





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 Simple Finance eliminates redundancy on different levels, and highlights the immediate benefits of the removal of redundant data for customers of SAP Simple Finance.

-  **"Removal of Redundancy"**
-  **Contents**
-  **Index**
-  **The Author**

Jens Krüger

SAP Simple Finance

An Introduction

407 Pages, 2015, \$69.95/€69.95

ISBN 978-1-4932-1215-6

 www.sap-press.com/3793

This chapter covers the benefits of a system without data redundancy, how it is now feasible thanks to SAP HANA, and how SAP Simple Finance eliminates redundancy. We highlight the immediate benefits of the removal of redundant data for customers of SAP Simple 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 Simple Finance. We begin our exploration of the paradigms underlying the SAP Simple 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. Then, in Section 3.3 we explore how SAP Simple 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 Simple 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, 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

But data redundancy has remained. 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 into 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, because it can be derived from the original data at any time by applying the same calculation on the fly.

¹ Edgar Codd. "A Relational Model of Data for Large Shared Data Banks." Communications of the Association for Computing Machinery. 1970.

² Edgar Codd. "Further Normalization of the Data Base Relational Model." IBM Research Report RJ909. 1971.

► **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 Simple Finance removes 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 9 describes a case in which SAP Simple Finance gets rid of 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 in some cases be wanted—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 Simple

Finance preferable compared to the traditional world with 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 an open invoice is cleared by a customer's payment, then it also has to be removed from the corresponding materialized view.

Regardless of whether this reconciliation needs to be done 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 had more than one “truth”—which means that it was 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 listed 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 in the database what is happening by adding corresponding database records. Everything else is then provided 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? And, why wasn't SAP ERP Financials historically an exception either?

In the past, in a traditional disk-based database system the answer 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 subchapter. 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 Simple Finance in Section 3.3.

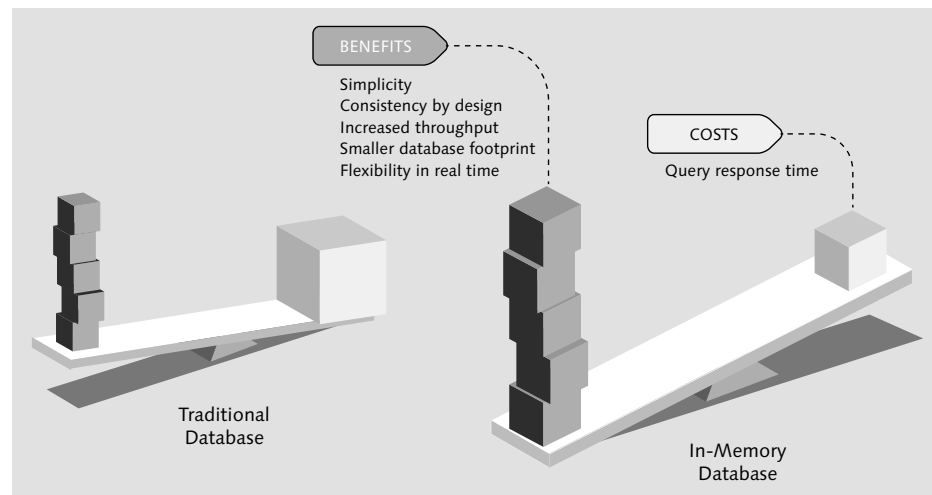


Figure 3.1 Replacing Materialization with On-the-Fly Calculation

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.

But 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 us 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 used 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 (that is, accessing a random location on

disk takes 10 milliseconds or 10 million nanoseconds). In comparison, in-memory latency is only 100 ns (0.1 ms). In other words, the database can make 100,000 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 with a single CPU with eight cores. Modern server systems often connect several CPUs, each with even more than 10 cores, further increasing the processing power. For our example, we assume a company with 100 million accounting document line items in its database and 50,000 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 pre-computed 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. Since 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 the 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 50,000 different numbers, the compressed representation of each customer number only needs two bytes

($2^{16} = 65,536$ 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 on average 2,000 line items per customer, this leads to 2,000 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 4,000 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 12 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 performance in-memory. What is possible with fast and stable access times now 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 Simple Finance optimize the new possibilities to create a redundancy-free system?

3.3 Simplifying the Core Data Model

As outlined above, 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 Simple 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, and others) into separate table structures is a case of duplicate data. Chapter 9 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 now take a closer look into 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.

Let's take a closer look at Financial Accounting. While doing so, we reference the old tables from Financial Accounting for reasons of comparison; with the Universal Journal (see Chapter 9), a next step merges 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.

