Bender, Benedict; Bertheau, Clementine; Körppen, Tim; Lauppe, Hannah; Gronau, Norbert

A proposal for future data organization in enterprise systems—an analysis of established database approaches

Information Systems and e-Business Management

Provided in Cooperation with:
Springer Nature

Suggested Citation: Bender, Benedict; Bertheau, Clementine; Körppen, Tim; Lauppe, Hannah; Gronau, Norbert (2022) : A proposal for future data organization in enterprise systems—an analysis of established database approaches, Information Systems and e-Business Management, ISSN 1617-9854, Springer, Berlin, Heidelberg, Vol. 20, Iss. 3, pp. 441-494, https://doi.org/10.1007/s10257-022-00555-6

This Version is available at:
https://hdl.handle.net/10419/313227

Information Systems and e-Business Management (2022) 20:441–494
https://doi.org/10.1007/s10257-022-00555-6

ORIGINAL ARTICLE

A proposal for future data organization in enterprise systems—an analysis of established database approaches

Benedict Bender[1] · Clementine Bertheau[1] · Tim Körppen[1] · Hannah Lauppe[1] · Norbert Gronau[1]

Received: 5 August 2021 / Revised: 15 January 2022 / Accepted: 23 March 2022 /
Published online: 4 May 2022

Abstract

The digital transformation sets new requirements to all classes of enterprise systems in companies. ERP systems in particular, which represent the dominant class of enterprise systems, are struggling to meet the new requirements at all levels of the architecture. Therefore, there is an urgent need to reconsider the overall architecture of the systems and address the root of the related issues. Given that many restrictions ERP pose on their adaptability are related to the standardization of data, the database layer of ERP systems is addressed. Since database serve as the foundation for data storage and retrieval, they limit the flexibility of enterprise systems and the chance to adapt to new requirements accordingly. So far, relational databases are widely used. Using a systematic literature approach, recent requirements for ERP systems were identified. Prominent database approaches were assessed against the 23 requirements identified. The results reveal the strengths and weaknesses of recent database approaches. To this end, the results highlight the demand to combine multiple database approaches to fulfill recent business requirements. From a conceptual point of view, this paper supports the idea of federated databases which are interoperable to fulfill future requirements and support business operation. This research forms the basis for renewal of the current generation of ERP systems and proposes to ERP vendors to use different database concepts in the future.

Keywords Database · Enterprise system · ERP system · Requirements · Problems · Future

✉ Benedict Bender
benedict.bender@wi.uni-potsdam.de

Extended author information available on the last page of the article

1 Introduction

The digital transformation as well as the further development of existing products and services set new requirements to the dominating class of enterprise systems in companies, the Enterprise Resource Planning (ERP) systems. Requirements stem from new needs of organizations and markets as well as technological advancements. Previous studies address the need to adapt enterprise systems to newer demands of businesses. Thereby, previous research focused on the lifecycle on enterprise systems rather than their architectural setup. Bender et al. (2021) identified problems with the current generation of ERP systems that occur during ERP selection, implementation, and usage and can be attributed to the system's design and architecture. The authors call for the consideration of the architecture to address the root of related issues. To remain its central role, enterprise systems need to appropriately cover upcoming demands.

The survey indicates the importance of the different architectural layer for the next generation of ERP systems. Thereby, most problems named occur around databases and data models. This is primarily about the use of relational databases because they are the databases of choice for most ERP vendors today. As such, the question in how far existing database types are suited to fulfill future business requirements as a foundation for enterprise systems.

The inflexibility of databases and data models becomes apparent for instance when new businesses are to be integrated in central ERP instances. The integration of a new data model into the existing ERP database is very costly and leads to additional maintenance efforts every time the version of the ERP system is upgraded. A related problem is the inability of an existing ERP system to cope with innovation. This could be a new product, a new process or something else. It is complicated and costly to add elements to the data model and nearly impossible to remove the additions later when the innovation has proven unsuccessful. Moreover, the integration of different products into a unique data model is impossible but reality in multinational and diversified companies. When one product is a food product stemming from a recipe and the other is a machine originating from a bill of materials. The application landscapes are complex and heterogeneous. They continue to grow because we have larger value creation networks (inter- and intraorganizational) due to increasing digitalization. This is accompanied with more and more data, from which value should be created and with which decisions can be made in a more profound way. This requires approaches for ERP systems to flexibly merge and migrate data from different systems like orders in the supply chain, customer data with a CRM system or manufacturing orders with a manufacturing execution system (MES).

The problems that arise at the database level due to new requirements of digital transformation give reason to assume that the relational concept may be outdated. To explore if and which database characteristics constitute a bottleneck to cope with future business requirements, this paper aims to evaluate different database types to determine whether they meet the future requirements for databases in ERP systems. Hence, the research questions are:

R1: What new requirements are emerging from the digital transformation for enterprise systems regarding databases?
R2: Which database types are suitable to meet future business requirements for ERP systems?

By identifying the suitability of database types as the foundation of enterprise systems, this paper contributes to the future design of enterprise system architecture with a focus on the database layer in specific. The results are important for enterprise systems vendors that rely on database products of specialized vendors. For them, the selection of databases as foundation for their enterprise systems is highly critical to provide a flexible and scalable systems and to meet future requirements of their customers. The paper contributes through guidance for renewal of the data base layer in ERP systems.

The paper is structured as follows. The second section presents related literature on prevalent database types. Section three describes the research methodology, whereby the selection and evaluation procedure are explained. Section four presents the results from the evaluation of the databases. The results are discussed and managerial implications, as well as the theoretical contribution and limitations, are introduced in section five. Finally, the conclusion that ties the manuscript together is drawn in section six.

2 Related literature

2.1 Databases

The database approach describes the idea in which way the data is organized within the database. As such the database approaches provides an abstraction of data by hiding internal details on data storage. The data model is used to describe the structure of the database approach and thereby provides an abstraction of its physical representation. The data types within a database also define the available operations and methods within the respective database (Elmasri and Navathe 2007).

For the realization of databases, different levels of data models exist. Typically, three levels of data models are distinguished (Elmasri and Navathe 2007). First, conceptual data models describe the principle approach of data organization. In contrast, physical data models specify how data is stored. In between those two, the representational data models provide the basic principles of data organization approaches. To distinguish different types of data models that are potentially suitable for the enterprise systems context, we use conceptual data models.

In the following, six conceptual data models are described. The databases were selected for the purpose of this paper because they are either widely used and established in practice for purposes other than ERP and or have gained certain popularity in the recent past (Elmasri and Navathe 2007). The research group considers the following databases to be eligible for ERP systems, starting with relational databases that are primarily used for ERP systems today.

2.1.1 Relational database

The relational database implements the relational data model which is typically modeled using an entity relationship model (Thalheim 2000). The foundation for the relation databases concept is a relation. A relation is the mathematical description of a table. The relational algebra defines the different operations that can be executed upon the data table (Elmasri and Navathe 2007). Typically, the data in relational databases is represented in the form of multiple tables that are set in relation to each other. The properties of individual tables and relations between the tables are predefined by the schema of the relational database. Database systems that implement relational databases are referred to as Relational Database Management Systems (RDBMS). The relational database approach was invented in the 1970s. Relational database systems are typically queried using the Structured Query Language (SQL). In the late 2000s, a new approach towards more scalable, relational databases has emerged. Those so called NewSQL Databases follow a relational structure and semantics while offering greater scalability and consistency (than relational or NoSQL Databases). The main contributor for this is the partitioning of data into smaller ranges ("shards"). Additionally, this enables distributed architectures which have significantly gained in relevance in times of BigData and cloud-computing.

2.1.2 Column-based database

Column oriented database management systems (CDBMS) store data by columns rather than by rows (as typically relational databases do). As such, column-oriented databases are different in data storage and physical representation of data which results in different possibilities concerning requests on the data (Elmasri and Navathe 2007). While a row-oriented databases are better suited for online transaction processing (OLTP) systems, column-oriented databases are well-suited for analytical task such as online analytical processing (OLAP) systems (Abadi et al. 2013). For reporting, typically not all attributes of a row are needed, therefore the access to a single digit number of columns is much faster than the traditional scanning of rows for these attributes (Abadi et al. 2009). The gain in speed stems from faster reading, because only necessary columns are read, because of more efficient compression algorithms due to a lower degree of entropy in the data of a column and the usage of sorted columns and techniques like "late materialization" which tries to project columns as late as possible. Column-oriented databases are mainly used in data warehouses, for data mining and for semantic web data management.

The column story approach is favorable where across many rows only a few columns have to be read and evaluated, for instance for reporting purposes. Similar to row-based databases, Column-based databases can be queried using SQL.

2.1.3 Object-relational database

In the 1990s the database generation of object-relational database systems (ORDBMS) came onto the market as an extension of traditional relational database systems by object-oriented concepts (Lufter 1999). Object-oriented databases are used

for storing information on objects with complex structures (Atkinson et al. 1990). Object-oriented databases allow to handle special requirements and to cover individual characteristics of objects within software applications. They integrate structural specifications and the operation that can be applied to those objects, whereby different objects typically vary in their possibilities. Objects, classes, and inheritance are directly represented in the database schema and language. This class of databases is designed to or can be directly derived from object-oriented programming (Elmasri and Navathe 2007). There is no need to model related data structures separately. The approach combines the specifications during application conceptualization and programming. The application development and database specifications are integrated by that approach. Similar to relational databases, data is stored in tables and the result of a (SQL) query is also a table (Mahnke and Steiert 2000).

2.1.4 Graph database

Graph databases belong to the family of "not only" SQL databases (NoSQL) (Benymol and Abraham 2020). The main difference to traditional concepts is the ability to avoid a rigid scheme. Therefore various (modern) database concepts can be assigned to the NoSQL family. This holds true for graph databases as well. In general, graph databases are based on graph-theoretical concepts, i.e. the representation of content and relationship of data records in nodes and edges (and properties, depending on the implementation). Therefore, object-oriented applications would fit more natural to graph databases due to more a more flexible schema (Chen et al. 2020). There is a large variety of graph database models which can be differentiated via integrity constrains, manipulation options and requirements of the data that are directly addressed by the model (Angles and Gutierrez 2008). The most common representation is the labeled property graph (LPG). It is made up of nodes which contain the values/properties and can be tagged with labels (e.g. for grouping) as well as directional, labeled edges which represent relationships between nodes (Robinson et al. 2015). Those labels for the nodes and edges represent meta data for graph algorithms, improved readability and semantics.

Traditional applications of graph databases include social networks e.g. Facebook with 1 Billion nodes and 140 Billion edges, semantic Web representations (Pokorný 2015), geographic applications or bioinformatics (De Virgilio et al. 2014). Graphs allows mapping of all kinds of real-world objects-oriented applications. It is therefore not surprising that application scenarios are also being researched in the field of business information systems (Rudolf et al. 2013). Graph databases are becoming mainstream (Pokorný 2015).

2.1.5 Key-value database

Similar to network and graph databases, key-value databases are related to the NoSQL family. In general, key-value databases map sets of keys (i.e. an individual and unique identifier) to corresponding values (i.e. the actual content of the data record). Besides this mapping, there is no structure or relationship represented in the organization of the data itself. Therefore, key-value stores can be referred to as

schemaless (Sadalage and Fowler 2013). Manipulation of the data is limited to simple CRUD (create, read, update, delete) operations. These constrains in organization and manipulation are at the same time the main advantages of key-value databases: the simplicity results in low latency and high throughput (Gessert et al. 2017).

2.1.6 Document-based database

Pure document stores are an enhancement of key-value store and therefore follow NoSQL characteristics. Every data record (value) corresponds to a unique key by which querying and manipulation is performed. The main difference is that the values are restricted to semi-structured formats (e.g. JavaScript Object Notation or Extensible Markup Language). When retrieving data from document-based databases it is possible select entire documents or specify parts of the content of documents (Gessert et al. 2017). Besides that, the flexibility of the document structure allows modification of attributes (e.g. adding or removing attributes) at runtime (Davoudian et al. 2018).

2.2 Database-requirements of future ERP systems

In software engineering and design of information systems, the assessment of relevant requirements is of primary importance. It assures precise development and later the suitability for real-world application. For enterprise systems, especially ERP systems, those requirements can affect all layers and components of the systems architecture (Gronau 2021). Because the main tasks of ERP systems are the management of resources and business objects, requirements for databases of ERP systems are of major relevancy for process execution and (data and process) consistency.

Evolving economic and business requirements demand for continuous development of the supporting enterprise systems, i.e., ERP systems and corresponding databases. Relevant requirements for the database level were identified in literature. To structure the requirements identified the classification of requirements for software intensive systems in the twenty-first century of Hansen et al. (2009) was used. In the following the different categories and their importance for ERP systems are highlighted.

