 ms on a single core.

For the total response time, we can ignore Steps 2 and 3, because they don't contribute significantly to the overall time. In total, on-the-fly calculation of sales to a particular customer in the current month takes approximately 6 ms in an in-memory database system—less than the 10 ms for accessing the materialized aggregate in a disk-based database. Replacing materialization with on-the-fly calculation is thus entirely feasible in this example.

In the case of more complex queries, an in-memory system with on-the-fly calculation may even increase its advantage; for example, a similar query to the one just considered but for all months of the current year instead of only the current month would require up to twelve disk accesses to the materialized values stored per month (unless an additional materialized aggregate is introduced). Although the response time suffers a corresponding increase in the disk-based system up to

120 ms, the on-the-fly calculation only needs to adapt the further selection criteria, keeping the response time of the in-memory system almost the same at roughly 6 ms.

We've already demonstrated the desirability of a redundancy-free system by highlighting the benefits. In addition, these calculations demonstrate its feasibility thanks to the faster in-memory performance. What is possible with fast and stable access times today would have been prohibitively expensive in a traditional disk-based system. Depending on the caching strategy, a disk-based database has to retrieve answers from disk, suffering long latency and slow bandwidth. Compared to that, an in-memory database offers not only faster but also more stable response times, because latency and bandwidth do not vary as they do in a disk-based system that frequently needs to go back to disk to retrieve data.

How does SAP S/4HANA Finance optimize its new possibilities to create a redundancy-free system?

3.3 Simplifying the Core Data Model

As outlined previously, enterprise applications in the past needed to store data redundantly in order to meet performance expectations of their users in view of limited database performance. Applications that remain bound to traditional disk-based databases still experience these limitations. SAP S/4HANA Finance is based on SAP HANA, so it makes use of the dramatically improved performance of an in-memory database. At the same time, its data model is a nondisruptive evolution of the SAP ERP Financials data model, removing any redundancy that has historically been necessary for performance reasons.

Looking at the data model of SAP ERP Financials, several instances of data redundancy quickly become apparent. The fundamental separation of different components (such as Financial Accounting, Controlling, Profitability Analysis, and others) into separate table structures is a case of duplicate data. Chapter 8 describes how the integration of these previously separate components and data models into a Universal Journal enables radically new approaches to finance in addition to the usual benefits of a redundancy-free system.

The data model of each of these components in turn contained data redundancy in terms of materialized views and materialized aggregates. To explain the

changes on this foundational level (essentially the first steps toward a redundancy-free system, completed by the Universal Journal), we'll now take a closer look at the data model of Financial Accounting's G/L. The explanations similarly apply to the other components; they, too, have been simplified in the same spirit.

While we take a closer look at Financial Accounting, we'll reference the old tables from Financial Accounting for reasons of comparison; with the Universal Journal (see Chapter 8), you can merge the data structures of hitherto separate components. The fundamental and essential data tuples of every financial accounting system—besides master data—are the accounting documents and their line items. The system records each transaction as an accounting document with at least two line items (one each for debit and credit entry), but potentially more.

Faced with millions or billions of accounting documents (headers, primarily stored in table BKPF) and their line items (table BSEG) and slow disk-based performance, SAP ERP Financials needed materialized views and materialized aggregates in order to provide sufficiently fast access to line items with specific properties or to aggregate values. For this reason, the core data model of Financial Accounting (as illustrated in Figure 3.2) contained, among others, six materialized views (three for open line items separated by accounts receivable, accounts payable, and G/L accounts, and three for cleared line items separated in the same manner) and three materialized aggregates for corresponding totals.

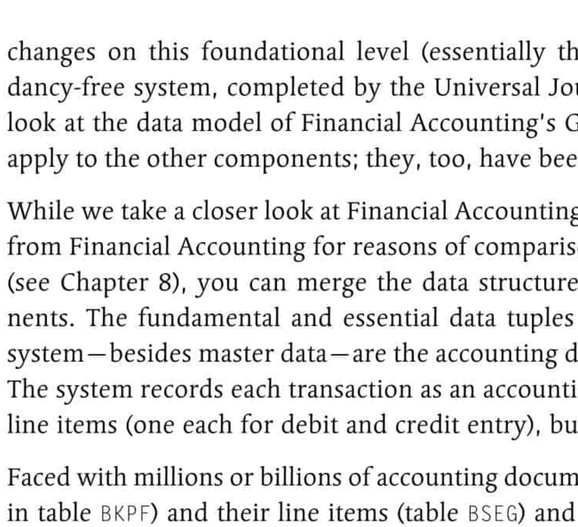


Figure 3.2 Old Data Model of SAP ERP Financials

For example, in an SAP ERP Financials system the materialized view of all open accounts receivable line items (table BSID) contains a copy of each line item (with a subset of attributes) from the original table of accounting document line items that fulfills the following condition: the line item is open (i.e., has not been cleared) and is part of the accounts receivable subledger. Needless to say, when a transaction clears the item, it also has to delete the corresponding tuple from the materialized view of open accounts receivable items and add it to the corresponding materialized view of cleared accounts receivable items.

In contrast, in SAP S/4HANA Finance all of these materializations have been removed in order to take the first step toward eliminating redundancy. The corresponding tables have been replaced with compatibility views. From the accounting documents and line items as the single source of truth, any derived data can be calculated on the fly, most times with higher performance than in a traditional disk-based system. This includes, but of course is not limited to, the views and aggregates that previously existed in materialized versions.