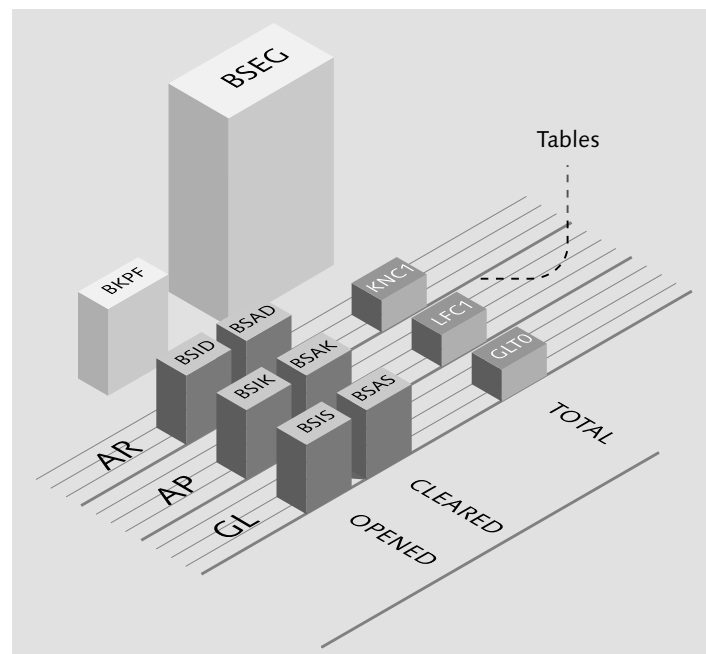


Figure 3.2 Old Data Model of SAP ERP Financials

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 (that is, 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 Simple Finance all of these materializations have been removed in order to take the first step toward eliminating redundancy. Instead, 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 have previously existed in materialized versions.

Figure 3.1 illustrates the resulting redundancy-free data model of Financial Accounting (before the Universal Journal) that is functionally equivalent to the previous data model. 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, since 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.

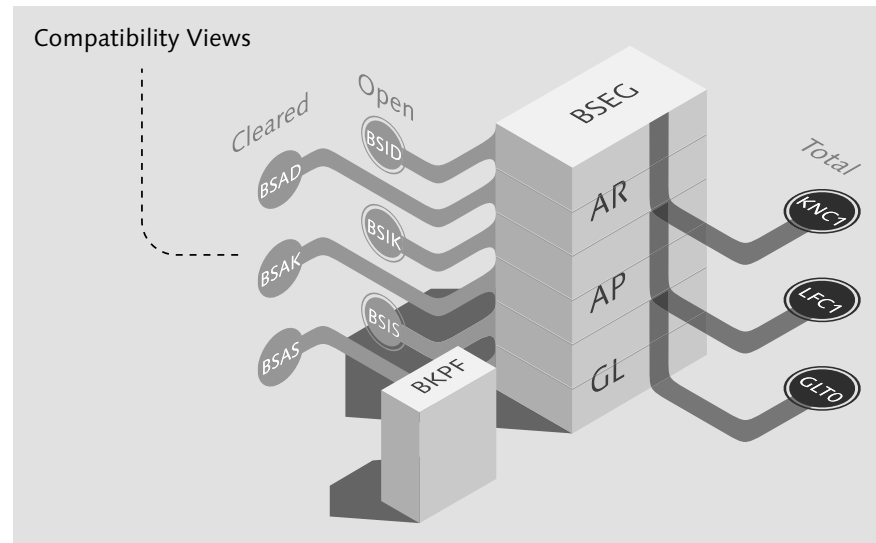


Figure 3.3 New Redundancy-Free Data Model of Financial Accounting

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 either, opening up the possibility to combine different accounting components in the Universal Journal. As outlined in Chapter 12, Section 12.4, cash management is another area with similar changes that remove data redundancy.

In summary, the SAP Simple 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 being 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 Simple 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 Simple 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 26,000 tuples affected by UPDATE, INSERT, and DELETE operations in a traditional SAP ERP Financials system already running on SAP HANA, SAP Simple Finance only needed to insert 11,000 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 Simple Finance only needed 100 ms from end to end, down by a factor of two. As a consequence, the throughput of a system running SAP Simple 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 Simple 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 Simple 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 Financial component, the removal of redundancy in SAP Simple 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; additional savings are possible by applying concepts such as data aging (see Chapter 7) so that in total a reduction by a factor of 14 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 Simple 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 Simple 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 Simple 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 Simple Finance, which allow 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. In this chapter, we

focused on the removal of redundancy in the form of materialized views and materialized aggregates. 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 9 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	31
1.4 Introducing SAP Simple Finance	32
1.4.1 Deployment Modes	37
1.4.2 Transformative Finance Technologies	39
1.4.3 Key Areas of Innovation	45

PART I Exploring the SAP Simple 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	73
2.4 Parallel Execution	74
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	77
2.5.2 Delta Store	77
2.5.3 Delta Merge	78

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 Simple 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	112
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	126
6.3.3	Process Cases	127
6.3.4	Ad Hoc Cases	127
6.4	Becoming a Real-Time Finance Company	127
6.4.1	Balancing Efficiency and Insights	128
6.4.2	Examining Industry Requirements	129
6.4.3	Considering Business Users' Needs	130
7	Data Aging	131
7.1	Information Lifecycle	131
7.2	Traditional Approach: Archiving	133
7.3	Modern Approach: Data Aging	134
7.3.1	Implementation	135
7.3.2	Avoiding Inconsistencies	137