2.2.1 Business process focus

The theme of business process focus describes requirements that are directly derived from business processes and their technological artifacts. For (ERP) databases, transformation of processes entails consequences in data management, especially scalability with regard to data loads. For example, in business models with predominant customer orientation or the field of (industrial) internet of things, efficient integration and transformation of large amounts of data is required while still being able to scale in relation to current data / workloads (ur Rehman et al. 2019).

2.2.2 Integration focus

Requirements for the theme of integration focus address the integration of existing applications. This includes both, the requirement of systems interoperability, i.e., adaptability to different operation types of the databases (database systems), standards and interfaces as well as interoperability of the database schemes.

2.2.3 Distributed requirements

Distributed requirements include both, requirements of diverse stakeholders as well as those that are distributed because of organizational and geographical distances in business operations. In terms of (ERP) databases, requirements of distribution are relevant for cooperation, safety and privacy purposes.

2.2.4 Layers of requirements

For layers of requirements, different levels of abstractions are addressed by the requirement. With increased complexity and size of data models, support of human readability is necessary for different layers of the database as well as its content and structural representation.

2.2.5 Centrality of architecture

Requirements that focus on architectural circumstances and thereby indirectly influence the application are collected in the theme of centrality of architecture. Whereas modular software is state-of-the-art, databases and -models mostly still don't realize the advantages (esp. higher flexibility, reliability, availability, scalability and lower cost) of modularity. Various approaches towards modular database architectures and distributed databases exist (Irmert et al. 2008; Parent et al. 2009; Seybold and Domaschka 2017), but these are hardly ever found in the practical application in enterprise systems.

2.2.6 Interdependent complexity

In general, complexity of software systems has risen significantly. Therefore, requirements that address complexity of requirements (especially interdependencies) are subsidized in the theme of interdependent complexity. Both, detailed and individualized representations of business objects and processes raise complexity.

2.2.7 Fluidity of design

So far, requirements addressed the design of the database. In contrast, the theme of fluidity of design addresses requirements concerning the continued evolution of the database, i.e., post-implementation. This is of special importance as ERP systems

are often operated for a prolonged time. For databases, requirements for changes of the database system as well as the (structural) representation of the data model may arise during operation.

2.2.8 System transparency and packaged software

The key themes of system transparency (i.e., requirements for cross-application usability) and packaged software (i.e., requirement concerning commercial off-the-shelf software) are not directly addressed by the database requirements as those are directed primarily towards higher-level system components.

Those priorly presented (ERP) database requirements are confronted with currently existent and emerging database technologies with respect to (expected) fulfillment of those requirements.

2.3 Literature review requirements

The following research is based on a systematic literature review (Paré et al. 2015). It aims to uncover new requirements for information systems that are emerging in enterprise systems relevant to data management. The following overview provides a summary of the literature research procedure. The requirements are structured according to the requirements framework used (see 2.2.)

The authors divided the research question from section one into two equivalent term fields, which are linked independently of one another, and then with one another. As a result, a so-called term matrix creates subject blocks and search terms according to a scheme (Paré et al. 2015) as illustrated in Fig. 1. The first building block guaranteed the thematic fit of articles on enterprise systems by selecting the keywords "Enterprise System", "Information System", "Enterprise Ressource Planning", "ERP" and "Information technology". The keywords of the second block added the new requirement angle due to digital transformation required for this paper. Since new requirements can also be derived from challenges or trends, the following keywords were added: "Requirement", "Challenge", "Trend" and "Transformation". A combined search string was then used to search publication by

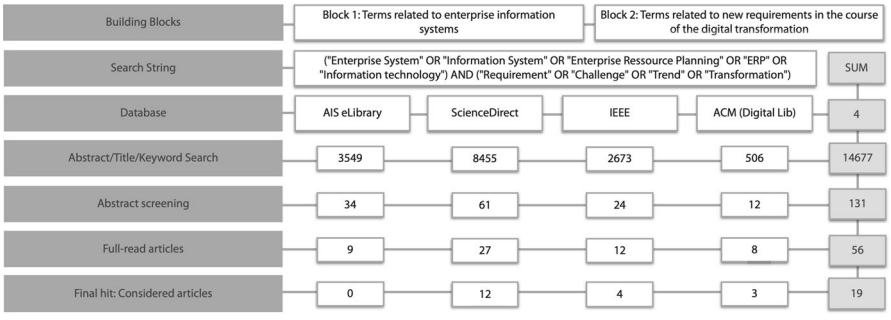


Fig. 1 Systematic Literature Review Process for Requirements Identification

title, keywords and abstract in AISeLibrary, Science Direct, IEEE and ACM Digital Library. 14,677 English-language articles were identified when the query was first submitted. To improve the quality and rigour of the search, the types of articles reviewed were limited to those published in peer-reviewed academic journals between 2010 and 2021 and articles that had a different subject focus other than information systems or technology were excluded. The next step was to read the titles and the abstracts of the identified articles, checking their relevance to the research question. Finally, the full text of 56 paper was assessed for relevance by applying inclusion and exclusion criteria to the full content. The search only considered articles that indicated a link between information systems and new requirements or challenges in the course of digital transformation, or where underlying data management criteria were identified. The search was complemented through a backward/forward approach–following Webster & Watson (2002) recommendations. After this step, a final corpus of 19 articles was selected. Appendix 1 presents the full reference list comprising the 19 selected articles, providing an overview of the key contributions and insights of all selected articles. This review analyzed all 19 studies using an explorative coding process, which was repeated iteratively to develop conclusive coding constructs (Saldaña 2021). More specifically, 437 text phrases have been extracted from the literature and iteratively coded into 23 requirements for future enterprise system databases. For reasons of clarity, all identified requirements were aggregated and mapped along Hansen et al. (2009) identified key themes and issues associated with system design requirements in the twenty-first century. Table 1 provides an overview of the category requirements with relevant findings and an adoption for the ERP context.

2.4 ERP database requirements

This research identified 23 requirements related to data management for information systems in literature and linked them to 6 underlying concepts to answer the first research question. Table 2 provides an overview of the results based on Webster and Watson (2002) and Hansen et al. (2009).

2.4.1 Business process focus

The theme of *business process focus* describes requirements that are directly derived from business processes and their technological artifacts. In terms of organizational context, (Cao and Zhu 2013) found that the increasingly rapid growth of data led to a mismatch between data volume and manual data processing, while business requirements in terms of data management are becoming increasingly complex. Therefore, **database scalability** is a future challenge for modern enterprise systems due to both the growing volume of data (Romero and Vernadat 2016a), the distribution and autonomy of data sources of different data sources (Liu et al. 2014), and the increasing complexity and change in global and multiple instantiated enterprise contexts (Koh et al. 2011). Often, complex interactions and couplings between different business objects, operators,

Table 1 Requirements and operationalization

Category	Former Research	Adoption for ERP
2.2.1 Business process focus	Requirements process focusing on the business process, and requirements for technological artifact driven by business process	Requirements that are directly derived from business processes and their technological artifacts
2.2.2 Integration focus	Requirements efforts focus on integrating existing applications rather than development of new ones	Requirements that address the integration of existing applications
2.2.3 Distributed requirements	In addition to diverse stakeholders, requirements process distributed across organizations, geographically, and globally	Requirements that include both, requirements of diverse stakeholders as well as those that are distributed because of organizational and geographical distances in business operations
2.2.4 Layers of requirements	Requirements iteratively developing across multiple levels of abstraction, design focus, or timing	Requirements that address different levels of abstractions are addressed
2.2.5 Centrality of architecture	Architectural requirements take a central role, and drive product and application requirements	Requirements that focus on architectural circumstances and thereby indirectly influence the application are collected in the theme of centrality of architecture
2.2.6 Interdependent complexity	While some forms of complexity have been reduced, overall complexity has risen significantly	Requirements that address the complexity of software systems has risen significantly
2.2.7 Fluidity of design	Requirements process accommodates the continued evolution of the artifact after implementation	Requirements that address the design of the database
2.2.8 System transparency and packaged software	Requirements driven by demand for a seamless user experience across applications and Purchase of commercial off-the-shelf (COTS) software rather than development – trend toward vendor-led requirements	Requirements are not directly addressed at the database level as those are directed primarily towards higher-level system components

Table 2 Matrix—database requirements linked to systems requirements by Hansen et al. (2009)

Main Concept	Requirement	# of paper
Business process focus	Scalability	4
	Performance (OLAP)	1
	Performance (OLTP)	2
	Performance (OLTP) multiple datasets	3
	Real-time capability	6
	Digital twin	1
	Business objects heterogeneity	3
Integration focus aspect	Extract step (ETL)	2
	Transform step (ETL)	2
	Schema interoperability	7
Distributed requirement aspects	Data protection in data model	3
	Compliance in data model	3
	Cooperation suitability in data model	2
	Data security in data model	2
Layers of requirement aspect	Granular security settings	2
	Comprehensibility of data model	2
Centrality of architecture	Modularity	1
	Process structure mapping	5
Interdependent complexity	Preventing additional complexity	4
	Structure formation for unstructured data	3
	Incomplete data	2
Fluidity of design	Data model changes over time	4
	Data model changes during runtime	4

and processes lead to transaction errors, which cause data usage problems as a further consequence (Cao and Zhu 2013). Since data quality has a direct impact on operational tasks and strategic decisions (O'Brien 2015), transaction errors can lead to both hidden and direct costs to the disadvantage of the business. The requirement of **performance** is thus an aspect that meets the added value of data usage and thus automatically generates an improvement in the quality of analytical data models. Especially in relation to IoT enabled ERP systems, the aspect of an energy efficient, resource saving, cost efficient and reliable functionality will be of great importance (Tavana et al. 2020). Two similar requirements stems from the vision of the smart factory, which provides accelerated data flow and insight into partners, suppliers, and customers through **real-time capability** (Koh et al. 2011; Tavana et al. 2020) and the ability to virtualize a **digital twin** (Sinha and Roy 2020). The idea of intelligent enterprise systems (vom Brocke et al. 2018) that dynamically adapt to the user and collect and process large amounts of data in real time is also considered to drive enterprise architectures (Romero and Vernadat 2016b). As technologies and business models evolve, it is also expected that providing real-time information for decision-making will involve greater

"runtime integration"(ibid.). Accordingly, future systems must be highly collaborative, with all artifacts able to connect, communicate, and share information and knowledge, while virtual copies of the physical material simulate and validate the entire manufacturing process (Sinha and Roy 2020). However, with the availability of different devices, operating systems, platforms and services, it becomes necessary to adequately combine and manage a large number of very **heterogeneous business objects** and their correspondingly generated data (Tavana et al. 2020). Already today, the inconsistency of internal data model and external data models (when changing data and data patterns of data sources) leads to challenges in the design of data service architectures (Liu et al. 2014), which can be attributed to rapid growth and a dynamic business environment (Cao and Zhu 2013).

2.4.2 Integration focus

Requirements for the theme of *integration focus* address the aspect of integrating existing applications. The challenge of integrating any form of data across the enterprise, as well as accessing, manipulating, and integrating across multiple data sources, is increasingly explored in the literature (Panetto et al. 2015; Romero and Vernadat 2016b). As products and organizations become more complex (El Kadiri et al. 2016), information and communication technologies must be designed to guarantee support of interplay networks (Panetto et al. 2016). This inevitably leads to the discussion of **schema interoperability** in information systems. In addition to current challenges such as data interoperability (i.e., the electronic exchange of data as well as the comprehensibility of the representation, the future will face further barriers on a conceptual, technological and organizational level when designing interoperable systems (Weichhart et al. 2016). Other studies have addressed this issue and coined a new term for heterogeneous database system by introducing "polyglot persistence", which is a proposition of a unified metamodel for relational and the NoSQL paradigms (Candel et al. 2022). Nevertheless, today's information technology users face the challenge that enterprise systems are usually so complex that it is difficult to access distributed, heterogeneous data (El Kadiri et al. 2016). Therefore, improvements in extract, transform, load processes (specially enhanced and automated extraction and transformation of data) are required both for better efficiency in handling large amounts of data (Sinha and Roy 2020) and dealing with heterogenous types of data. **Extract and transform steps** provide the starting condition for future information architecture to be aligned with Big Data as well as predictive, prescriptive and inductive analytics and social business intelligence (Romero and Vernadat 2016b).