Figure 3.3 illustrates the resulting redundancy-free data model of Financial Accounting (before the Universal Journal) that is functionally equivalent to the previous data model.

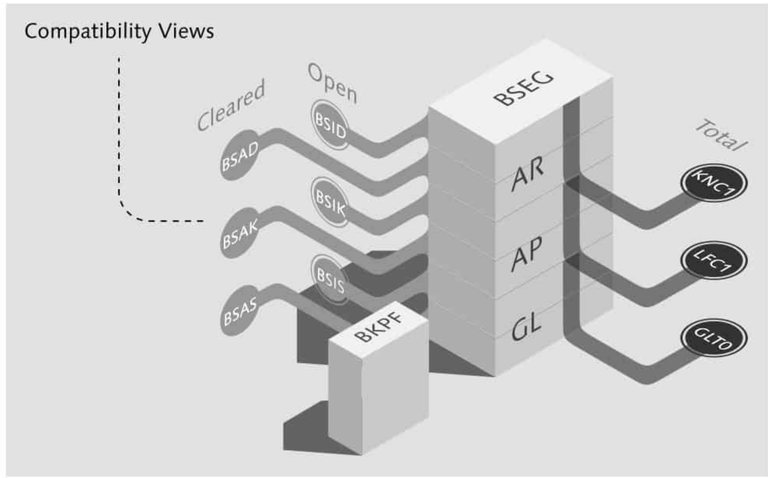


Figure 3.3 New Redundancy-Free Data Model of Financial Accounting

In the spirit of simplification, it consists only of the essential tables for accounting documents and for accounting document line items that record the business

transactions. In addition, the compatibility views transparently provide access to the same information that was redundantly stored in materialized views and materialized aggregates before. These redundant tables are in turn obsolete, because SAP HANA calculates the same information on the fly. Appendix A lists the tables that have been replaced with compatibility views.

The compatibility views bear the same name as their historical predecessors to ensure that the changes are nondisruptive and do not require SAP customers to modify their custom programs. Any program—be it part of the SAP standard or a customer modification—that in the past accessed the materialized view of open accounts receivable line items is now seamlessly routed to the corresponding compatibility view. The compatibility view calculates the result for each query on demand—without compromising performance, thanks to SAP HANA (as explained in Section 3.2). In this case, the view selects the open items belonging to the accounts receivable subledger directly from the original table of accounting document line items. Any additional selection conditions—for example, a specific customer—are immediately passed through to the query optimizer and integrated into the query execution plan.

As mentioned at the beginning of this section, the same approach applies to other components as well. For example, materialized aggregates on top of controlling documents are no longer necessary, opening up the ability to combine different accounting components in the Universal Journal. As outlined in Chapter 11, Section 11.4, cash management is another area with similar changes that remove data redundancy.

In summary, the SAP S/4HANA Finance data model is now entirely based on line items, without any prebuilt materialized aggregates or other data redundancy. Not only is the data model simpler, but the program architecture is also simpler: The system "simply" records all business transactions as they happen. Everything else is calculated on the fly by algorithms on top of the data. Without any negative effect on your existing investments in SAP systems, you immediately benefit from switching to SAP S/4HANA Finance. Let's look at how.

3.4 Immediate Benefits of the New Data Model

Removing materialized views and materialized aggregates from the financial accounting data model has an immediate positive impact on the transactional

throughput of the system. In the case of SAP S/4HANA Finance, posting an accounting document requires neither inserting redundant duplicates into materialized views nor updating redundant aggregate values. The corresponding effort and database operations to maintain consistency are no longer necessary.

As a consequence, the number of tuples inserted or modified during database operations was indeed cut by half according to experimental measurements. These experimental measurements are based on real-world data of a large SAP customer. Five hundred accounting documents were posted, each with six line items. Instead of twenty-six thousand tuples affected by `UPDATE`, `INSERT`, and `DELETE` operations in a traditional SAP ERP Financials system already running on SAP HANA, SAP S/4HANA Finance only needed to insert eleven thousand database tuples into the tables of the financials component for the entire test—a savings of more than a factor of two.

Of course, fewer tuples translate directly to less end-to-end transaction time spent posting a document. Instead of over 200 ms per document in SAP ERP Financials on SAP HANA, posting in SAP S/4HANA Finance only required 100 ms from end to end, down by a factor of two. As a consequence, the throughput of a system running SAP S/4HANA Finance doubled in this scenario.

The experimental measurements by design did not even include the effect of contention that often appears in systems with data redundancy. Enterprise systems usually handle a lot of transactions in parallel. In the case of materialization, the concurrent aggregate updates in particular can lead to contention, because materialized aggregates have to be locked for updating. In the case of, for example, heavily used G/L accounts, the database system has to handle the otherwise parallel postings sequentially if they access the same G/L account in order to consistently update the totals for this account. This unfortunate situation can no longer occur in SAP S/4HANA Finance, because all transactions simply insert tuples into the database, which does not require locks.

When considering the overall system architecture, the statement that in-memory databases with a column-oriented architecture are not as fast when it comes to modifying operations as they are for reading access no longer holds true. The measurements for SAP S/4HANA Finance show that the speed even of modifying transactions is on par with, or better than, that of SAP ERP Financials running on a traditional database with row-based storage.

The database footprint is another area of improvement. Again focusing on the Financials component, the removal of redundancy in SAP S/4HANA Finance alone has the potential to drastically reduce the database footprint. Memory previously occupied by redundant data, such as materialized views and materialized aggregates, is no longer needed. The database footprint of SAP SE's own internal system for Financials (the main productive SAP system within SAP) has been reduced by a factor of almost three; gaining additional savings is possible by applying concepts such as data aging so that in total a reduction by a factor of fourteen is feasible. Calculations based on SAP customer data show equally impressive numbers—for example, a reduction by a factor of 6.5.