7.3.3	Paged Attributes	137
7.4	Vision	138
PART II Using SAP Simple Finance		
8	Building Blocks of SAP Simple Finance	143
8.1	Seeing the Big Picture	143
8.2	Reloading SAP ERP Financials Functionality	146
8.2.1	Financial Accounting (FI)	146
8.2.2	Management Accounting (CO)	148
8.2.3	Financial Supply Chain Management (FIN-FSCM)	149
8.2.4	Treasury and Risk Management (FIN-FSCM-TRM)	150
8.2.5	Cash Management	151
8.3	New Applications in SAP Simple Finance	151
9	Single Source of Truth: Universal Journal	155
9.1	Mastering Business Challenges without SAP HANA	155
9.2	The New SAP HANA-Based Architecture	159
9.2.1	Structure of the Universal Journal	160
9.2.2	Merging G/L Accounts and Cost Elements	162
9.2.3	Compatibility Views	163
9.3	Immediate Benefits of the New Model	164
9.3.1	Depreciation Runs	165
9.3.2	Multidimensional Profitability Analysis	166
9.3.3	New Analysis Patterns	168
9.3.4	Financial Statement Analysis	169
9.4	Building on Top of the Universal Journal	170
9.4.1	Fundamental Functional Enhancements	170
9.4.2	Important Simplification Cases	171
9.4.3	Prediction Based on Facts	171
10	Flexible Reporting	177
10.1	The Challenge of Flexible Operational Reporting Needs	178
10.2	Addressing the Challenges Using SAP Simple Finance	181
10.3	Conceptual Advances in Reporting	185
10.3.1	Merging of Accounts and Cost Elements	185
10.3.2	Accounts and Other Hierarchies	186
10.4	Advances for End Users	187

11 Flexible Analytics and SAP HANA Live 191

- 11.1 Replacing Complex Analytics with Simplicity 192
- 11.2 SAP HANA Live Paradigms 193
- 11.3 SAP HANA Live Architecture 196
 - 11.3.1 Virtual Data Model 197
 - 11.3.2 Extensible Data Model 199
- 11.4 Unmatched Operational Intelligence 200

12 SAP Simple Finance Applications 203

- 12.1 Business Cockpits for Business People 203
 - 12.1.1 Working with Business Cockpits 205
 - 12.1.2 Defining a Business Cockpit 207
 - 12.1.3 Outlook 209
- 12.2 Financials Operations 210
 - 12.2.1 Receivables Management 211
 - 12.2.2 Payables Management 214
 - 12.2.3 Outlook 216
- 12.3 Ariba Networks Integration 217
 - 12.3.1 Order Collaboration 218
 - 12.3.2 Invoice Collaboration 219
 - 12.3.3 Discount Management 220
 - 12.3.4 Outlook 221
- 12.4 Cash Management 221
 - 12.4.1 Bank Account Management 223
 - 12.4.2 Cash Operations 224
 - 12.4.3 Liquidity Management 228
 - 12.4.4 Simplified Data Model 230
 - 12.4.5 Outlook 231
- 12.5 Integrated Business Planning for Finance 232
 - 12.5.1 Business Context 233
 - 12.5.2 Underlying Architecture 234
 - 12.5.3 Planning Application 237
 - 12.5.4 Business Planning Process 238
 - 12.5.5 Outlook 239
- 12.6 Profitability 241
 - 12.6.1 SAP Fiori Applications 243
 - 12.6.2 From Costing-Based to Account-Based Profitability Analysis 244
 - 12.6.3 Outlook 247

- 12.7 Financial Closing and Consolidation 247
 - 12.7.1 Process Management 248
 - 12.7.2 Avoiding Reconciliation Efforts 249
 - 12.7.3 Faster Period Close with SAP HANA 250
 - 12.7.4 New Closing Applications 251
 - 12.7.5 Outlook 253
- 12.8 Revenue Accounting 254
 - 12.8.1 Contract Identification 255
 - 12.8.2 Performance Obligation Identification 255
 - 12.8.3 Transaction Price Determination 257
 - 12.8.4 Transaction Price Allocation 257
 - 12.8.5 Revenue Recognition 257
 - 12.8.6 Flexible Adaptability 258
 - 12.8.7 Outlook 258
- 12.9 Real-Time Governance and Compliance 259
 - 12.9.1 Introduction to Governance, Risk, and Compliance 259
 - 12.9.2 Changing Trends and Challenges of Staying Compliant 260
 - 12.9.3 Embedded Real-Time Monitoring of High-Risk Transactions 262
 - 12.9.4 Outlook 266

13 Revamping the User Experience with SAP Fiori Apps 267

- 13.1 The Design Philosophy of SAP Fiori 268
- 13.2 Developing Intuitive Products 269
- 13.3 SAP Fiori Design Principles in SAP Simple Finance 270
- 13.4 SAP Fiori Application Framework and Finance 274
 - 13.4.1 Launchpad 275
 - 13.4.2 Application Types 276
 - 13.4.3 Application Patterns and Controls 276
- 13.5 Extending SAP Fiori 278