2.4.3 Distributed requirements

Distributed requirements include both, requirements of diverse stakeholders as well as those that are distributed because of organizational and geographical distances in business operations. Increasingly, companies are realizing the benefits of collaborating with partners by integrating core competencies and linking processes. However,

this new type of collaboration requires transactional accuracy at the data level (Koh et al. 2011). The new arising complexity can only be addressed if **compliance requirements** are already integrated into the **data model**. For example, locating and removing important personal data in distributed tables in ERP systems is a major challenge for many organizations today (Politou et al. 2018). Often, GDPR compliance tools are used to control access to stored personal data. But also in terms of AI-based services in future enterprise system, data that will be processed to compute information systems in organizations need to be legally identified (Frick et al. 2019). The next step is to classify the data to determine whether it is suitable for distribution to business partners (Liu et al. 2014). The further automation of business processes across company boundaries requires that databases communicate independently about the data elements concerned—in such a case, the **cooperation suitability** must be recognizable in the **data model**. Enterprise landscapes are becoming increasingly digitized and connected, which is shaped by business collaboration that share and exchange data (Liu et al. 2014). As dealing with security issues for cloud-ERP is already a challenging and complex process today (Abd Elmonem et al. 2016), the future information system landscape with sensor networks and IoT will increasingly have to consider privacy and security aspects (Tavana et al. 2020). Consequently, the mapping of **data security** as well as **data protection in the data model** can be classified as future requirements for databases of information systems.

2.4.4 Layers of requirement

Requirements that address different levels of abstractions are addressed in the theme of *layers of requirement*. As data models increase in complexity and size (Lapalme et al. 2016), human readability support is required for various layers of the database, as well as their content and structural representation. Alberts (2013) identified information system usability and information fragmentation as one of the biggest challenges for knowledge workers in the digital workplace. Therefore, **comprehensibility of data models** becomes another important requirement for future databases in enterprise systems. This is primarily understood to include the context, relationship and content of the data presented. It is important to note that the comprehensibility of the data model automatically entails a higher demand for **granular security settings** from superordinate database production to table-based selections. In particular, regulations such as the GDPR or inter-company relationships provide strict and specific requirements for the management, verification of access and processing of data (Politou et al. 2018), which have to be secured even more under the condition of better comprehensibility.

2.4.5 Centrality of architecture

Requirements that focus on architectural circumstances and thereby indirectly influence the application are collected in the theme of *centrality of architecture*. Increasing complexity and rising business requirements in today's ERP systems not only lead to reduced process understanding from the user's perspective (El Kadiri et al. 2016) but often result in unintended and unfamiliar feedback loops (Cao and

Zhu 2013). At the same time, systems are continuously customized to support the requirements of a changing business environment (Cao and Zhu 2013), causing an even greater complexity. Since business collaboration requires data to be shared and exchanged (Liu et al. 2014), process interoperability requires service exchange coordination (Weichhart et al. 2016). The aspect of **process structure mapping** thereby becomes another requirement for databases of future enterprise systems to realize linking core business processes internally and externally (Koh et al. 2011). In order to integrate processes quickly, the ability to decompose system components and group them into smaller independent subsystems is also required (Sinha and Roy 2020). **Modularity** thereby not only enables the integration of individual processes but also represents the initial condition for achieving different products, variants or increasing the production volume.

2.4.6 Interdependent complexity

In general, complexity of software systems has risen significantly. Therefore, requirements that address complexity of requirements (especially interdependencies) are subsidized in the theme of *interdependent complexity*. Several studies have identified increasing complexity in the corporate context as one of the major challenges of the future (Lapalme et al. 2016; Alberts 2013; Koh et al. 2011). In this context, it becomes even more important that users of information systems can easily find the right, important and relevant information at the right time (El Kadiri et al. 2016). **Preventing additional complexity** thus becomes an important requirement for the design of future digital work. In the long term, information systems will not only face the challenge of processing different document, software and data formats (Alberts 2013). Rather, under the new characteristics of the Internet of Things, the requirement of **structure formation for unstructured data** and dealing with **incomplete data** becomes important (Liu et al. 2014). Dobaj et al. (2018) has addressed these different data management requirements in a study and attempted to develop an application that could be suitable for use in the upcoming IoT environment. It was found that different services have different data storage requirements. While for some the relational database was the best choice, other services may have needed a NoSQL database (ibid.). As semi-structured, unstructured and incomplete data sets become more relevant for value creation, databases must enable the (efficient) integration, transformation and storage of these forms of data.

2.4.7 Fluidity of design

So far, requirements addressed the design of the database. In contrast, the theme of *fluidity of design* addresses requirements concerning the continued evolution of the database, i.e. post-implementation. In a study of data quality issues in ERP systems, Cao and Zhu (2013) found that in particular dynamic and rapidly changing business environments cause a mismatch between technical changes and the stable working environment required by ERP systems. Enterprise systems are often customized after implementation, (Table 3) which immediately increases their inherent complexity. As technologies and business models evolve, "runtime integration" is

Table 3 Fluidity of design database evaluation

Fluidity of Design aspect	Relational database	Column-based database	Object-relational database	Graph database	Key-value database	Document- based database
4.4.1 Data model changes over time	0	0	2	2	1	0
4.4.2 Data model changes during runtime	0	0	1	2	2	2

expected to increase (Romero and Vernadat 2016a). This means that future systems will need to represent collaborative enterprises by not only allowing **data models to change over time**, but also achieving constant business IT adaptation.

2.4.8 System transparency and packaged software

The key themes of system transparency (i.e., requirements for cross-application usability) and packaged software (i.e., requirements concerning commercial off-the-shelf software) are not directly addressed by the database requirements as those are directed primarily towards higher-level system components. Those priority presented (ERP) database requirements are confronted with currently existent and emerging database technologies with respect to (unexpected) fulfillment of those requirements.

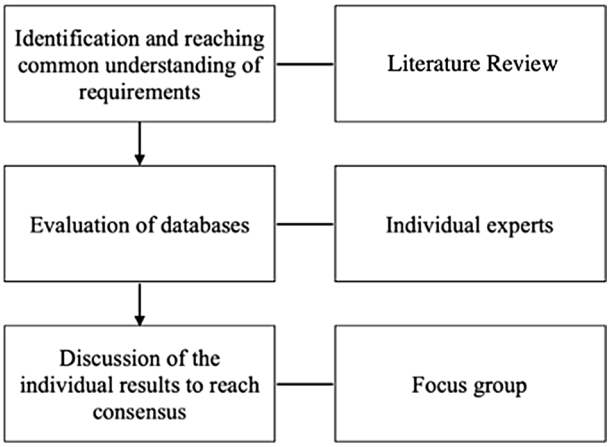
3 Methodical approach

This paper has an exploratory nature. To effectively address the complexity and multifacetedness of the topic, qualitative methods were used. This is a suitable approach because little knowledge about the use of different databases with ERP systems is available in research and practice (Remenyi et al. 1998). This is because relational databases are the primary choice of ERP systems vendors. In the past, no research has been conducted on the possibility to use other databases for ERP systems. Since ERP systems have many problems, especially at the database level, there is an urgent need for action and at the same time it opens up a research gap. Table 4 depicts the method used for this study.

3.1 Database assessment methodology

The identification of requirements using a systematic literature review resulted in a total of 23 requirements. This list of requirements forms the basis for the evaluation of different databases that was finally conducted by the same focus group. According to Finch and Lewis (2003) focus groups differ from in-depth interviews because data emerge in interactions among participants, in this case ERP researchers. Different perspectives and experiences are represented. Reflection takes place in the

Table 4 Research method



group, questions are asked of each other to bring about clarification (ibid.). A goal of the focus group was to gain a common understanding on the database related requirements for future ERP systems.

To evaluate the suitability of requirements, related definitions were used as a basis for judgement. For operationalization different specifications for the respective levels were provided. Each database was evaluated with regards to its response to the requirements identified on a scale ranging from 0 to 2. Each researcher evaluated the addressing of the requirement per database individually. The results were discussed afterwards in the research group to achieve intersubjectivity on the matter which is important in qualitative research. The final result was reached by consensus based on the individual results.

This methodological procedure allowed for identification of requirements and evaluation of relevant database types regarding the requirements and is subsequently used for a basis of discussion of future ERP database models.

3.2 Example requirement

To illustrate the approach concerning the requirements assessment an example is provided. The complete list of underlying definitions and respective level can be found in the appendix. The aspect of comprehensibility of the data model is identified in different contexts of enterprise systems (Alberts 2013); (Weichhart et al. 2016). The definition for comprehensibility follows the understanding of Gleicher (2016):

Definition: Comprehensibility is understood as the ability of the various stakeholders to understand relevant aspects of the data model.

Subsequently the different level for the database assessment were identified:

Level 0: Data model is only comprehensible for the creator or database experts.

Level 1: Data model is comprehensible for at least database designers and key-user.

Level 2: Data model is comprehensible for software users (e.g. by visualization and explanations).

Based on the definition and the operationalization with corresponding levels, each database is assessed.

4 Results of evaluation of the databases

The six databases were evaluated to determine whether they meet the requirements for future ERP systems. A common understanding about the individual requirements was reached within the focus group and can be seen in the Appendix 1 This section is structured following the key themes by Hansen et al. (2009) starting with the business process focus. For every requirement the evaluation of the different databases is presented.

4.1 Business process focus

4.1.1 Scalability

With regards to scalability, all databases are evaluated to be suitable except graph databases (Table 5). After the structure of a relational database has been defined, new data records can be stored in it. It is also possible to import existing records from other databases and tables, provided that the tables have an identical structure and the data types are compatible (Arnold et al. 2019). Adding new columns (contents) to a table does not pose any difficulties for a column-based database (Sridhar and Johnson 2018). Object-relational databases allow the definition of object types. Each database object can be structured and identified by a unique object identifier and provided with properties (with associated operators) (Brakatsoulas et al. 2004). Properties of existing objects can be inherited by new sub-objects, so that operations need to be coded only once. In particular, the property of mapping inheritance allows new data objects to be easily created without effort. Key value stores consist of a simple data structure without an extended query language. Key value pairs are stored as tuples of two strings. Arbitrary key value pairs can be included in the store and stored as lists or arrays. A record in a key value store consists of exactly one key and exactly one value. Several such key-value pairs are managed in so-called buckets. A bucket therefore resembles a two-column relational table. Due to the minimalism on the conceptual and internal level, key value stores exhibit very good speed and scalability properties (Chen et al. 2017). Document stores represent data sets as independent documents. A document is a structured collection of multiple key-value pairs and usually conforms to Java Script Object Notation (JSON) or Extensible Markup Language (XML) formats. Attributes and values of a dataset are indexed using key-value pairs in the corresponding document. Documents are managed within collections and uniquely identified by the key-value pair with the respective key-id. The primary storage unit within JSON is a so-called object, which contains several unsorted key-value pairs. The value of a key-value pair can correspond to

Table 5 Business Process Focus Database Evaluation

Business Process focus aspect	Relational database	Column-based database	Object-relational database	Graph database	Key-value database	Document- based database
4.1.1 Scalability	2	2	2	1	2	2
4.1.2 Performance (OLAP)	0	2	0	2	0	0
4.1.3 Performance (OLTP)	2	0	1	1	2	2
4.1.4 Performance (OLTP) multiple datasets	1	1	1	0	0	0
4.1.5 Real-time capability	0	0	0	0	2	0
4.1.6 Digital twin	0	0	2	2	0	2
4.1.7 Business objects heterogeneity	0	0	1	2	2	2

different primitive data types and can be extended arbitrarily (Harding et al. 2003; Balmin et al. 2004). Data objects with specific properties (equivalent to entity) are represented as nodes in graph databases. Connections between these objects or the characterizing relationship between the objects are represented by edges. The number of nodes and edges of a represented relation determines the effort in scaling (Chebotko et al. 2013). With increasing complexity the scaling becomes more complex. On the contrary, for dynamic data models, which represent strongly linked data, the graph database is very suitable (Yoon et al. 2018).

4.1.2 Performance (OLAP)

As for performance in the sense of OLAP, graph databases and column-based databases meet these requirements, whereas all other databases are not suitable at all. In the relational representation, the storage of an object is segmented across many different relations. Since the relational model only knows tuple sets consisting of values complex application objects must be recovered from the individual relations by means of numerous joins when queried by the DBMS. Relational databases are not designed for multidimensional analysis. Summarizing data is a time-consuming operation in relational databases, especially when all tuples of a relation have to be considered (Kolahi 2007). In a column-based database, the tuples are stored column wise by creating a new file with its own indexes for each column. As a result, a column-based database reads only the columns that are needed in a particular query without being interfered in processing by unnecessary row contents (Plattner 2009). In contrast to the relational approach, the column-based approach allows fast joins and aggregations (Xu et al. 2016). Since the tables are already sorted and all values of an attribute are stored continuously, there is no need for sorting before joining or aggregating. Since object-relational databases represent the extension of relational databases by adding object-oriented structures, the restrictions of relational databases remain. Object-relational databases are also not designed for multidimensional analyses. Keys and values of a key value data set are stored as simple byte arrays that cannot be interpreted by the respective key-value database. It is therefore the task of the application logic to convert a key-value pair into the desired data format after reading it from a key-value store. Analytical queries cannot be executed within a key-value database (Atikoglu et al. 2012). Graph databases enable fast answering of multidimensional analytical queries (He and Singh 2008). Since relations as a construct are directly supported by the database, no JOIN operations are necessary. Traversals (breadth-first or depth-first search) is thus efficiently possible. Calculations such as the simplest or most efficient search through a graph can be realized in this way (Ciglan et al. 2012). Additionally, set operations are feasible as union set or intersection set of graphs. Measures used to describe graphs are (1) order = number of nodes (2) size: number of edges (3) degree = number of edges directed to a node (4) proximity = distance of a node from all other nodes of the graph and (5) distance = distance between two nodes. Since different data sets are mapped as independent documents in the document database. Data processing is not intended for multi-dimensional analyses (Chien et al. 2001).