Such a reduction in database footprint immediately translates to hardware savings and a lower TCO. In addition, SAP HANA also enables you to remove your separate data warehouse thanks to integrated OLAP capabilities. By doing so, you gain an additional factor of two across the whole footprint. The numbers shown in both of these cases are for a system already running on SAP HANA as the database. Factoring in additional savings in the database footprint enabled by SAP HANA's storage architecture and compression, the potential savings are even more impressive.

In summary, SAP S/4HANA Finance demonstrates that in-memory technology makes it entirely feasible to remove redundancy from the Financials data model. Redundancy is no longer necessary for reporting or analysis purposes; these tasks can be run in high speed directly based on line items as algorithms, instead of relying on materialized aggregates prebuilt into the data model. As a consequence, users are no longer restricted to only those analytics questions that have been hardwired into the system. Instead, they are free to analyze data flexibly as needed with the performance they expect.

In this chapter, we highlighted the benefits you gain from the redundancy-free data model of SAP S/4HANA Finance. Benefits fall both into the area of TCO reduction (increased throughput and lower database footprint) and entirely new levels of flexibility. Overall, the removal of redundancy in SAP S/4HANA Finance yields a significant simplification of system and processes. We demonstrated the feasibility of these fundamental changes in terms of performance and highlighted some of the key improvements as seen in real-world environments.

We'll see these topics elsewhere in this book. The next chapter outlines the non-disruptive nature of the innovations in SAP S/4HANA Finance, which allows you

to switch seamlessly and immediately benefit from the advantages outlined here and throughout the book. Chapter 5 explores the ensuing possibilities in terms of flexibility based on the purely line item–based data model. We focused on the removal of redundancy in the form of materialized views and materialized aggregates in this chapter, and the removal of materialized result sets is explained in Chapter 6 as one of the benefits of a real-time finance system that doesn't require batch jobs. Chapter 8 outlines how further duplicate data stores are unified with the Universal Journal, which is the next big step toward a redundancy-free system that eliminates the need for reconciliation.

Contents

Foreword 15
Preface 17

1 Introduction 27

1.1 Market Trends 27
1.2 Leadership Trends 29
1.3 Technology Trends 32
1.4 Introducing SAP S/4HANA Finance 33
1.4.1 Deployment Modes 37
1.4.2 Transformative Finance Technologies 40
1.4.3 Key Areas of Innovation 45

PART I Exploring the SAP S/4HANA Finance Model

2 In-Memory Technology and SAP HANA 65

2.1 Keeping Data in Memory 67
2.1.1 Ensure Durability 68
2.1.2 New Performance Bottleneck 69
2.2 Columnar Data Organization 69
2.3 Data Encoding and Compression 72
2.3.1 Dictionary Encoding 72
2.3.2 Data Compression 73
2.3.3 Operation on Compressed Data 74
2.4 Parallel Execution 75
2.4.1 Parallel Execution in Columnar Store 75
2.4.2 Parallel Aggregation 76
2.5 Delta Store and Merge 77
2.5.1 Insert 78
2.5.2 Delta Store 78
2.5.3 Delta Merge 79

3 Removal of Redundancy 81

3.1 Benefits of a Redundancy-Free System 82
3.2 In-Memory Technology Removes Redundancy 86

3.3 Simplifying the Core Data Model 89
3.4 Immediate Benefits of the New Data Model 92

4 Nondisruptive Innovation 97

4.1 Compatible Refactoring 98
4.2 Transitioning to SAP S/4HANA Finance 101

5 Unlimited Flexibility Based on Line Items 105

5.1 Gaining Unparalleled Insights 105
5.2 Removing Predefined Aggregates 107
5.3 Deciding without Information Loss 110
5.4 Optimizing Operations with Big Data 113
5.5 Innovating with Line Item Granularity 115

6 Finance in Real Time 117

6.1 Accepting the Limits of Retroactive Analysis 118
6.2 Benefitting from Real-Time Finance 120
6.2.1 Improving Profit 121
6.2.2 Securing Liquidity 123
6.2.3 Managing Risk 124
6.3 Identifying Real-Time Finance Patterns 126
6.3.1 Data Cases 126
6.3.2 Method Cases 127
6.3.3 Process Cases 127
6.3.4 Ad Hoc Cases 128
6.4 Becoming a Real-Time Finance Company 128
6.4.1 Balancing Efficiency and Insights 128
6.4.2 Examining Industry Requirements 130
6.4.3 Considering Business Users' Needs 131

PART II Using SAP S/4HANA Finance

7 Building Blocks of SAP S/4HANA Finance 135

7.1 Seeing the Big Picture 135
7.2 Reloading SAP ERP Financials Functionality 138
7.2.1 Financial Accounting 138

7.2.2 Management Accounting 140
7.2.3 SAP Financial Supply Chain Management 142
7.2.4 SAP Treasury and Risk Management 142
7.2.5 Cash Management 143
7.3 New Applications in SAP S/4HANA Finance 143