PART III Running SAP Simple Finance

14 Deploying SAP Simple Finance 285

- 14.1 The Classic On-Premise Deployment 286
 - 14.1.1 Architecture Overview 286
 - 14.1.2 New On-Premise Installations 288
 - 14.1.3 Landscape Migration for Existing SAP ERP Systems 289
 - 14.1.4 Handling General Ledger Functionality during Upgrade 290

14.2	SAP HANA Enterprise Cloud Deployment	290
14.2.1	Service Offerings in the Cloud	292
14.2.2	Hybrid Scenarios	294
14.2.3	Subscription Licensing and Options	295
15	Landscape Consolidation with Central Finance	299
15.1	Harmonization and Standardization	299
15.2	Central Journal Principles and Capabilities	303
15.3	From Central Journal to Central Finance	308
15.4	Central Finance in the Cloud	311
16	Adoption Scenarios	313
16.1	Greenfield vs. Brownfield Approach	314
16.1.1	Greenfield Approach	314
16.1.2	Brownfield Approach	315
16.2	SAP Simple Finance in SAP Business Suite on SAP HANA	316
16.2.1	Manufacturing Company	316
16.2.2	High-Tech Company	318
16.3	Central Finance and Central Journal	318
16.3.1	Trade Company	319
17	Success Story: SAP SE Runs SAP Simple Finance	323
17.1	Defining a Finance Vision	323
17.2	Moving to SAP Simple Finance	326
17.2.1	System Landscape and Organization	326
17.2.2	Finance Transformation	328
17.2.3	Scope and Structure of the Project	329
17.3	Reaping the Benefits	333
17.4	Anticipating the Next Steps	336
17.5	Conclusion	336
18	Partner Ecosystem	339
18.1	Navigating the Partner Model	339
18.2	Partnerships around SAP Simple Finance	341
18.2.1	Service Partners	341
18.2.2	Software Partners	344
18.2.3	SAP HANA Cloud Platform	345

18.2.4	SAP Startup Focus Program	346
18.2.5	Reseller Partners	347
18.2.6	Hardware Partners	347
18.2.7	Additional Partnerships for the SAP HANA Enterprise Cloud	348
18.3	SAP Simple Finance Partnerships in the Future	348
PART IV Moving Forward with SAP Simple Finance		
19	Design Thinking	353
19.1	The Foundation of Design Thinking	354
19.2	Impacts on a Business Process Landscape	358
19.3	Identify and Define Changes	359
19.3.1	Team Setup	360
19.3.2	Problem Space	360
19.3.3	Solution Space	362
19.3.4	Packaging and Handover	363
19.4	Set Up a Design Thinking Workshop	363
19.5	Opportunities in the Business Process Landscape	365
20	The Future of Finance	369
20.1	Simulation and Predictive Analysis	369
20.2	A Proactive Finance Solution	372
Appendices		
377		
A	Changes to the Data Model	379
B	Additional Information	381
B.1	References	381
B.2	Resources	387
C	The Authors	389
Index.....		401

Index

A

ABAP, 99, 163, 286
Accounting document, 90–91, 301
Accounts payable, 45, 147, 211, 215–216, 219, 233
Accounts receivable, 45, 147–148, 211, 213, 216, 233
ACID, 68
Actual data, 132–133, 301
Actual line item, 371
Adjustment, 300
Adoption, 302, 313
Aggregated data, 107, 179
Aggregation, 74, 76
Allocation, 162, 180, 185, 239, 245–246
Analytical apps, 276
Analytics, 32, 144, 191, 194, 335
Analyze Liquidity Plan app, 230
Application management services (AMS), 292, 340
Application patterns, 276
Architecture, 301
Archiving, 133
 rules, 137
Ariba, 152, 217, 285
 Network, 38, 217–218, 220
Ariba Invoice, 219
Assessment service, 294
Asset Accounting, 45, 148, 159, 161, 165, 250, 310
 new, 148
Assignment cycle, 246
ASUG, 269
Atomic, 193

B

Bad debt, 211
Balance sheet, 42, 119, 146, 159, 161, 169, 172, 186, 240

Bank Account Management, 50, 151, 224, 230, 303
Bank statement import status, 225
Batch job, 106, 118, 216, 235, 335, 361
Big data, 292
Bookmarking, 210
Bring-your-own-license (BYOL), 293
Brownfield, 314–315
Budgeting, 118
Business add-in, 306
Business cockpit, 53, 152, 203–204, 207
Business intelligence (BI), 55
Business model, 130, 292, 366
Business network, 152, 217
Business object, 135
Business Planning Framework (BPF), 238, 240
Business process, 358
Business Rule Framework plus (BRFplus), 254, 256
Business transformation, 106