4.1.3 Performance (OLTP) for performing a task on an individual data set including several fields

As for performance in the sense of OLTP and more specifically with regards to performing a task on an individual data set including several fields, relational, key-value and document-based databases are most suitable. Column-based databases score the least. Due to the ERD (entity relation diagram) characteristic of relational databases, fast and application-oriented, specific changes or search queries of a data set can be performed in a straightforward manner by means of unique indexing (Thomson et al. 2014; Tongkaw and Tongkaw 2016). Since a new file is created for each column in column-oriented databases, each individual column must also be loaded and manipulated individually when processing a contiguous data set. The effort required for such a request is directly related to the data complexity of the respective data object. While updating a single row in a row-oriented database can be written by a single access, the column-oriented approach requires multiple writes (Kanade and Gopal 2013). Due to the similarity to relational databases, fast and application-oriented specific changes or search queries of a data set can be carried out in an uncomplicated way by unique indexing of an object. Since different objects can be addressed and changed individually, there is a high degree of heterogeneity (Bertino and Martino 1991). The lack of standardization of the objects is at the expense of performance. The request and change operations offered by Key Value Stores are reduced to an absolute minimum. Only the two change operations set and delete are available for manipulating the dataset. These operations can be used to store and delete individual key-value pairs. Graph databases additionally enable the management of transaction-oriented queries (Chebotko et al. 2013). Since information is not stored in different tables, there is no need to perform join operations, such as those formulated in SQL. However, in graph databases, information is searched for by visiting neighboring nodes (He and Singh 2008). The effort of this operation is linear to the number of nodes stored. Document stores represent data sets as independent documents. Attributes and attribute values of a data set are stored in the corresponding document using key-value pairs. Documents are managed within so-called collections and uniquely identified by the key-value pair with the key-id. The modification of a specific record can be realized on the basis of XML or JSON requests (Li and Manoharan 2013).

4.1.4 Performance (OLTP) for performing a task across many data sets

None of the databases really meet the requirement of performing a task across many datasets. Since each object is broken down into individual parts and the data is processed in a quantity-oriented manner, queries on large data volumes and powerful operations against the data are complicated in relational databases (Kolahi 2007). However, the performance of an OLTP query depends on the processing by the relational database system. If operations involve non-specific queries or manipulation across multiple data sets, the performance of relational databases decreases. The disadvantage of column-based databases is their low query power (Kanade and Gopal 2013). Although column-based databases provide many of the operators known

from relational databases, the formulated restriction conditions can only be executed on the primary keys of a table. Regarding object-relational store, the performance of an OLTP query depends on the processing by the relational database system. As a result, the performance of object-relational databases decreases if operations involve non-specific search queries or manipulation across multiple data sets. The query processing of key-value databases is not quantity-oriented, but record-oriented (Mei et al. 2018). Thus, a data manipulation over multiple Key Values represents a large number of transactions. Since the effort of a search query in graph databases increases linearly with the number of nodes stored, performance is degrading over many data sets (Angles and Gutierrez 2008). Since document databases do not have a fixed database schema (Chung and Jesurajaiah 2005), the structure of the stored documents cannot be defined in advance at database level. For this reason, it is possible for individual documents within a collection to differ completely from one another in terms of structure. Modification across many records is therefore not feasible.

4.1.5 Real-time capability

For real-time capability only the use of key-value databases is suitable at full degree because all the other databases real-time capable data processing or storage. Instead, due to the minimalism nature at the conceptual and internal level, key value databases have very good speed and scalability characteristics. Since query and modification operations are performed with virtually no effort due to the absence of ACID properties, key value stores typically achieve very high transfer rates for simple read and write requests while maintaining very low latency (Wang et al. 2020).

4.1.6 Depiction of a digital twin

The depiction of a digital twin is allowed by object-relational, graph and document-based databases. The remaining databases do not meet this requirement. Relational, column-based and key-value databases do not have any representation options that do justice to such an approach. On the contrary, the object orientation of object-relational databases allows to create objects of a concrete product. Graph databases are particularly suitable for dynamic data models that represent strongly linked data and can represent the life cycle of an object. Through the properties of the property graph, nodes and edges can be assigned properties in which information is stored (Alocci et al. 2015). The schema-free property of document store databases makes them suitable for representing a digital twin by storing the sum of all properties in one object.

4.1.7 Heterogeneity of business objects

The requirement of heterogeneity of business objects is very well met by graph, key-value and document databases. Relational and column-based databases are not suitable at all. In order to evaluate the heterogeneity of business objects, an analysis of the relation schema is required. Usually, relations are defined as a set of tuples

with the same attribute labels in a relational database. Tuples can be understood as a manifestation of a business object and are defined by a set of attributes. The compatibility between different tuples is based on the compatibility to the same relation schema, i.e. the same attribute identifiers must be used. The dependency of a tuple on a relation schema means that it is not possible to map heterogeneous business objects in relational databases (Kolahi 2007). Since column-based databases do not contain a global schema but manage self-describing structure, the data records located in a table can differ from each other in their structure. A mapping of generalized and specialized entities to different tables, as required in relational data modeling, is therefore not necessary. Object-relational databases allow different representations of the same object by defining new object classes. In addition, object-oriented databases allow the definition of arbitrarily complex objects and thus permit realistic modeling with high structural complexity (Mo and Ling 2002). Within key-value databases, certain relationship information can be redundantly stored in multiple entities. This redundancy can ensure heterogeneity of business objects. In a graph database, each individual node and edge can have its own information and its own properties (He and Singh 2008). Information and properties can be added to or deleted from existing nodes. With some restrictions, this also applies to edges. Since document databases do not have a fixed database schema (Chung and Jesurajaiah 2005), the structure of stored documents at the database level cannot be defined beforehand. As a result, it is possible for individual documents within a collection to be completely structurally different from one another.

4.2 Centrality of architecture

4.2.1 Modularity (division of the scheme into modules)

To meet the requirement of modularity, key-value and document databases should be the choice while relational and column-based databases should be neglected (Table 6). In the relational representation, the storage of an object is segmented across many different relations (Kolahi 2007). Since the relational model only knows tuple sets consisting of values, complex application objects must be restored from the individual relations by means of numerous joins when a query is made by the DBMS. This fact quickly makes SQL queries complicated and confusing, to the drawback of query complexity (Leinders and Van den Bussche 2007). It is not possible to split the system into additional modules. In column-based databases, the

Table 6 Centrality of Architecture Database Evaluation

Centrality of architecture aspect	Relational database	Column-based database	Object-relational database	Graph database	Key-value database	Document- based database
4.2.1 Modularity	0	0	1	1	2	2
4.2.2 Process structure mapping	0	0	0	2	0	0

mapping of individual records in a table is not subject to any structural restrictions on the part of a global database schema. The storage of an object segments on columns. However, the division into further modules is not possible (Kanade and Gopal 2013). Object-relational databases allow the definition of object types (often called classes in reference to object-oriented programming), which can be composed of further object types. Only two first-order predications are stored in a key-value database. These are the predication between the name of the set and the corresponding key values, and the predication between the key values and the associated attribute values. Since key value stores consequently have no database schema in the true sense, they are also referred to as schema-free. Since keys are the only way to interact with data objects in key value stores, their modeling determines the performance and scope of the queries to be supported (Mei et al. 2018). Graph databases are optimized to efficiently store and make tangible highly interconnected information (Mai et al. 2018). These can be subdivided into sub-graphs at most. Document databases are, at the first level, a kind of key-value database. For any given key (the document ID), a record can be stored as a value. These records are called documents. On the second level, these documents now have their own internal structure. The documents have no relationship to each other, but contain a self-contained collection of data (Chung and Jesurajaiah 2005).

4.2.2 Mapping of the process structure through data management

To be able to map the process structure through data management, only graph databases are evaluated to fit while all others are not suitable. Relational databases are suitable for the implementation of administrative activities; however, the support of dispositive activities is not given due to the lack of representation of structural analogies. A process-oriented data management for the representation of specified processes cannot be realized by the relational and object-relational schema. Due to the minimal data structure of key-value databases, where keys are the only way to interact with data objects, this approach does not provide process-oriented data storage. By configuring framework conditions of a process sequence in a document, corresponding data structures can be displayed. However, this mapping is only possible within a specific document and cannot be applied to the entire database. Since a graph database consists of nodes and edges that connect related information and properties, business processes can be abstracted and formalized to the graph structure (Ciglan et al. 2012). Triggers of an event can be guided by the sequence of activities while input conditions for determining activities can be defined by properties of the graphs.

4.3 Distributed requirements

4.3.1 Mapping of data protection in data model

The requirement of mapping of data protection in the data model is met best by object-relational, graph and document databases whereas key-value databases are

not a good choice (Table 7). Although relational databases have data protection mechanisms that provide authorized users only with the tables or table sections necessary for their activity (implementable by views). Nevertheless, there are no options for the general attribution of selected data to indicate a need for protection (Miklau et al. 2007). Column-oriented databases do not offer an approach for the representation of data protection aspects. At most, such data can be identified by appropriate attribution. Through the property of inheritance of object-relational databases (Lorenz 2015), information regarding data protection can be indicated both on the structure and on associated operators. Information regarding privacy can be assigned at the node and edge level. In addition, the relationship of a selected node to networked information can be marked as requiring protection (Comyn-Wattiau and Akoka 2017). Since meta information can be stored in every data record (and thus in every document) document-based databases are suitable for the representation of data protection (Naedele 2003; Brakatsoulas et al. 2004). Key value databases cannot manage additional meta information such as attribute names or data types.

4.3.2 Mapping of compliance requirements in the data model

As for mapping of compliance requirements in the data model, the result is similar to the mapping of data protection. Relational and column-based databases do not provide an approach for implementing end-user policies on personal data, its processing or disclosure. At most, such data can be identified by appropriate attribution. By inheritance, information can be made recognizable regarding compliance both on the structure and on associated operators of object-relational databases (Lorenz 2015). Key-value databases cannot manage additional meta information such as attribute names or data types. Information regarding compliance requirements can be assigned at the node and edge level within graph databases (Comyn-Wattiau and Akoka 2017). In addition, the relationship of a selected node to networked information can be identified as compliance relevant. Since meta information can be stored in every data set (and thus in every document), document-based databases are suitable for the representation of compliance-relevant conditions (Naedele 2003; Fong et al. 2010).

4.3.3 Mapping of cooperation suitability in the data model

For the mapping of cooperation suitability in the data model the same evaluation results as for the previous two requirements applies. The inheritance property of object-relational database (Lorenz 2015) can be used to indicate information regarding the suitability of the structure for cooperation as well as the associated operators. Key-value databases cannot manage additional meta information like attribute names or data types. In graph databases, information regarding cooperative properties can be assigned at the level of nodes and edges (Comyn-Wattiau and Akoka 2017). In addition, the relationship of a selected node to networked information can be marked as cooperation-significant. Since meta information can be stored in each data set (and thus in each document) document-based databases are suitable for the representation of cooperation suitability (Naedele 2003; Fong et al. 2010).

Table 7 Distributed requirements database evaluation

Distributed requirements aspect	Relational database	Column-based database	Object-relational database	Graph database	Key-value database	Document- based database
4.3.1 Data protection in data model	1	1	2	2	0	2
4.3.2 Compliance in data model	1	1	2	2	0	2
4.3.3 Cooperation suitability in data model	1	1	2	2	0	2
4.3.4 Data security in data model	1	1	2	2	0	2

4.3.4 Mapping of data security in relation to stakeholders in the data model

For the mapping of data security in relation to stakeholders in the data model, the evaluation is also similar to the other distributed requirements. Relational and column-based databases do not provide an approach for ensuring data security with respect to stakeholders. At most, such data can be identified by appropriate attribution. The inheritance property of object-relational databases (Lorenz 2015) can be used to indicate information regarding data security with respect to stakeholders, both on the structure and on associated operators. Key-value databases cannot manage additional meta information such as attribute names or data types. Information regarding data security related to stakeholders can be assigned at the node and edge level (Comyn-Wattiau and Akoka 2017). In addition, the relationship of a selected stakeholder node to networked information requiring protection can be indicated. Since meta information can be stored in each data set (and thus in each document) document-based databases are also suitable for the representation of data security in relation to stakeholders (Naedele 2003; Fong et al. 2010).