8 Single Source of Truth: Universal Journal 147

8.1 Mastering Business Challenges without SAP HANA 147
8.2 New SAP HANA-Based Architecture 151
8.2.1 Structure of the Universal Journal 152
8.2.2 Merging G/L Accounts and Cost Elements 154
8.2.3 Compatibility Views 155
8.3 Immediate Benefits of the New Model 156
8.3.1 Depreciation Runs 157
8.3.2 Multidimensional Profitability Analysis 158
8.3.3 New Analysis Patterns 160
8.3.4 Financial Statement Analysis 161
8.4 Building on Top of the Universal Journal 162
8.4.1 Fundamental Functional Enhancements 162
8.4.2 Important Simplification Cases 163
8.4.3 Prediction Based on Facts 163

9 Flexible Reporting 169

9.1 The Challenge of Flexible Operational Reporting Needs 170
9.2 Addressing Challenges Using SAP S/4HANA Finance 173
9.3 Conceptual Advances in Reporting 178
9.3.1 Merging of Accounts and Cost Elements 178
9.3.2 Accounts and Other Hierarchies 179
9.4 Advances for End Users 180

10 Flexible Analytics with Core Data Services 185

10.1 Replacing Complex Analytics with Simplicity 186
10.2 Paradigms of Core Data Services in SAP S/4HANA 187
10.3 Building Blocks of Core Data Services in SAP S/4HANA 190
10.3.1 Domain-Specific Database Languages and Services 192
10.3.2 Use of Semantic Annotations 192
10.3.3 Virtual Data Model 193
10.4 Unified View of Data for Analytics 194

11 SAP S/4HANA Finance Applications 197

11.1 Business Cockpits for Businesspeople 197
11.1.1 Working with Business Cockpits 199
11.1.2 Defining a Business Cockpit 201
11.1.3 Outlook 203
11.2 Financials Operations 205
11.2.1 Receivables Management 206
11.2.2 Payables Management 209
11.2.3 Outlook 211
11.3 Ariba Network Integration 212
11.3.1 Order Collaboration 213
11.3.2 Invoice Collaboration 214
11.3.3 Discount Management 215
11.3.4 Outlook 216
11.4 Cash Management 216
11.4.1 Bank Account Management 219
11.4.2 Cash Operations 220
11.4.3 Liquidity Management 223
11.4.4 Simplified Data Model 225
11.4.5 Outlook 226
11.5 SAP Business Planning and Consolidation for Finance 227
11.5.1 Business Contexts 228
11.5.2 Underlying Architecture 229
11.5.3 Planning Application 232
11.5.4 Business Planning Process 233
11.5.5 Outlook 234
11.6 Profitability 236
11.6.1 SAP Fiori Applications 238
11.6.2 From Costing-Based to Account-Based Profitability Analysis 238
11.6.3 Outlook 242
11.7 Financial Closing and Consolidation 242
11.7.1 Process Management 243
11.7.2 Avoiding Reconciliation Efforts 244
11.7.3 Faster Period Close with SAP HANA 245
11.7.4 New Closing Applications 246
11.7.5 Outlook 248
11.8 Revenue Accounting 249
11.8.1 Contract Identification 250
11.8.2 Performance Obligation Identification 251

11.8.3 Transaction Price Determination 252
11.8.4 Transaction Price Allocation 252
11.8.5 Revenue Recognition 252
11.8.6 Flexible Adaptability 253
11.8.7 Outlook 253
11.9 Real-Time Governance and Compliance 254
11.9.1 Introduction to Governance, Risk, and Compliance 255
11.9.2 Changing Trends and Challenges of Staying Compliant 256
11.9.3 Embedded Real-Time Monitoring of High-Risk Transactions 258
11.9.4 Outlook 262

12 Revamping the User Experience with SAP Fiori Apps 263

12.1 SAP Fiori Design Philosophy 264
12.2 Developing Intuitive Products 265
12.3 SAP Fiori Design Principles in SAP S/4HANA Finance 266
12.4 SAP Fiori Application Framework and Finance 271
12.4.1 Launchpad 271
12.4.2 Application Types 272
12.4.3 Application Patterns and Controls 273
12.5 Extending SAP Fiori 274

PART III Running SAP S/4HANA Finance

13 Deploying SAP S/4HANA Finance 281

13.1 On-Premise Edition 282
13.1.1 Architecture Overview 283
13.1.2 New On-Premise Installations 284
13.1.3 Landscape Migration for Existing SAP ERP Systems 285
13.1.4 Handling General Ledger Functionality during Upgrade 286
13.1.5 Simplification List 287
13.2 Cloud Editions 288
13.3 Managed Cloud Options 289
13.3.1 Service Offerings in the Cloud 290
13.3.2 Hybrid Scenarios 293
13.3.3 Subscription Licensing and Options 294
13.4 Extensibility 296

14 Landscape Consolidation with Central Finance 299

14.1 Harmonization and Standardization 300
14.2 Principles and Capabilities 304
14.3 Evolution to an Operational System 309
14.4 Central Finance in the Cloud 312

15 Adoption Scenarios 313

15.1 Greenfield vs. Brownfield Approach 314
15.1.1 Greenfield Approach 314
15.1.2 Brownfield Approach 315
15.2 SAP S/4HANA Finance in SAP Business Suite on SAP HANA 316
15.2.1 Manufacturing Company 316
15.2.2 High-Tech Company 318
15.3 Central Finance and Central Reporting 319
15.3.1 Trade Company 319

16 Success Story: SAP SE Runs SAP S/4HANA Finance 323

16.1 Defining a Finance Vision 324
16.2 Moving to SAP S/4HANA Finance 326
16.2.1 System Landscape and Organization 327
16.2.2 Finance Transformation 328
16.2.3 Scope and Structure of the Project 329
16.3 Reaping the Benefits 334
16.4 Anticipating the Next Steps 338
16.5 The Takeaway 338