C

Calculation principle, 118
Calibration, 264
Carbon Copy Invoices app, 220
Cash discount, 215, 220
Cash Discount Utilization app, 215
Cash flow, 34, 49, 119, 211, 226, 228–229, 233, 240, 370
Cash Management, 48, 92, 151–152, 216, 221, 303
Cash position, 225
Cash Position app, 227
Central Finance, 23, 103, 123, 250–251, 295, 299, 302–303, 309–311, 315, 318, 320
Central Journal, 47, 295, 302–303, 305–306, 311
Centralization, 222, 310
Centralized finance, 47
CFO, 27, 106, 121, 300, 327

Change management, 358
 Chart of accounts, 178, 305
 CIO, 36
 Classic G/L, 101
 Classification problem, 374
 Closing, 30, 45, 106, 152–153, 156, 167, 178, 185, 247–248, 250, 253, 320, 334, 369
 Cloud, 23, 32, 285, 290, 294, 297, 311
 Cloud deployment, 297
 Code pushdown, 59, 342
 Coherent principle, 272
 Cold data, 134
 Collection worklists, 212
 Collections Management, 150
 Column scan, 74, 87
 Column store, 67, 69–71, 73, 75, 77, 93
 Commenting, 210
 Company code, 183
 Compatibility view, 91, 98–99, 101, 160, 163, 178
 Compatible refactoring, 98
 Complexity, 17
 Compliance, 30, 35, 259–261, 264
 Concur, 285
 Consistency, 40, 82, 84, 201
 Consolidation, 45, 247–248, 251, 253, 310
 Core data model, 89
 Correlation, 111, 114
 Cost center, 117, 161, 166–168, 178–181, 183, 186, 234
 Cost Center Accounting, 178
 Cost element, 159, 161–162, 178, 185–186
secondary, 164
 Cost Element Accounting, 149
 Cost object, 306
 Cost object structures, 307
 Cost of goods sold, 246
 CPU cache, 65, 69, 74
 Credit Risk Analyzer, 150
 Customer journey map, 361–362
 Customizing, 305

D

Dashboard, 207, 218, 263, 275, 371
 Data aging, 131, 134, 136, 139, 333
 Data availability, 191
 Data bucket, 117
 Data centers, 291
 Data collection, 300
 Data compression, 71, 73, 87
 Data dictionary, 99
 Data distribution, 111
 Data duplication, 194
 Data governance, 191
 Data growth, 133
 Data harmonization, 126
 Data latency, 191, 237
 Data privacy, 291
 Data redundancy, 21
 Data replication, 237
 Data security, 291
 Data storage requirements, 133
 Data temperature, 134, 136
 Database footprint, 85, 93–94, 101, 134, 165, 333
 Database migration option (DMO), 289
 Database performance, 132
 Database table, 131
 DataStore object, 135
 Days sales outstanding (DSO), 57, 110, 211, 234, 275
 Decentral system, 301, 305
 Decentralization, 222
 Delightful principle, 274
 Delta merge, 78
 Delta store, 77–79
 Delta version, 159
 Deployment, 23, 32, 285, 287
 Depreciation, 166
 Design process, 269
 Design Thinking, 24, 329, 353–354, 359, 365
in business, 365
process, 355
workshop, 363
 Detective control, 262, 266
 Device agnosticism, 270
 Dictionary, 72
 Dictionary encoding, 72–74, 88
 Dispute Management, 150
 Document splitting, 158, 161, 290, 317
 Double-buffer, 79
 Drill down, 120, 122, 181, 183, 192, 204, 206, 209, 211, 226
 Driver trees, 209

DSAG, 269
 Due diligence, 109, 261, 345
 Duplicate data, 83
 Dynamic models, 371

E

EHP 7, 286, 288, 290, 294, 315
 Exposure Hub, 231, 325
 Extensibility, 159, 170
 Extension, 98

F

Fact sheet apps, 276, 325
 False negatives, 264
 False positives, 264–265
 Fieldglass, 285
 Financial Accounting, 22, 89, 91, 144, 146, 155
 Financial close, 34
 Financial Closing Cockpit, 334
 Financial operations, 150, 152, 210
 Financial planning, 33
 Financial statement, 156, 164, 175, 178, 181, 186, 246, 248, 254
 Flexibility, 18, 85, 94, 108, 144, 192
 Flexible analytics, 22
 Flexible reporting, 22
 Floorplan Manager (FPM), 287
 Forecast, 33, 118, 371
 Fragmented landscape, 105
 Fraud, 335
 Functional area, 179, 183
 Functional Area Ledger, 157

G

General Ledger (G/L), 45, 89, 101, 147, 159, 161, 165, 170, 179, 183, 186, 234, 245, 248, 250, 258, 290, 317, 329, 333
account, 93, 159, 161–162, 164, 167, 181, 185
 Geomaps, 209
 Goodwill, 109

GR/IR, 195, 252
 Greenfield, 285, 288, 314, 318, 362

H

Hard disks, 67
 Hardware partner, 347
 Harmonization, 300, 302, 310, 324
 HEC → SAP HANA Enterprise Cloud
 Hierarchy, 108, 115, 370
 High-performance application, 102
 Historical data, 132–133
 Hot data, 134
 HTML5, 270, 287
 Hybrid deployment, 285, 294
 Hybris, 285