4.4 Fluidity of Design

4.4.1 Allow data models to change over time

Object-relational and graph databases are the only databases that allow data models to change over time in a sufficient manner. The normalization process of relational databases creates dependencies that always refer to attributes within a table. The dependencies of the attributes among each other lead to the fact that tables must be divided or attribute values must be distributed on several tuples. Although changes to relational relationships can be made, these changes would have to be made consistently across all affected relationships (and thus all other data sets) to prevent mutation anomalies (Kolahi 2007). Column-based databased store the attributes of a table column-wise and not row-wise. Changes can also be made here; however, these changes would have to be made consistently across all affected columns (and thus further data sets). In object-relational databases, properties of objects can be inherited (Lorenz 2015). This inheritance property applies to both the structure and the associated operators. The classification of object types allows the data model to be easily modified. Graph databases are particularly suitable for dynamic data models and can be adapted at any time via the addition or deletion of nodes and edges (Mondal and Deshpande 2012). Key-value databases do not have a data model; only the modification of value pairs is possible. With document databases it applies that the ID is always a part of the document and not separately regarded. Accesses are not only possible via the ID, but—similar to SQL—any selections on secondary attributes are supported. However, if the key is changed, no more queries can be made.

4.4.2 Allow data models to change during runtime

To allow data models to change during runtime, the use of graph, key-value and document databases is suitable. In relational and column-based databases changes cannot be executed across multiple records at runtime, therefore a record change transaction would inevitably lead to data inconsistencies, which in turn would lead to errors in individual queries. In object-relational databases properties of objects can be inherited (Lorenz 2015). This inheritance property applies to both the structure and the associated operators. Classification of an object type can also be performed during runtime. Graph databases are particularly suitable for dynamic data models and can also be adapted at runtime by adding or deleting nodes and edges (Mondal and Deshpande 2012). Key value pairs can also change at runtime. There is no coherent data model that is affected by changes at runtime. Due to the dependency between ID and document changes at runtime can only be made within the structure of a document.

4.5 Interdependent complexity

4.5.1 Preventing additional complexity

The requirement of preventing additional complexity is not met well by any of the databases. Object-relational and graph databases behave neutrally in this regard (Table 8). In the relational representation, the storage of an object is segmented across many different relations. Since the relational model only knows tuple sets consisting of values, complex application objects must be restored from the individual relations by means of numerous joins when a query is made by the DBMS (Kolahi 2007). This reality quickly makes SQL queries complicated and confusing (Leinders and Van den Bussche 2007), to the detriment of query complexity. In the column-oriented representation, the storage of a tuple is segmented to different columns in a separate structure. A new file is created for each column (Abadi et al. 2008). The complexity increases linearly with the properties of the objects. In object-relational databases, object classes can be formed with associated object types. However, the lack of a standard can lead to heterogeneity among the various implementations, to the drawback of the complexity of the DBMS. Graph databases are optimized for traversal, i.e. navigating accesses. However, traversal performance suffers from this complexity as the level depth of the query progresses (Ciglan et al. 2012). Key value databases only have the representation of key value pairs. As the amount of data increases, it leads to an increasing amount of complexity within the database. A collection located in a document database contains a set of documents. The complexity comes only from the possible structure of these documents (Arenas and Libkin 2004). It is true that the ID is always a part of the document and is not considered separately. Accesses are not only possible via the ID, but any selections on secondary attributes are supported. For efficient searching, document databases support the definition of

Table 8 Interdependent complexity database evaluation

Fluidity of Design aspect	Relational database	Column-based database	Object-relational database	Graph database	Key-value database	Document- based database
4.5.1 Preventing additional complexity	0	0	1	1	0	0
4.5.2 Structure formation for unstructured data	0	0	0	0	0	1
4.5.3 Incomplete data	2	2	2	2	0	2

indexes on these very attributes. An index on the document ID is usually always present.

4.5.2 Structure formation for unstructured data

The structure formation for unstructured data is a requirement that is not met by any of the databases presented. Document-based databases are the most suitable in comparison. Relational, column-based and object-relational databases are able to store BLOBs (Binary Large Object) (Shapiro and Miller 1999), which for example contain image or audio files. However, the unstructured content of BLOB files cannot be read or processed by relational database types. This means that database functions such as sorting, filtering or searching for specific content are not possible in a BLOB. Although, unstructured data can be stored in key-value and graph database, no structure formation takes place. Document-based databases represent data records as independent documents. Due to this fact, unstructured data can be represented semi-structurally (Arenas and Libkin 2004).

4.5.3 Incomplete data

The requirement of incomplete data is addressed in all databases with the exception of key-value databases. In relational, column-based and object-relational databases incomplete data can be mapped by NULL values (Zaniolo 1984). Since key-value databases consist of only one key-value pair they cannot be mapped if a component is missing.

4.6 Integration focus

4.6.1 Support of the extract step (ETL process)

Relational, column-based and object-relational databases support the extract process in the ETL process well while the other databases do not support it (Table 9). This is because of the given structure of relational, column-based and object-relational databases (Morris et al. 2008). Due to the missing structure of graph, key-value and document-based databases the extract process is not supported.

Table 9 Support of the extract step (ETL process)

Integration focus aspect	Relational database	Column-based database	Object-relational database	Graph database	Key-value database	Document- based database
4.6.1 Extract step (ETL)	2	2	2	0	0	0
4.6.2 Transform step (ETL)	1	1	2	0	0	0
4.6.3 Scheme interoperability	2	2	1	0	0	0

4.6.2 Support of the transform step (ETL process)

For the support of the transform step in the ETL process a similar image is drawn like for the extract step but in this case relational and column-based databases do not perform as well anymore. Object-relational databases have schemas with metadata (Huynh et al. 2000), that support the transform process. All other databases have no automation of schema metadata. In relational databases data can only be transformed by defining primary and secondary keys. Graph, key-value, and document-based databases are missing a suitable structure for the ETL process.

4.6.3 Scheme interoperability

As for scheme interoperability relational and column-based databases perform best, next in line are object-relational databases. The remaining databases are not suitable. The given schema within relational and column-based databases allows the transformation to other schemas. The predefined schema within object-relational databases allows the transformation to other schemas. However, the transformation of the schemas depends on the heterogeneous initial situation of the defined object types. Graph and key-value databases do not have a schema and are not suitable for transformation to another schema. The schema of individual documents in document databases can be customized. Document databases do not have a uniform schema (Chung and Jesurajaiah 2005) and are therefore not schema interoperable.

4.7 Layers of requirement

4.7.1 Granular security settings

Regarding granular security settings, the use of object-relational and graph databases is most suitable. Key-value and document-based databases are not recommended in this regard (Table 10). For relational and column-based databases, the granular security settings apply down to the column level. In object-relational databases individual objects can be secured by encapsulation (Torres et al. 2017). With graph databases, security settings can be made down to the metadata level. Key Value databases offer neither a data model nor approaches for granular security

Table 10 Layers of Requirement Database Evaluation

Layers of requirement aspect	Relational database	Column-based database	Object-relational database	Graph database	Key-value database	Document- based database
4.7.1 Granular security settings	1	1	2	2	0	0
4.7.2 Comprehensibility of data model	0	0	1	2	0	1

settings. Also, no granular security settings can be made within documents in a document database.

4.7.2 Comprehensibility of the data model

The requirement of comprehensibility of the data model is solely addressed well in the graph data base. Not well suited are relational, column-based and key-value databases. Neutrally suitable are object-relational and document databases. The decomposition of data into individual parts required during normalization in relational databases makes it difficult to search for more complex relationships in terms of content and performance (Kolahi 2007). Data that is stored separately from one another, but is to be used together to create information objects, must be brought together again at the application level in the form of joins. Consequently, knowledge of the basics of the SQL database language is necessary for understanding relational database applications. In column-bases databases, data is broken down into individual parts that must be reunited at the application level in order to be used together. The classification of individual object types in object-relational databases facilitates the understanding of the data model. Since different objects can be addressed and changed individually, there is a high degree of heterogeneity (Bertino and Martino 1991). The lack of standardization of the objects is at the expense of complexity. Graphs are suitable for the representation of complex interrelationships (He and Singh 2008) are helpful for the comprehensibility of the data model by the spatial relation due to the graphic representation of nodes and edges. The key-value database is based on a table with only two columns: One contains the value, the other contains a key. Especially under growth of the data volume, this representation is not suitable for the comprehensibility of the data model. In document-based databases a structure for the administration of the data can be given within a document. This can contribute to the comprehensibility of the data model.

5 Discussion

This paper followed the goal of exploring database models with regard to future requirements for ERP-Systems. For that, a classification of future requirements for data storage in ERP databases was presented and evaluated for different database models. In the following, the results of this evaluation are discussed. Implications for future research and for practice are given and limitations of this study are presented.

As relational and column-based databases have more static database schemes and therefore share similar characteristics regarding the data organization, those two databases are subsumed as tabular databases in the following. Object-relational, graph and document databases provide more flexible, object-oriented schemes. Those database types are therefore called structural-analog databases. Lastly, schema-less key-value databases form their own category as they do not share structural similarities with the other types.

As mentioned before, tabular databases are prevalent in the realm of ERP systems (Plattner 2009). Besides the problems associated with tabular databases for ERP

systems, the previously presented analysis revealed disadvantages and advantages of tabular databases. In general, tabular databases provide advantages over structural-analog databases regarding the theme of integration focus. The static, normalized schemas of tabular databases allow for seamless integration of ETL-Processes and database schemes. Contrarily, the theme of distributed requirements (i.e., directly and flexibly extending the data model and modeled business objects with metadata) is extensively addressed by schema-analog databases. Pure tabular representations prevent object-orientation and granular processing of data objects (see also the theme of layers of requirements). This flexibility and object-orientation further allows structural-analog databases to embrace characteristics of fluidity of design, i.e., continuous development and altering of data structures. This is also observed for architecture requirements in the theme of centrality of architecture. Modularity and structural analogy are primarily addressed by structural-analog databases. Especially in terms of the latter, tabular databases fail to satisfy the requirements. This partially results in less comprehensible data models (layers of requirements) compared to structural-analog databases, too. For the theme of business process focus, the analysis did not reveal general, decisive advantages or disadvantages between tabular and structural-analog databases. Still, as managing business processes is the primary responsibility of ERP systems, the business process focus is of special importance for future database models.

In total, the analysis of the priorly assessed requirements revealed strengths and weaknesses for all database models (Fig. 2). Traditional, tabular databases offer advantages for storing and processing similar and reoccurring datasets of

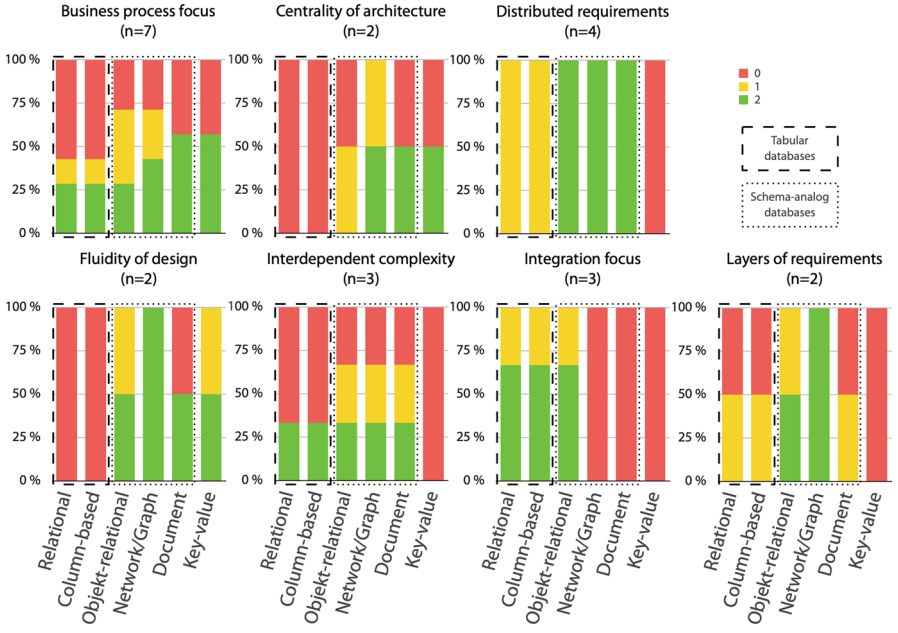


Fig. 2 Fulfillment of ERP requirements by database type

transactional data (e.g., process data, billing data, sensor data). On the other hand, structural-analog databases excel in mapping individual, flexible and changing business objects. Previous studies that evaluated NoSQL database models for enterprise systems confirm our findings; increased flexibility of NoSQL databases allows for better mapping of business objects while facing difficulties regarding transaction consistency (ACID; atomic, consistency, isolation and durability) (Radulović et al. 2016; Ekren and Erkollar 2020; Sokolova et al. 2020). Hence, answering the research question, there is not only one single database model suitable for fulfilling future database requirements.