PART IV Moving Forward with SAP S/4HANA Finance

17 Design Thinking 343

17.1 Foundation of Design Thinking 344
17.2 Impacts on a Business Process Landscape 348
17.3 Identify and Define Changes 349
17.3.1 Team Setup 350
17.3.2 Problem Space 351
17.3.3 Solution Space 352
17.3.4 Packaging and Handover 353

17.4 Set Up a Design Thinking Workshop 354
17.5 Opportunities in the Business Process Landscape 355

18 Digital Enterprise and the Future of Finance 359

18.1 Behind the Term "Digital Enterprise" 360
18.1.1 Change Has Never Been So Rapid 361
18.1.2 Automation and Big Data Handling Are Becoming Imperative 362
18.1.3 Finance Is the Score Keeper of Business Performance 362
18.2 From Mechanics to Analytics 363
18.2.1 Eliminating the Drudgery of Accounting 364
18.2.2 Creating an In-Context Enterprise 366
18.2.3 Integrating Planning and Analysis with Daily Operations 367
18.2.4 Enhancing the Use of Information 368
18.3 Simulation and Predictive Analysis 369
18.4 Proactive Finance Solution 371
18.5 Summary 375

Appendices 377

A Changes to the Data Model 379
B Additional Information 381
C The Authors 391

Index 405

Index

A

ABAP, 99, 155, 283, 297
Accounting document, 90, 91, 301
Accounts payable, 46, 140, 205, 209, 211, 214, 228
Accounts receivable, 46, 139, 140, 205, 208, 211, 228
Actual data, 301
Actual line item, 371
Adjustment, 300
Adoption, 302, 313
Aggregated data, 107, 171
Aggregation, 74, 76
Allocation, 154, 172, 177, 234, 239, 241
Analytical apps, 272
Analytics, 136, 185, 189, 337
Analyze Liquidity Plan app, 225
Annotations, 192
Application Interface Framework (AIF), 306
Application management services (AMS), 291
Application patterns, 273
Application program interface (API), 297
Architecture, 301
Ariba Invoice, 214
Ariba Network, 212, 215, 285
Assessment service, 292
Asset Accounting, 46, 140, 151, 153, 157, 245, 310
Assignment cycle, 241
ASUG, 265
Automation, 362

B

Bad debt, 206
Balance sheet, 43, 119, 138, 151, 153, 161, 164, 179, 235
Bank Account Management, 50, 143, 219, 225, 303
Bank statement import status, 220
Batch job, 106, 118, 210, 230, 336
Big data, 290, 362
Bring-your-own-license (BYOL), 291
Brownfield, 314, 315
Budgeting, 118
Business add-in, 306
Business cockpit, 54, 144, 197, 198, 201
Business intelligence (BI), 55
Business model, 130, 290
Business network, 144, 212
Business Planning Framework (BPF), 233, 235
Business process, 348
Business Rule Framework plus (BRFplus), 249, 251
Business transformation, 106

C

Calculation principle, 118
Calibration, 260
Carbon Copy Invoices app, 214
Cash discount, 209, 215
Cash Discount Utilization app, 209
Cash flow, 49, 119, 205, 220, 223, 224, 228, 235, 370
Cash Management, 49, 92, 143, 144, 211, 216, 303
Cash position, 220
Cash Position app, 222
Central Finance, 103, 123, 244, 246, 294, 299, 302, 303, 310, 311, 312, 315, 319
Central Finance interfaces, 307
Central Journal, 303, 312
Centralization, 217, 311
Centralized finance, 48
CFO, 27, 121, 300, 327
Change management, 348
Chart of accounts, 170, 306
CIO, 37
Classic extensibility, 297
Classic G/L, 101
Classification problem, 373
Closing, 31, 45, 106, 144, 145, 148, 158, 170, 177, 242, 243, 245, 248, 335
Cloud, 23, 33, 282, 289, 293, 295, 312

Cloud deployment, 295
Code pushdown, 60
Coherent principle, 268
Collection worklists, 207
Collections Management, 142
Column scan, 74, 88
Column store, 69, 70, 71, 73, 75, 77, 78, 93
Company code, 176
Compatibility view, 91, 92, 98, 99, 101, 152, 155, 170
Compatible refactoring, 98
Complexity, 17
Compliance, 31, 255, 256, 257, 260
Concur, 281
Connected enterprise, 364
Consistency, 40, 82, 84
Consolidation, 47, 242, 243, 246, 248, 311
Consumption views, 193
Core data model, 89
Core Data Services (CDS), 186, 187, 198
ABAP, 190, 191
SAP HANA, 190
VDM, 187, 190, 193, 194
Correlation, 112, 114
Cost center, 117, 153, 157, 159, 160, 170, 171, 172, 173, 174, 176, 179, 229
Cost Center Accounting, 170
Cost element, 151, 153, 154, 170, 178, 179
secondary, 156
Cost Element Accounting, 141
Cost object, 306
Cost object structures, 307
Cost of goods sold, 240
Costing method, 363
CPU cache, 65, 69, 74
Credit Risk Analyzer, 142
Customer journey map, 351, 353
Customizing, 305