I

Ideate, 356, 362
 IDEO, 354
 Implementation, 23
 Indirect partner, 347
 Individualization, 286
 Industry requirements, 129
 Information loss, 108, 110
 Information silo, 105
 In-memory computing, 31, 67, 85, 89, 93, 143, 266, 301, 324
 In-memory database, 20, 66, 69, 289
 Input/output (I/O) access, 65
 Insert, 77
 Intercompany reconciliation, 60, 303, 317
 Internal order, 161
 International Accounting Standards Board (IASB), 254
 International Financial Reporting Standards (IFRS), 147, 156, 161, 254–255, 257
 Invoice clearing, 196
 Invoice collaboration, 219

J

Join, 197
 Joint Venture Accounting, 156–157

K

Key figure, 244
 Key performance indicator (KPI), 49, 110, 122, 124, 204, 207–208, 211–212, 223, 271, 370, 372

L

Ledger, 161–162, 170
 Licensing, 295–296
 Line item, 21, 42, 87, 90–91, 120, 144, 301
 Liquidity, 123

M

Main memory, 65, 68–69
 Management Accounting, 22, 45, 144, 148, 155, 159, 161, 234, 248, 250
 Margin, 122, 207
 Margin Analysis app, 243
 Market Risk Analyzer, 150
 Market segment, 161, 167, 169, 172, 234
 Material Ledger (ML), 161, 165
 Materialization, 91, 93
 Materialized aggregate, 82, 85–90, 92, 94, 98–99, 157, 160, 333
 Materialized result set, 83
 Materialized view, 82, 84, 89–92, 98, 160
 Memory consumption, 139
 Memory space, 133
 Mergers, 109, 300
 Migration, 289, 297, 315, 329, 333
 Migration services, 294
 Mobile, 271
 Mobile technology, 32
 Modifications, 293
 Multidevice experience, 272
 Multidisciplinary team, 360
 My Spend app, 187, 206, 279

N

Navigation, 271, 275
 Net margin, 205, 243

Net Margin Results app, 243
 New G/L (see also General Ledger), 147–148, 158
 Nondisruptive, 17, 21, 89, 91, 97–98, 103, 136, 144, 160, 292, 330, 341
 Normalization, 82

O

OLAP, 40, 83, 94, 107, 165, 192, 231, 287, 301
 OLTP, 40, 83, 107, 165, 192, 300–302
 Onboarding services, 294
 One price, 296
 One-system strategy, 300
 On-premise deployment, 23, 37, 285–286, 293–294, 311, 341
 On-the-fly calculation, 45, 83, 88, 92
 Operational contract, 255
 Operational data provider (ODP), 287
 Operational efficiency, 211
 Optimization problem, 374
 Order management, 218
 Outlier, 111
 Overhead Cost Controlling, 149

P

P&L statement, 42, 113, 119, 122, 146, 157, 159, 169, 171, 174–175, 177, 181, 183, 185, 318
 Paged attributes, 138
 Parallel accounting, 158–159, 254
 Parallel ledger, 161, 290
 Parallel processing, 66, 74, 87
 Partitioning, 134–136
 Partner, 24, 340, 345, 347
 Partner ecosystem, 339, 345
 PayMeNow, 220
 Payment Details app, 226
 Payment proposal, 220
 Percentage of completion, 258
 Performance obligation, 255, 257
 Persona, 356, 361
 Planning, 44, 50–51, 113, 118, 120, 124, 152, 229, 231–234, 236–237, 240, 308

Planning Application Kit (PAK), 329
 Planning data, 301
 Platform-as-a-service (PaaS), 345
 PO-Flip technology, 219
 Point of view, 356
 Portfolio Analyzer, 150
 Preaggregation, 242
 Precalculated data, 107
 Predefined aggregates, 108
 Predicted line item, 371
 Prediction, 33, 118, 260, 265, 369–370, 372
 Prediction journal entries, 173
 Predictive Analysis Library (PAL), 375
 Predictive payment, 216
 Private cloud, 38
 Private view, 197–198
 Problem space, 360
 Process Collections Worklist app, 212
 Process control, 248
 Process Receivables app, 213
 Product Cost Controlling, 149, 242
 Profit Analysis app, 243
 Profit center, 117, 179, 183, 186, 248
 Profit Center Accounting, 101, 156–157, 186, 317, 328
 Profitability, 119, 241
 Profitability Analysis, 37, 45, 149, 153, 157, 159, 161, 166, 178, 241, 258, 328
 account based, 161, 164, 184, 242, 244–246, 316
 costing based, 157–158, 161, 184, 241–242, 244, 246
 Project, 161, 167
 Prototype, 356, 362
 Public cloud, 38
 Public Sector, 157