5.1 A multi-model database approach for enterprise systems

To address existing limitations of databases in the context of enterprise systems, multi-model database approaches allowed to combine the advantages of multiple database systems. Accordingly, their individual specialization can be used to address future business requirements in the domain of enterprise systems. The idea of multi-model database is to support multiple data models in an integrated environment. Against the approach to rely only on a single data model, this allows for more flexible adaption. The concept received increased attention with increasing popularity of NoSQL databases (Garulli 2012).

Our results confirm these findings, i.e. that a broader range of requirements can be addressed by combining multiple database forms in a hybrid architecture. Furthermore, the complementarity of tabular (SQL) and schema-analog (mostly NoSQL) databases has already been explicitly pointed out (Nimis et al. 2014; Radulović et al. 2016; Sokolova et al. 2020). Some NoSQL database systems (e.g., MongoDB) even provide dedicated functionality for collaboration with relational databases and are therefore (technically) suitable for such use (Radulović et al. 2016). The complementarity of tabular and schema-analog databases is also confirmed by our data, as relational databases, while convincing for many traditional requirements, often lag behind with respect to the future requirements highlighted in this paper.

However, the results of this paper further detail this finding and raise the suggestion that a combination of tabular relational databases and schema-analog graph databases may exist an optimal multimodal architecture to meet future requirements (Fig. 3). Indeed, the results show that graph databases have the highest degree of fulfillment among schema-analog databases, except regarding the integration focus. However, since in the latter the relational databases perform optimally, this confirms the previous argument of an appropriate combination of such (Nance et al. 2013). Furthermore, particularly in the context of ERP systems, which process master (a) and transactional data (b), this partitioning of the data could be addressed by a hybrid architecture of relational and graph databases. Master data business objects (e.g., resources, products, human capital, customers, supply chain actors) must capture many different characteristics and behaviors (processes) of business objects. Those differences between business objects could be stored in flexible graph databases. The handling of many different variants (which quickly leads to high complexity in relational databases) could be easily and flexibly realized by a graph

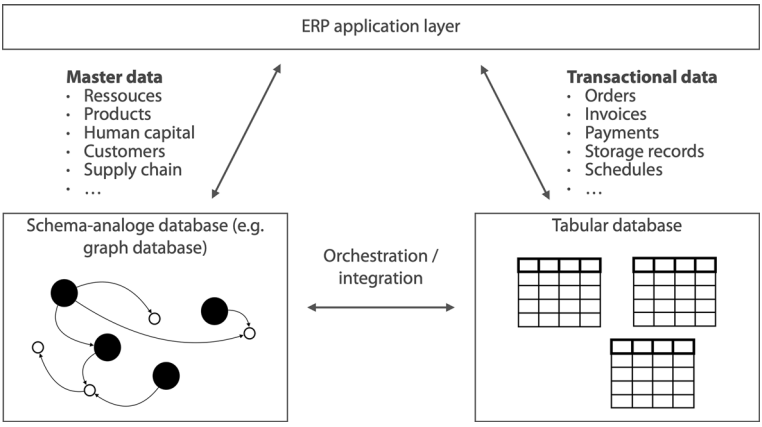


Fig. 3 Separation between schema-analogue and tabular databases

description consisting of edges and nodes. Transactional data (with high requirements for ACID conformity) would continue to be stored in relational databases (e.g., orders, invoices, payments, storage records, schedules). Such a subdivision between transactional and master data or on the basis of structuredness to different databases has already received attention (Bjeladinovic 2018), but is gaining in importance with regard to the future requirements for ERP systems, especially for an implementation from relational and graph databases.

This separation exploits advantages of schema flexibility in graph databases for modeling business objects while ensuring consistency of transactional data in relational databases. For the integration and orchestration of those (tabular and structural-analog) databases, specialized procedures are necessary for synchronization, esp. in terms of uniformity of data records. This could, for example, be executed by virtualization algorithms, that allow for joint (SQL) querying of both the relational as well as the graph database (Lawrence 2014). Moreover, research also proposed concepts for schema transformation and query translation between SQL and NoSQL databases (Dai 2019).

In addition, increasing need for performance and distribution is countered by key-value databases enabling caching of high-frequency process data that may be discarded soon after its processing (“puffer-databases “) (Nimis et al. 2014). This separation between relational and graph databases based on structuredness and purpose of the data (master and transactional data) would allow efficient and robust execution of business processes while accounting for individual characteristics of the processed business objects.

As ERP systems are generally used throughout an organization, they are expected to manage various different business objects. This paper focused on requirement fulfillment of the storage structure of different databases. Here, specific implementations (database management systems; DBMS) were deliberately disregarded as those database management systems generally only supplement surrounding functionality (e.g., interfaces and query languages). The underlaying structure of data

organization can still be assigned to one of the general database types discussed in this paper. Still, besides requirements for the data organization within an ERP database, more general requirements regarding the DBMS arose during the initial assessment. Here, requirements regarding the theme of interdependent complexity, i.e., preventing complexity and efficiently handling incomplete and unstructured data. With increasing heterogeneity of business objects and concepts like Big Data, those requirements are expected to gain even more relevance in the future. DBMS (independent of the underlaying data model) are required to actively prevent complexity and reduce efforts of optimization. Besides that, with hybrid database models (as proposed in this paper), those DBMS must ensure robustness of transactions and processing of data between (different) database models of the hybrid architecture and higher software layers. Particular attention should be paid to aspects of master data quality. Already in (traditional) relational databases, master data maintenance requires a considerable amount of time and effort (Knolmayer and Röthlin 2006; Haug et al. 2013).

5.2 Contributions to research

In this work, current requirements for databases in business systems were collected, which arise in the context of the digital transformation. The work not only offers the definition of generally valid criteria by which it is possible to evaluate databases regarding their performance for future requirements. In this regard, the results provide an enterprise system specific assessment of existing data approaches.

Furthermore, five different types of databases were categorized into tabular and schema-analogue databases regarding their characteristics and evaluated individually. The evaluation enables a functional classification of the databases by concretely pointing out weaknesses and strengths. In addition, the analysis shows which future requirements cannot be mapped in current databases and thus offers innovation impulses for developers. Furthermore, this work offers a new approach to the definition of data partitioning by separating data according to their structure and use (master and transactional data) into relational and graph databases in order to enable an efficient and robust execution of business processes. Following this, a hybrid architecture of these database types is proposed which optimally covers the future requirements for ERP system databases. Regardless of the underlying data model, this paper addresses the increasing relevance of DBMSs. In the long term, DBMS will play an important role in avoiding complexity and dealing efficiently with incomplete and increasingly unstructured data.

5.3 Implications for practice

In total, hybrid (multi-model) architectures that combine relational database models with graph databases could satisfy future requirements through exploitation of individual advantages of those databases. This entails changes both for ERP system developers as well as developers of databases systems. On the one hand, developers of database systems need to adapt to more modular, federated and interoperable

architectures by incorporating priorly mentioned requirements for the databases management system. On the other hand, developers of ERP systems are forced to evaluate different implementations of divergent databases systems and integrate them according to existing software components. The results may only allow to rethink the existing architecture of enterprise systems regarding the database layer. The results allow system providers to adapt to new business requirements. Thereby, system providers are well advised to not rely only rely on standard products for their systems, that is to combine multiple approaches or to consider multi-model databases as the foundation for the future systems generations.

5.4 Future research

For a holistic analysis of the future requirements of enterprise systems, the criteria for evaluating the databases must also be applied to the other levels of the architecture regarding future effects and requirements. In particular, the way in which the various levels will be affected by future requirements and how these will change in the long term must be examined. Hereby, the evaluation criteria should be weighted to allow an adaptation to the different deployment conditions in practical contexts. In this regard, the propositions presented in this paper should be measured regarding their performance under comparable hardware-conditions with multiple systems. So far, the assessment is based on theoretical reasoning originated in academic literature.

Furthermore, future DBMS architecture of information systems should be focused in research. ERP systems have the potential to obtain valuable insights by analyzing a combination of historical data alongside with active process information. This means that to support such hybrid workloads, database management systems need to handle both, fast ACID transactions (OLTP) and complex analytical queries (OLAP). The paper has shown that existing database technologies for enterprise systems are not well-suited to include process structures in their data model and that real time ability is also missing. This might be a grand challenge to overcome these obstacles so that enterprise systems can fully meet the demands of the Internet of Things, the 4th Industrial Revolution, and Digital Transformation. As a solution approach, the concept of a hybrid database system of two or more heterogeneous databases, which would be capable of decomposing and executing queries on transactional and master data, was proposed. At this point, future research should specifically examine and operationalize the different strengths and weaknesses of each database for the different application requirements of master- and transactional data. Since different types of databases have strengths and weaknesses, they should be evaluated depending on the characteristics of the data and the types of queries applied. First publications already provide an overview of previous research on hybrid database models (Vyawahare et al. 2018) and highlight specific differences between relational and graph-oriented databases (Arulraj et al. 2016). These results should be transferred to the proposed hybrid storage system of transactional and master data and furthermore tested for a concrete use case. As a hybrid system aims to unify the strengths of a database by eliminating the limitations of another, the

advantages of combining different types of storage need to be explored. Unknown to this point is, which changes this hybrid database architecture demands from other levels of the enterprise systems architecture.

5.5 Limitations

In this paper, the focus was set on a few database concepts that are commonly used in practice and not all available database concepts were covered. Furthermore, the survey results that are the basis of the research are neither fully representative nor exhaustive because the research's emphasis lied on recent issues. The same applies for the problems with the current ERP system generation that were drawn from literature. Regarding the approach for the collection of the requirements and the database assessment several limitations need to be named. First of all, the requirements stem from a focus group of ERP experts and the assessment of the database was conducted only conceptually by the same focus group. The underlying operationalizations are based on assumptions and theoretical reasoning. An actual test of the interaction between an ERP system in use and the various databases, taking into account the requirements, did not take place. To address the subjectivity in this matter, the assessment was done by every expert individually in a first round and a consensus was reached in a second round. Also, the focus of this research lied exclusively on the database concept, while some issues might be addressed by the respective DBMS. However, DBMS were deliberately excluded. Finally, this research only addresses requirements that consider one layer of the ERP architecture and it is potentially short-sighted to consider some of these requirements only at the database level.

6 Conclusion

This paper addressed the question which database models are suitable to meet future requirements for ERP systems. The paper follows the assumption that the relational concept may be outdated for future business requirements. Given that many restrictions ERP pose on their adaptability are related to the standardization of data, the database layer of ERP systems is addressed. Databases serve as the foundation for data storage and retrieval. As such, they limit the flexibility of enterprise systems and the chance to adapt to new requirements accordingly. This supports the aforementioned assumption and underlines the relevance of this research. Digital transformation sets new requirements for the future of ERP systems and its databases that should be taken into account when addressing the problems that are currently prevalent. To identify these requirements and answer the research question a qualitative approach was chosen. A focus group was conducted with five enterprise system researcher to specify relevant requirements for ERP databases following the framework for requirements in the twenty-first century by Hansen et al. (2009) in an iterative process until a common understanding of the individual requirements was reached. All requirements are related to one of the following categories:

business process focus, centrality of architecture, distributed requirements, fluidity of designs, interdependent complexity, integration focus and layers of requirement. This list of requirements forms the basis for the evaluation of different databases that was finally conducted by the same focus group. In this paper, related literature on databases that are relevant in the realm of ERP systems is presented to gain a theoretical understanding of the complexity of the research question. The result of this paper is the conceptual evaluation and discussion of the suitability of six databases for ERP systems. Hybrid database architectures combining relational databases and graph databases can be used to fulfill future requirements for ERP systems. This research forms the basis for renewal of the current generation of ERP systems and proposes to ERP vendors to use different database concepts in the future.