D

Dashboard, 201, 213, 259, 271, 371
Data aging, 334
Data availability, 185
Data bucket, 117
Data centers, 289
Data collection, 300
Data compression, 72, 73, 88
Data definition language (DDL), 191
Data dictionary, 99
Data distribution, 111
Data governance, 185
Data harmonization, 126
Data latency, 185, 232
Data privacy, 289
Data replication, 232
Data security, 289
Database footprint, 85, 94, 101, 157, 335
Database migration option (DMO), 285
Days sales outstanding (DSO), 57, 110, 206, 229, 272
Decentral system, 301, 306
Decentralization, 217
Delightful principle, 269
Delta merge, 79
Delta store, 77, 78
Delta version, 151
Deployment, 23, 33, 281
Depreciation, 158
Design process, 265
Design Thinking, 329, 343, 344, 349, 355
in business, 355
process, 345
workshop, 354
Detective control, 257, 258, 262
Device agnosticism, 266
Dictionary, 73
Dictionary encoding, 72, 73, 74
Digital economy, 361
Digital enterprise, 359, 363
Dispute Management, 142
Document splitting, 150, 153, 286, 318
Double-buffer concept, 80
Drill down, 122, 173, 176, 198, 200, 205, 220, 366
DSAG, 265
Due diligence, 109, 257
Dynamic models, 371

E

Early adopters, 361
EHP 7, 283, 284, 286, 292, 315
EHP 8, 283, 284, 285, 292
Error handling, 306
Exposure Hub, 226, 325
Extensibility, 151, 162, 296

F

Fact sheet apps, 272, 325
False negatives, 260
False positives, 260
Financial Accounting, 22, 90, 91, 136, 138, 147
Financial close, 34
Financial closing cockpit, 335
Financial operations, 142, 144, 205
Financial planning, 34
Financial statement, 148, 156, 167, 170, 173, 179, 241, 242, 249
Flexibility, 18, 94, 108, 136, 187
Forecast, 33, 118
Fragmented landscape, 105
Fraud, 337
Functional area, 171, 176
Functional Area Ledger, 149

G

General Ledger (G/L), 46, 90, 101, 139, 151, 153, 157, 162, 171, 176, 178, 229, 240, 243, 245, 253, 286, 318, 330, 334
account, 93, 151, 153, 154, 156, 159, 173, 178
Goodwill, 109
GR/IR, 189, 247
Greenfield, 282, 314, 319, 353

H

Hard disks, 68
Harmonization, 300, 302, 311, 325
Hierarchy, 108, 115, 369
High-performance application, 102
HTML5, 266, 283
Hybrid deployment, 281, 293

I

Ideate, 347, 353
IDEO, 344
In-context enterprise, 366
Individualization, 282
Industry requirements, 130
Information loss, 108, 110
Information silo, 105
In-memory computing, 32, 67, 85, 89, 93, 135, 261, 302, 324
In-memory database, 66, 70, 285
Input/output (I/O) access, 65
Insert, 78
Intercompany reconciliation, 61, 303, 317
Interface views, 193
Internal order, 153
International Accounting Standards Board (IASB), 249
International Financial Reporting Standards (IFRS), 139, 148, 153, 249, 250, 252
Invoice clearing, 190
Invoice collaboration, 214

J

Joint Venture Accounting, 148, 149
Journal entry, 365

K

Key figure, 239
Key performance indicator (KPI), 27, 50, 110, 122, 124, 198, 201, 205, 206, 218, 268, 369, 371
Key user extensibility, 297

L

Latency, 362
Ledger, 153, 154, 162
Licensing, 294
Line item, 43, 87, 90, 91, 136, 301
Liquidity, 123

M

Machine learning, 360, 364
Machine observation, 365
Main memory, 65, 68, 70
Managed extensibility, 297
Management Accounting, 22, 46, 136, 140, 147, 151, 153, 229, 243, 245
Margin, 122, 201
Margin Analysis app, 238
Market Risk Analyzer, 142
Market segment, 153, 159, 161, 164, 229
Material Document Table, 366
Material Ledger (ML), 153, 157
Materialization, 91, 93
Materialized aggregate, 82, 85, 86, 87, 88, 89, 90, 92, 94, 98, 99, 152, 335
Materialized result set, 83
Materialized view, 82, 84, 89, 90, 91, 92, 98, 152
Mergers, 109, 300
Migration, 285, 315, 316, 330, 334
Migration services, 292
Mobile, 268
Modifications, 291
Multidevice experience, 268
Multidisciplinary team, 350
My Spend app, 180, 200, 275

N

Navigation, 267, 272
Net margin, 200
New G/L (see also General Ledger), 139, 140, 150
Nondisruptive, 17, 89, 92, 97, 98, 103, 136, 152, 290, 331
Normalization, 82

O

OData, 193
OLAP, 40, 83, 94, 107, 157, 186, 226, 283, 301
OLTP, 40, 83, 107, 157, 186, 300, 301, 302
Onboarding services, 292
One price, 294
One-system strategy, 300
On-premise deployment, 23, 38, 281, 282, 291, 293, 312
On-the-fly calculation, 47, 83, 88, 92
Operational contract, 251
Operational efficiency, 206
Optimization problem, 373
Order management, 213
Overhead Cost Controlling, 141

P

P&L statement, 43, 113, 119, 123, 138, 149, 151, 161, 163, 166, 167, 169, 173, 176, 177, 319
Parallel accounting, 150, 151, 250
Parallel ledger, 153, 286
Parallel processing, 67, 75, 87
PayMeNow, 215
Payment Details app, 220
Payment proposal, 215
Percentage of completion, 253
Performance obligation, 251, 252
Persona, 346
Planning, 45, 51, 52, 114, 118, 124, 144, 145, 224, 226, 227, 228, 229, 231, 232, 235, 308
Planning and analysis, 364, 367
Planning Application Kit (PAK), 329
Planning data, 301
Planning models, 367
PO-Flip technology, 214
Point of view, 346
Portfolio Analyzer, 143