Q

Query view, 197, 199

R

Ramp up, 342
 Read-only data, 132

Real time, 21, 30, 33, 106, 109, 119, 125, 130, 237
 Real-time data, 191
 Real-time simulation, 370
 Reconciliation, 27, 30, 84, 106–108, 144, 155–156, 164, 167, 245, 248, 250, 317, 319, 324
 Redundancy, 81–86, 89, 91–92, 94, 165, 197
 removal of, 17, 81, 89, 94, 144, 160, 231
 Regression, 111
 Relational database, 69
 Relational model, 82
 Replacement, 97
 Replication, 295
 Reporting, 40, 144–145, 177–178, 181, 185, 194, 316
 Reseller partner, 347
 Responsive principle, 271
 Retroactive analysis, 118
 Reuse view, 197–198
 Revenue, 172, 174
 Revenue accounting, 254
 Revenue recognition, 172, 254, 256–257
 rules, 174
 Risk, 28, 34, 121, 124, 130, 303
 Role-based principle, 270
 Row store, 70
 Run Simple, 17–18, 348

S

SAP Business Planning and Consolidation (BPC), 328
 SAP Business Suite, 38
 SAP Business Suite 4 SAP HANA, 17, 37
 SAP Business Suite on SAP HANA, 315, 343
 SAP Business Warehouse (BW), 135, 287, 301, 334
 SAP BusinessObjects, 36
 SAP BusinessObjects Analysis for Office, 329
 SAP BusinessObjects BI, 344
 SAP BusinessObjects Dashboards, 199, 287, 317
 SAP BusinessObjects Design Studio, 199, 329
 SAP Cash Management, 224, 228–229, 231, 330

SAP Community Network, 269, 348
 SAP Crystal Reports, 199
 SAP Customer Engagement Intelligence app, 102
 SAP Customer Relationship Management (CRM), 329
 SAP Enterprise Performance Management, 36
 SAP ERP, 51, 237, 286, 289, 294, 307
 SAP ERP Financials, 17, 36, 85, 89–90, 93, 97, 144, 146, 151, 156–157, 286
 SAP Financial Supply Chain Management, 149–150
 SAP Fiori, 23, 37, 145–146, 187–188, 204, 231, 243, 267–268, 272, 276, 279, 287, 295, 318, 330–331, 342
design principles, 270
launchpad, 223, 274, 325
 SAP Fiori apps, 275, 315
 SAP Fraud Management, 260
 SAP Fraud Management app, 102, 374
 SAP Gateway, 287
 SAP Governance, Risk, and Compliance (GRC), 36, 153, 259
 SAP GUI, 98, 287
 SAP HANA, 17, 21, 37, 40, 42, 48, 59, 65, 71, 81, 83, 94, 97, 99, 107, 126, 134, 139, 143, 146, 157, 159–160, 162, 165, 181, 191, 194, 204, 223, 232, 236, 238, 241, 248, 250, 258, 260, 264, 268, 285, 299, 301, 304, 324, 342, 347, 375
 SAP HANA Accelerator, 103
 SAP HANA Cloud Platform, 326, 344–345
 SAP HANA Enterprise Cloud, 37, 285, 290, 292–293, 295, 340–341
 SAP HANA Live, 40, 102, 192–193, 196, 198–200, 204, 208, 287–288, 324, 331
 SAP Integrated Business Planning, 52, 229, 233–234, 236, 238–239
 SAP Intercompany Reconciliation app, 248, 251
 SAP Invoice and Goods Received Reconciliation app, 195, 253
 SAP Jam, 215
 SAP Landscape Transformation (SLT), 303, 305–307, 321
 SAP Lumira, 55, 199, 317, 329
 SAP Master Data Governance, 295, 305–306
 SAP Mobile Platform, 344
 SAP PartnerEdge, 339, 342, 344
 SAP PartnerEdge Portal, 342
 SAP Rapid Deployment solution, 288
 SAP Receivables Manager app, 57, 374
 SAP Revenue Accounting and Reporting add-on, 153, 254, 257–258
 SAP SE, 24, 323, 339
 SAP Service Marketplace, 343
 SAP Simple Finance apps, 23, 279
 SAP Smart Business, 56, 204, 207, 317, 342
 SAP Smart Business cockpit for financial close, 249
 SAP Startup Focus, 346
 SAP Treasury and Risk Management, 150
 SAP Working Capital Analytics app, 57
 SAPUI5, 269, 279, 335, 342
 Sarbanes-Oxley (SOX) Act, 261
 Segment, 179, 183, 248
 Semantic data model, 193
 Semantics, 300
 Service partner, 341–342
 Settlement, 246
 Shared service center, 113
 Side by side, 328
 Sidecar, 304
 Simple principle, 272
 Simplicity, 84
 Simplification, 17–18, 81, 91, 117, 143, 145, 171, 230, 237
 Simulated line item, 371
 Simulation, 370, 372
 Single source of truth, 40, 81, 107, 118, 123, 144, 151, 191, 194, 234, 299, 303
 Smart Invoicing, 219
 Smartphone Pack, 257
 Soft close, 33, 45
 Software partner, 344
 Software Update Manager, 289
 Software-as-a-service (SaaS), 37, 345
 Solution space, 362
 Special ledger, 101
 Special Purpose Ledger, 157
 SQL, 65, 107
 Stakeholders, 113
 Static model, 371
 Statutory reporting, 300
 Structural change, 129
 Subledger, 147