Appendix 1

Theme	Requirement	Understanding
Business process focus	Scalability	Scalability is assessed as the ability to increase the capacity (volume) of the database by adding new data objects to the existing ones
Business process focus	Performance (OLAP)	OLAP (Online Analytical Processing) is used to support decision-making by enabling multidimensional analysis of historical, consolidated and integrated data through complex read-only queries. To facilitate this type of analysis, data is usually organized into data cubes that are categorized by dimensions (e.g., customers, geographic sales region and time period) and stored with characteristics. To build metrics, OLAP locates the intersection of the dimensions and calculates, for example, how products of a certain region were sold over a certain price in a certain time period. Especially in the ERP context, there is a need for reports with condensed values or running scenarios based on real actual and planned figures through what-if analyses. In the following, the OLAP capability of the databases is evaluated

Theme	Requirement	Understanding
Business process focus	Performance (OLTP)—Performing a task on an individual data set including several fields	OLTP (Online Transaction Processing) serves the operational business by enabling immediate data retrieval or manipulation of current, detailed or isolated data through short, simple transactions. The access pattern is based on read/write/update operations; the index/hash is on the primary key. The basis of the following evaluation proceeds on the assumption that a modification of a single data set has to be undertaken
Business process focus	Performance (OLTP)—Performing a task across many data sets	As a complementary approach to OLTP consideration, the following section evaluates the performance of a transaction across many data sets
Business process focus	Real-time capability	Under real-time capability, a response time of less than 50 ms is expected
Business process focus	Depiction of a digital twin	A digital twin is understood to be the possibility of accessing (additional) information in digital form about a physical object or person. The goal of a digital twin is the detailed mapping of a product over its entire life cycle. In terms of databases, the ability to map a digital shadow for every product, every capacity and every resource of a company is assessed
Business process focus	Heterogeneity of business objects	Under heterogeneity of business objects, the selected databases are evaluated based on their ability to individually map heterogeneous instances of business objects
Centrality of architecture	Modularity	Modularity in databases is characterized by the fact that adjustments or a selection of database modules can be made by changing database definitions. In the following, the ability to divide a database schema into further modules is evaluated
Centrality of architecture	Mapping of the process structure through data management	In the following, the ability to align or adapt data management along business processes is evaluated
Distributed requirements	Mapping of data protection in data model	The aim of data protection is to prevent data misuse by protecting confidential data from access by unauthorized persons. In the following, the ability to map data protection in the data model is evaluated

Theme	Requirement	Understanding
Distributed requirements	Mapping of compliance requirements in the data model	The GDPR presents companies with the technical and organizational challenge of a holistic data governance strategy. In the following, the ability to map compliance requirements in the data model is evaluated
Distributed requirements	Mapping of cooperation suitability in the data model	A key driver of value creation in the context of digitization is the linking of processes and data at both intra- and interorganizational levels. In this way, supply chains can be more closely interlinked so that they fit precisely into the production process or the recipient can receive them directly. When exchanging data with partners, however, the cooperation suitability of the shareable data must be evaluated and identified. In the following, the mapping of cooperation suitability in the data model is evaluated
Distributed requirements	Mapping of data security in relation to stakeholders in the data model	Data security is about preventing data from being damaged or lost. In the following, the ability to map data security in relation to stakeholders in the data model is evaluated
Fluidity of design	Allow data models to change over time	Agility of a database means that the underlying data model can be adapted to wishes and specific circumstances without major changes and can thus be adapted to existing or changing organizational circumstances. This capability is assessed in the following
Fluidity of design	Allow data models to change during runtime	In the following, the ability to make changes to an existing data model during runtime is evaluated

Theme	Requirement	Understanding
Interdependent complexity	Preventing additional complexity	The continuous growth of data requires database systems to prevent additional complexity. The complexity of a database is hereby defined as the ratio of keys (primary and foreign keys) and indexes to the total number of attributes. The more attributes of a database are part of primary and foreign keys and the more attributes are indexed, the higher the complexity in dealing with database systems. The lower the complexity, the more short-term and uncomplicated the data access. In the following, the selected database types are evaluated with regard to their complexity
Interdependent complexity	Structure formation for unstructured data	Unstructured data is information that exists in a non-identifiable and non-normalized data structure. Unstructured data does not allow any conclusions to be drawn about the content, only about the data type. In the following, the ability to handle unstructured data is evaluated
Interdependent complexity	Incomplete data	Unless data is obtained in a controlled or experimental environment, general data collection is regularly accompanied by the problem of missing values. Databases must therefore be able to handle incomplete data. In the following, the ability to handle incomplete data is evaluated
Integration focus	Support of the extract process (ETL process)	Extraction is the first step of the ETL process. The extraction of data describes the reading of the selected data from the source systems. In the case of complete extractions, the relevant objects are unloaded in their entirety from the underlying operational database systems. In the following, the ability to support the extract process is evaluated

Theme	Requirement	Understanding
Integration focus	Support of the transform process (ETL process)	The extraction is followed by the transformation phase. The delivered data is adapted to the format and schema of the target database. In the simplest case, values can be taken over directly. If this is not possible, appropriate transformations must be performed. These transformations can handle schema conflicts, among other things. Since data often originates from heterogeneous data sources, these must first be formatted to a uniform format. This includes, among other things, the adjustment of data types (e.g. format of storage), conversion of encodings (e.g. "w" from German female to "f" for English female), standardization of character strings (e.g. capital letters), standardization of dates (e.g. numerical representation of the date from one to another), standardization of the format of the data (e.g. "f" for female), standardization of the format of the data (e.g. "w" from female to "f" for female), conversions of measurement units (e.g. cm to mm) or combining and separating attributes (e.g. splitting "first name last name" into "first name" and "last name"). In the following, the support of the Transform process is evaluated
Integration focus	Scheme interoperability	The interoperability of database schemas presupposes the existence of schemas. By identifying individual components of a database and determining their relationships to each other, translation and integration can occur towards integration. An integration mechanism is created by mapping one schema onto another schema. In the following, the schema interoperability capability of the selected databases is evaluated
Layers of requirement	Granular security settings	For each database, it is necessary to evaluate how granular the level of detail of the security objects to be protected can be. This can range from the entire database as a single security object, to classes or relations, to data objects or individual attributes as security objects

Theme	Requirement	Understanding
Layers of requirement	Comprehensibility of the data model	The comprehensibility of a data model means that every user (regardless of department) can understand the content of a data model. The aim of the data model is to represent a complex reality in a technically correct, yet clear and comprehensible way. In the following, we evaluate the comprehensibility of the data model of the selected databases

Appendix 2

Requirement	Definition	Measurement	Level 0	Level 1	Level 2
Scalability	Scalability is the ability to scale up and scale down capacity based on demanded workload (Herbst et al. 2013)	Degree of expandability of the database in order to incrase or reduce capacity	Scalability is limited	As the amount of data increases, the scaling becomes more complex (related to interdependencies)	The capacity of the database can be increased or decreased depending on the work requirement
Performance (OLAP)	The approach of OLAP (online analytical processing) is a composition of analytical queries over data-cubes, that allow to view multidimensional data models from different perspective (Kovacic et al. 2022)	Ability to perform analytical queries	Analytical queries cannot be performed	Analytical queries can be performed	Analytical queries can be combined in any way

Requirement	Definition	Measurement	Level 0	Level 1	Level 2
Performance (OLTP)	Data in OLTP systems is organized according to the relation model, i.e. data is highly normalized in order to ensure consistency and to run day-to-day operations on these systems (Schaffner et al. 2009)	Ability to efficiently record business transactions	A high level of redundancy increases the risk of data inconsistencies and update dependencies	Business transactions can be recorded	Additional structures for optimized query performance like indexes, materialized views, or precomputed aggregates enable efficient recording of business transactions
Performance (OLTP multiple datasets)	On-Line Transaction Processing (OLTP) applications demand rapid, interactive processing for large numbers of relatively simple transactions (Thakkar and Sweiger 1990)	Rapid processing over a multiple number of datasets	Slow processing over multiple datasets	Performance decreases with number of transactions	High performance for large numbers of transactions
Real-time capability	Real-time systems are characterised by the need to meet both functional requirements and timing constraints, typically expressed in terms of deadlines (Maiza et al. 2020)	Fulfillment of narrow timing constraints	No consideration of timing constraints	Consideration only of timing constraints down to seconds	Consideration of timing constraints down to milliseconds

Requirement	Definition	Measurement	Level 0	Level 1	Level 2
Digital twin	DTs can be defined as (physical and/or virtual) machines or computer-based models that are simulating, emulating or mirroring the properties and behavior of a physical entity (Barricelli et al. 2019)	Degree of mirroring, simulating or emulating	No mirroring, simulating or emulating	Mirroring, simulating or emulating the static properties of a physical entity but not the behavior	Mirroring, simulating or emulating both the static properties of a physical entity and the behavior
Business objects heterogeneity	Individualization allows the customer to influence characteristic features of the product by choosing from a variety of pre-defined and custom-defined modules (Schaede et al. 2018)	Degree, to which product features can be individualized by the customer	No individualization (mass production)	System reflects customer's choices of characteristic features of the product by choosing from a variety of pre-defined modules (mass customization)	System reflects customer's choices of characteristic features of the product by choosing from a variety of pre-defined and custom-defined modules (individualization)
Extract step (ETL)	The extraction of the appropriate data from the sources (Vassiliadis 2009)	Degree, to which product features support data extraction process	No support of the extract process	Basic features for the extract process	Advanced features for the extract process
Transform step (ETL)	Transformation of source data and the computation of new values (and, possibly records) in order to obey the [target] structure (Vassiliadis 2009)	Degree, to which product features support data transformation process	No support of the transform process	Basic features for the transform process	Advanced features for the transform process

Requirement	Definition	Measurement	Level 0	Level 1	Level 2
Schema interoperability	Semantic connections between data that are not related concerning the execution code is made obvious by documenting the conceptual model underlying the component. (Tolk and Wang 2009)	Degree, to which semantic data connections can be retrieved from the data model	Semantic connections can only be retrieved from source code	Some semantic connections are represented in the data model	Semantic connections are represented in the data model
Data protection in data model	The data model should provide information about the data in terms of privacy protection (Dashrat 2014)	Characterization of elements of the data model on a meta-level	Not possible to define data protection relevance	Possible to define data protection relevance for groups of objects	Possible to define data protection relevance for individual objects
Compliance in data model	The data model should provide information about the data in terms of compliance (Dashrath 2014)	Characterization of elements of the data model on a meta-level	Not possible to mark up elements	Possible to mark up elements during build-time	Possible to mark up elements during runtime
Cooperation suitability in data model	The data model should provide information about the suitability of the data in terms of transferability and joint processing (Dashrath 2014)	Characterization of elements of the data model on a meta-level	Not possible to define collaboration eligibility	Possible to define collaboration eligibility for groups of objects	Possible to define collaboration eligibility for individual objects

Requirement	Definition	Measurement	Level 0	Level 1	Level 2
Data security in data model	The data model should provide information about the data in terms of safety significance (Dashrath 2014)	Characterization of elements of the data model on a meta-level	Not possible to define data security	Possible to define data security for groups of objects	Possible to define data security for individual objects
Granular security settings	Database security is concerned with ensuring the secrecy, integrity, and availability of data stored in a database. To define the terms, secrecy denotes the protection of information from unauthorized disclosure either by direct retrieval or by indirect logical inference. (Pernul 1994)	Degree, to which individual data security settings can be set for entries	No support of database security or only on full database configurable	Database security can be configured on parts of the database	Database security can be configured on small segments of the database individually
Comprehensibility of data model	Comprehensibility is understood as the ability of the various stakeholders to understand relevant aspects of the data model (Gleicher 2016)	Ability to recognize data model logic without further support	Data model is only comprehensible for the creator or database experts	Data model is comprehensible for at least database designers	Data model is comprehensible for software users (e.g. by visualization and explanations)

Requirement	Definition	Measurement	Level 0	Level 1	Level 2
Modularity	Modularity is the degree to which the components of a system can be separated and recombined (Schilling 2000)	Number of recombination possibilities of the system's modules	No recombination	Limited ability for recombination	Huge ability for recombination
Process structure mapping	A structural analogy between process and system can be cast as a mapping of predicates and terms that play the same role in both descriptions (Leuzzi and Ferilli 2018)	Number of mappable predicates and terms between process and system	No mappable items	Limited number of items with the same role that can be matched between process description and data model	All items with the same role can be matched between process description and data model
Preventing additional complexity	The database should reduce the complexity of the data—firstly in terms of the structure of the data itself, but also in terms of the effort required by consumers and developers (Kearney et al. 1986)	Ability and degree of complexity reduction	Not possible to reduce the complexity of the data	Possible to reduce the complexity of processing	Possible to reduce the complexity of the actual structure of the data as well as of the processing effort
Structure formation for unstructured data	Unstructured data does not come with a data model that enables a computer to use them directly (Kiefer 2016)	Degree to which data approach allows to identify data structure itself	No data model within the data, structure varies widely	Data model is not made explicit, but data is structured in a coherent ways	Data is structured logically and data model is present / given