Preaggregation, 237
Precalculated data, 107
Predefined aggregates, 108
Predicted line item, 371
Prediction, 34, 118, 256, 261, 372
Prediction journal entries, 165
Predictive payment, 211
Problem space, 351
Process Collections Worklist app, 207
Process control, 243
Process Receivables app, 208
Product Cost Controlling, 141, 237
Profit center, 117, 171, 176, 179, 242
Profit Center Accounting, 101, 148, 149, 179, 318, 329
Profitability, 119, 236
Profitability Analysis, 38, 46, 141, 145, 149, 151, 153, 158, 170, 236, 253, 329
account based, 153, 156, 176, 237, 239, 240, 241, 317
costing based, 149, 153, 176, 236, 237, 239, 241
Project, 153, 159
Prototype, 347, 353
Public Sector, 149

Q

Query language (QL), 191

R

Real time, 31, 34, 106, 109, 119, 126, 131, 232
Real-time data, 185
Real-time simulation, 369
Reconciliation, 27, 31, 84, 106, 107, 108, 136, 147, 148, 156, 159, 239, 243, 245, 317, 320, 325
Redundancy, 81, 82, 83, 84, 85, 86, 89, 91, 92, 94, 157
removal of, 17, 81, 89, 94, 136, 152, 226
Regression, 112
Relational database, 69
Relational model, 82
Replacement, 97, 98
Replication, 293
Replication server, 304
Reporting, 41, 136, 137, 169, 170, 173, 177, 317
Responsive principle, 267
Retroactive analysis, 118
Revenue, 164, 166
Revenue accounting, 250
Revenue recognition, 164, 249, 251, 252
rules, 166
Risk, 28, 35, 121, 124, 131, 303
Robotics, 359, 364
Role-based principle, 267
Row store, 70, 71
Run Simple, 17, 18

S

SAP Ariba, 144, 212, 281
SAP Best Practices for SAP S/4HANA, 285
SAP BPC for Finance, 53, 224, 228, 229, 231, 233, 234
SAP Business Planning and Consolidation (BPC), 329
SAP Business Suite, 39
SAP Business Suite for SAP HANA, 17, 315
SAP Business Warehouse (BW), 283, 301, 336
SAP BusinessObjects, 37
SAP BusinessObjects Analysis for Office, 329
SAP BusinessObjects Dashboards, 317
SAP BusinessObjects Design Studio, 329
SAP Cash Management, 219, 220, 224, 225, 226, 330
SAP Community Network, 265
SAP Customer Engagement Intelligence app, 102
SAP Customer Relationship Management (CRM), 329
SAP Enterprise Performance Management, 37
SAP ERP, 52, 232, 283, 285, 292, 308
SAP ERP Financials, 17, 37, 85, 89, 90, 97, 136, 138, 143, 148, 149, 283
SAP Fieldglass, 281
SAP Financial Supply Chain Management, 142, 143

SAP Fiori, 23, 38, 137, 138, 180, 181, 198, 226, 238, 263, 264, 268, 273, 275, 283, 294, 318, 330, 332
design principles, 266
launchpad, 218, 271, 325
SAP Fiori apps, 191, 271, 315
SAP Fraud Management app, 102, 256, 374
SAP Gateway, 283
SAP GUI, 98, 283
SAP HANA, 17, 38, 40, 42, 48, 60, 65, 71, 81, 83, 94, 97, 99, 107, 127, 135, 138, 149, 151, 152, 154, 157, 173, 185, 198, 218, 227, 231, 233, 236, 245, 253, 255, 260, 265, 281, 299, 301, 304, 324, 369, 374
SAP HANA accelerator, 103
SAP HANA Cloud Platform, 326
SAP HANA Enterprise Cloud, 281, 289, 290, 292, 293
SAP HANA Live, 331
SAP Hybris, 281
SAP Intercompany Reconciliation app, 246
SAP Invoice and Goods Received Reconciliation app, 190, 248
SAP Jam, 210
SAP Landscape Transformation (SLT), 293, 304, 306, 308, 321
SAP Lumira, 55, 317, 329
SAP Master Data Governance, 306
SAP Predictive Analysis Library (PAL), 374
SAP Rapid Deployment solution, 284
SAP Receivables Manager app, 57, 374
SAP Revenue Accounting and Reporting add-on, 145, 249, 250, 252, 253
SAP S/4HANA, 37, 275
flexible analytics, 102
manufacturing, 366
SAP SE, 323
SAP Smart Business, 56, 198, 202, 317
SAP Smart Business cockpit for financial close, 243
SAP solutions for Governance, Risk, and Compliance (GRC), 37, 145, 254
SAP SuccessFactors, 281, 285
SAP Treasury and Risk Management, 142
SAP Working Capital Analytics app, 58
SAPUI5, 266, 275, 337
Sarbanes-Oxley (SOX) Act, 256
Segment, 171, 176, 242
Semantics, 301
Settlement, 241
Shared service center, 113
Side-by-side adoption scenario, 329
Side-by-side extension, 297
Sidecar, 304
Simple principle, 268
Simplicity, 84
Simplification, 17, 18, 81, 91, 117, 135, 137, 163, 225, 232
Simplification List, 287
Simulated line item, 371
Simulation, 359, 370, 372
Single source of truth, 40, 81, 107, 123, 136, 144, 185, 189, 229, 299, 303
Smart device, 360
Smart Invoicing, 214
Soft close, 34, 46
Solution space, 352
Special Ledger, 101, 149
SQL, 65, 107, 192
Stakeholders, 113
Static model, 371
Statutory reporting, 300
Structural change, 130
Subledger, 139
Subscription, 291, 294
System governance, 291