Subscription, 293, 295
 SuccessFactors, 285
 System governance, 293
 System landscape transformation, 295

T

Table
BKPF, 90, 101, 135
BSEG, 90, 101, 135
BSID, 101
COEP, 101, 163
 Template, 314, 318
 Throughput, 58, 84, 92, 94, 165
 Time boxing, 364
 Total cost of ownership (TCO), 38, 85, 94, 108, 165, 236–237, 295
 Transaction KB11N, 171
 Transactional apps, 276, 325
 Transformations, 300
 Treasury, 125, 303

U

Union, 197
 Universal Journal, 22, 45, 83, 89, 92, 101, 119, 143–145, 155, 159–161, 163, 166–167, 170, 173–174, 177, 181–183, 187, 234, 248, 250, 316, 330, 336

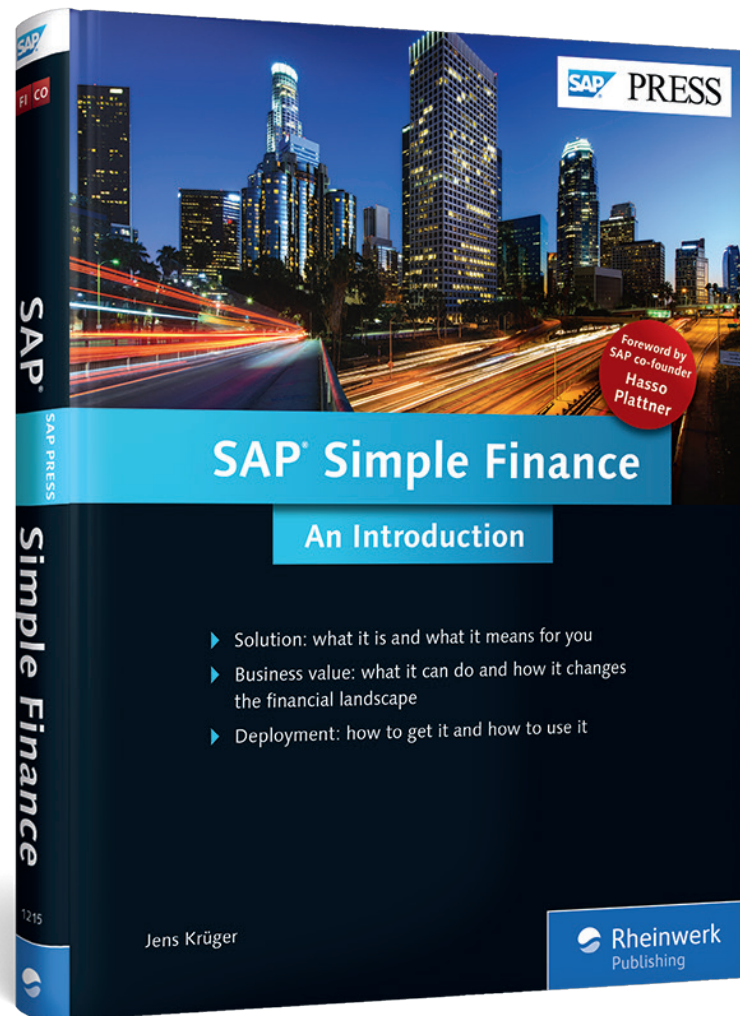
Upgrade Right, 256
 US Financial Accounting Standards Board (FASB), 254
 US Generally Accepted Accounting Principles (GAAP), 147, 161
 User experience, 23, 33, 145, 325
 User interface (UI), 191, 288
 User research, 356, 361

V

Value integrator, 128
 Value map, 17, 36, 145
 Variance, 246
 Verifications, 196
 View, 82
 Virtual data model (VDM), 40, 102, 194, 196, 199, 204
 Visualization patterns, 208

W

Web Dynpro, 287
 What-if scenarios, 115
 Work breakdown structure (WBS), 179
 Working capital, 110, 124
 Work-in-progress calculation, 250
 Write operations, 77



Jens Krüger

SAP Simple Finance

An Introduction

407 Pages, 2015, \$69.95/€69.95

ISBN 978-1-4932-1215-6

 www.sap-press.com/3793



Dr. Jens Krüger heads the LoB Finance and SAP Innovation Center unit in the board area of Products & Innovation, where he reports to Bernd Leukert, a member of the SAP Executive Board. Jens oversees a global development organization that is dedicated to delivering SAP Simple Finance in the cloud powered by SAP HANA, in close collaboration with customers and partners. Prior to joining SAP in September 2013, Jens was co-representative of Dr. 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, which 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.

© 2015 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.