T

Table
BKPF, 90, 101
BSEG, 90, 101
BSID, 101
COEP, 101, 155, 156
Template, 314, 319
Throughput, 59, 85, 93, 94, 157
Time boxing, 354
Total cost of ownership (TCO), 85, 94, 108, 157, 231, 232, 294
Transaction KB11N, 163
Transactional apps, 272, 325
Transactional line items, 369
Transformation, 300, 359
Treasury, 126, 303

U

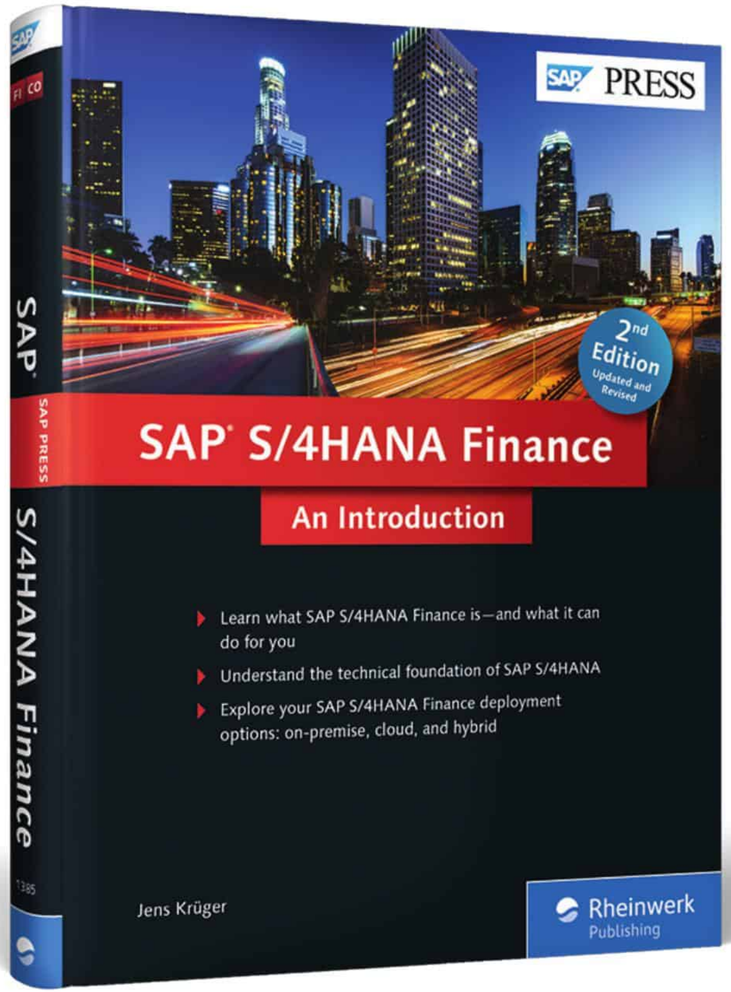
Universal Journal, 22, 46, 83, 89, 92, 101, 119, 135, 136, 137, 147, 151, 152, 153, 158, 159, 162, 166, 167, 169, 173, 176, 179, 229, 243, 245, 317, 330, 337, 359, 366
US Financial Accounting Standards Board (FASB), 249
US Generally Accepted Accounting Principles (GAAP), 139, 153
User experience (UX), 23, 34, 137, 325
User interface (UI), 185, 284
User research, 346, 351

V

Value integrator, 128
Value map, 17, 36, 137
Variance, 241
Verifications, 190
View, 82
Virtual data model (VDM), 41, 102, 188, 193, 198
Visualization patterns, 203

W

What-if scenarios, 115
Work breakdown structure (WBS), 171
Working capital, 110, 124
Work-in-progress calculation, 245
Write operations, 78



Jens Krüger

SAP S/4HANA Finance

An Introduction

411 Pages, 2015. $69.95/€69.95
ISBN 978-1-4932-1385-6

www.sap-press.com/4122



Dr. Jens Krüger Dr. Jens Krüger heads the Finance LoB in the board area of Products & Innovation. Jens oversees a global development organization that is dedicated to developing innovative applications in SAP's core business. Jens and his team were key in developing the first module of SAP S/4HANA—SAP S/4HANA, Finance edition, formerly known as SAP Simple Finance.

In his responsibility as co-head of the SAP Innovation Center Potsdam and managing director of the SAP Labs Berlin, Jens plays a key role in shaping SAP's future Finance portfolio and in further building thought leadership across the SAP ecosystem. Before joining SAP in September 2013, Jens was co-representative of Prof. Hasso Plattner's research chair at the Hasso Plattner Institute for Software Systems Engineering. He was one of the founding members of the prominent joint research project with SAP that built the first prototype of SAP's award-winning in-memory platform, SAP HANA.

We hope you have enjoyed this reading sample. You may recommend or pass it on to others, but only in its entirety, including all pages. This reading sample and all its parts are protected by copyright law. All usage and exploitation rights are reserved by the author and the publisher.

© 2016 by Rheinwerk Publishing Inc. This reading sample may be distributed free of charge. In no way must the file be altered, or individual pages be removed. The use for any commercial purpose other than promoting the book is strictly prohibited.