Requirement	Definition	Measurement	Level 0	Level 1	Level 2
Incomplete data	Incomplete data are [cases] in which certain features are missing from particular feature vectors (Williams et al. 2007)	Ability to handle incomplete data on the database level for subsequent applications	The majority of entries considered a specific interest in data is missing	Some entries considered a specific interest in data is missing	No entries are missing considering a specific interest in data
Data model changes over time	The lifetime of the database should be extendable by adapting its structure to changing conditions (Engel and Browning 2008; Yoder et al. 2001)	Flexibility of the data organization within the data model in terms of the effort required to make changes	Not possible to change the data model without affecting the existing data	Possible to change the data model during build-time without affecting existing data	Possible to change the data model during run-time without affecting existing data
Data model changes during runtime	The data model should be changeable during runtime to make effects immediately available without requiring downtime of the system (Yoder et al. 2001)	Degree to which structural changes to the data organization can take place during operation	Not possible to change the data model at run-time	Possible to extend the data model during run-time	Possible to extend and resize the data model during run-time

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Abadi D, Boncz P, Harizopoulos S (2013) The design and implementation of modern column-oriented database systems. Now Publishers Inc., Hanover, MA, USA

Abadi DJ, Madden SR, Hachem N (2008) Column-stores vs. row-stores: how different are they really. pp 967–980

Abadi DJ, Boncz PA, Harizopoulos S (2009) Column-Oriented Database Syst Proc VLDB Endow 2:1664–1665

Abd Elmonem MA, Nasr ES, Geith MH (2016) Benefits and challenges of cloud ERP systems–A systematic literature review. Future Comput Inf J 1:1–9. https://doi.org/10.1016/j.fcij.2017.03.003

Alberts I (2013) Challenges of information system use by knowledge workers the email productivity paradox. Proc Assoc Inf Sci Technol 50:1–10. https://doi.org/10.1002/meet.14505001089

Alocci D, Mariethoz J, Horlacher O et al (2015) Property graph vs RDF triple store:a comparison on glycan substructure search. PLoS ONE 10:e0144578

Angles R, Gutierrez C (2008) Survey of graph database models. ACM Comput Surv (CSUR) 40:1–39

Arenas M, Libkin L (2004) A normal form for XML documents. ACM Trans Database Syst (TODS) 29:195–232

Arnold J, glavic B, Raicu I (2019) A High-performance distributed relational database system for scalable olap Processing. IEEE, pp 738–748

Arulraj J, Pavlo A, Menon P (2016) Bridging the archipelago between row-stores and column-stores for hybrid workloads. pp 583–598

Atikoglu B, Xu Y, Frachtenberg E et al (2012) Workload analysis of a large-scale key-value store. pp 53–64

Atkinson M, DeWitt D, Maier D et al (1990) The Object-oriented database system manifesto. In: KIM W, NICOLAS J-M, NISHIO S (eds) Deductive and object-oriented databases. north-holland, amsterdam, pp 223–240

Balmin A, Papakonstantinou Y, Vianu V (2004) Incremental validation of XML documents. ACM Trans Database Syst (TODS) 29:710–751

Barricelli BR, Casiraghi E, Fogli D (2019) A survey on digital twin: definitions, characteristics, applications, and design implications. IEEE Access 7:167653–167671. https://doi.org/10.1109/ACCESS.2019.2953499

Bender B, Bertheau C, Gronau N (2021) Future ERP Systems: a Research Agenda: In: Proceedings of the 23rd International conference on enterprise information systems. SCITEPRESS - Science and technology publications, online streaming, pp 776–783

Benymol J, Abraham S (2020) Performance analysis of nosql and relational databases with mongodb and mysql. Mater Today Proc 24:2036–2043. https://doi.org/10.1016/j.matpr.2020.03.634

Bertino E, Martino L (1991) Object-oriented database management systems. Concept Issues Comput 24:33–47

Bjeladinovic S (2018) A fresh approach for hybrid SQL/NoSQL database design based on data structuredness. Enterp Inf Syst 12:1–19. https://doi.org/10.1080/17517575.2018.1446102

Brakatsoulas S, Pfoser D, Tryfona N (2004) Modeling, storing and mining moving object databases. IEEE, pp 68–77

Candel CJF, Ruiz DS, García-Molina JJ (2022) A unified metamodel for NoSQL and relational databases. Inf Syst. https://doi.org/10.1016/j.is.2021.101898

Cao L, Zhu H (2013) Normal accidents: data quality problems in ERP-enabled manufacturing. J Data and Inf Qual (JDIQ) 4:1–26. https://doi.org/10.1145/2458517.2458519

Chebotko A, Abraham J, Brazier P, et al (2013) Storing, indexing and querying large provenance data sets as RDF graphs in apache HBase. IEEE, pp 1–8

Chen L, Dai W, Qiu M, Jiang N (2017). a Design for Scalable and Secure Key-Value Stores. In 2017 IEEE International Conference on Smart Cloud (SmartCloud) (pp. 216-221). IEEE.

Chen J, Song Q, Zhao C, Li Z (2020) Graph database and relational database performance comparison on a transportation network. In: Singh M, Gupta PK, Tyagi V et al (eds) Advances in computing and data sciences. springer singapore, singapore, pp 407–418

Chien S-Y, Tsotras VJ, Zaniolo C, Zhang D (2001) Storing and querying multiversion XML documents using durable node numbers. IEEE, pp 232–241

Chung SM, Jesurajaiah SB (2005) Schemaless xml document management in object-oriented databases. IEEE, pp 261–266

Ciglan M, Averbuch A, Hluchy L (2012) Benchmarking traversal operations over graph databases. IEEE, pp 186–189
Comyn-Wattiau I.,Akoka J (2017December) Model driven reverse engineering of NoSQL property graph databases:The case of Neo4j (pp 453–458) IEEE
J Dai (2019) SQL to NoSQL: What to do and How. IOP Publishing, p 012080
Davoudian A, Chen L, Liu M (2018) A survey on NoSQL stores. ACM Comput Surv (CSUR) 51:1–43
Dobaj J, Iber J, Krisper M, Kreiner C (2018) A microservice architecture for the industrial Internet-of-Things. In: Proceedings of the 23rd European Conference on Pattern Languages of Programs. pp 1–15
Ekren G, Erkollar A (2020) The potential and capabilities of nosql databases for erp systems: In: Ekren G, Erkollar A, Oberer B (eds) Advanced mis and digital transformation for increased creativity and innovation in business. IGI Global, pp 147–168. https://doi.org/10.4018/978-1-5225-9550-2.ch007
El Kadiri S, Grabot B, Thoben K-D et al (2016) Current trends on ICT technologies for enterprise information systems. Comput Ind 79:14–33. https://doi.org/10.1016/j.compind.2015.06.008
Elmasri R, Navathe S (2007) Fundamentals of database systems, 5th edn. Pearson/Addison Wesley, Boston
Engel A, Browning TR (2008) Designing systems for adaptability by means of architecture options. Syst Eng 11(2):125–146. https://doi.org/10.1002/sys.20090
Finch H, Lewis J, Turley C (2003) Focus groups. A guide for social science students and researchers, Qualitative research practice, pp 170–198
Fong J, Shiu H, Yeung YF (2010) Concurrent data materialization for xml-enabled database with semantic metadata. Int J Software Eng Knowl Eng 20:377–422
Frick NRJ, Brünker F, Ross B, Stieglitz S (2019) Towards Successful Collaboration: Design Guidelines for AI-based Services enriching Information Systems in Organisations. Perth (Australia)
Garulli L (2012) OrientDB. Orient Technologies [Online]. http://www.orientdb.org/luca-garulli.htm/
Gessert F, Wingerath W, Friedrich S, Ritter N (2017) NoSQL database systems: a survey and decision guidance. Comput Sci-Res Dev 32:353–365
Gleicher M (2016) A framework for considering comprehensibility in modeling. Big Data 4:75–88. https://doi.org/10.1089/big.2016.0007
Gronau N (2021) ERP-Systeme - Architektur, Management und Funktionen des Enterprise Resource Planning. De Gruyter Oldenbourg. https://doi.org/10.1515/9783110663396
Hansen S, Berente N, Lyytinen K (2009) Requirements in the 21st century: current practice and emerging trends. In: Lyytinen K, Loucopoulos P, Mylopoulos J, Robinson B (eds) Design Requirements engineering: a ten-year perspective. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 44–87
Harding PJ, Li Q, Moon B (2003) XISS/R: XML indexing and storage system using RDBMS. Elsevier, pp 1073–1076
Haug A, Stentoft J, Zachariassen F, Schlichter J (2013) Master data quality barriers: an empirical investigation. Ind Manag Data Syst 113:243–249. https://doi.org/10.1108/02635571311303550
He H, Singh AK (2008) Graphs-at-a-time: query language and access methods for graph databases. pp 405–418
Herbst N, Kounev S, Reussner RH (2013) *Elasticity in Cloud Computing: What It Is, and What It Is Not*. 23–27
Huynh TN, Mangisengi O, Tjoa AM (2000) Metadata for object-relational data warehouse. p 3
Irmert F, Daum M, Meyer-Wegener K (2008) A new approach to modular database systems. pp 40–44
Kanade AS, Gopal A (2013) Choosing right database system: row or column-store. IEEE, pp 16–20
Kearney JP, Sedlmeyer RL, Thompson WB, Gray MA, Adler MA (1986) Software complexity measurement. Commun ACM 29(11):1044–1050
Kiefer C (2016). assessing the Quality of Unstructured Data: An Initial Overview. 62–73
Knolmayer GF, Röthlin M (2006) Quality of material master data and its effect on the usefulness of distributed erp systems. In: Roddick JF, Benjamins VR, Si-said Cherfi S et al (eds) Advances in conceptual modeling - theory and practice. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 362–371
Koh L, Gunasekaran A, Goodman T (2011) Drivers, barriers and critical success factors for ERPII implementation in supply chains:a critical analysis. J Strateg Inf Syst 20:385–402. https://doi.org/10.1016/j.jsis.2011.07.001
Kolahi S (2007) Dependency-preserving normalization of relational and xml data. J Comput Syst Sci 73:636–647

Kovacic I, Schuetz CG, Neumayr B, Schrefl M (2022) OLAP Patterns: a pattern-based approach to multidimensional data analysis. Data Knowl Eng 138:101948. https://doi.org/10.1016/j.datak.2021.101948

Lapalme J, Gerber A, Van der Merwe A et al (2016) Exploring the future of enterprise architecture: a zachman perspective. Comput Ind 79:103–113. https://doi.org/10.1016/j.compind.2015.06.010

Lawrence R (2014) Integration and virtualization of relational SQL and NoSQL systems including MySQL and MongoDB. IEEE, pp 285–290

Leinders D, Van den Bussche J (2007) On the complexity of division and set joins in the relational algebra. J Comput Syst Sci 73:538–549

Leuzzi F, Ferilli S (2018) A multi-strategy approach to structural analogy making. J Intell Inf Syst 50(1):1–28. https://doi.org/10.1007/s10844-017-0447-6

Li Y, Manoharan S (2013) A performance comparison of SQL and NoSQL databases. IEEE, pp 15–19

Liu X, Hu C, Li Y, Jia L (2014) The advanced data service architecture for modern enterprise information system. IEEE

Lorenz M (2015) The impact of column-orientation on the quality of class inheritance mapping specifications. IEEE, pp 597–597

Lufter J (1999) Objektrelationale datenbanksysteme - aktuelles schlagwort. Inform Spektrum 22:288–290. https://doi.org/10.1007/s002870050146

Mahnke W, Steiert HP (2000) To a man with an ORDBMS everything looks like a row in a table. In: Proceedings of the Third International Symposium on Cooperative Database Systems for Advanced Applications. CODAS 2001. IEEE Comput. Soc, Beijing, China, pp 65–71

Mai ST, Amer-Yahia S, Assent I et al (2018) Scalable interactive dynamic graph clustering on multicore CPUs. IEEE Trans Knowl Data Eng 31:1239–1252

Maiza C, Rihani H, Rivas JM, Goossens J, Altmeyer S, Davis RI (2020) A survey of timing verification techniques for multi-core real-time systems. ACM Comput Surv 52(3):1–38. https://doi.org/10.1145/3323212

Mei F, Cao Q, Jiang H, Tian L (2018) LSM-tree managed storage for large-scale key-value store. IEEE Trans Parallel Distrib Syst 30:400–414

Miklau G, Levine BN, Stahlberg P (2007) Securing history: Privacy and accountability in database systems. Citeseer, pp 387–396

Mo Y, Ling TW (2002) Storing and maintaining semistructured data efficiently in an object-relational database. IEEE, pp 247–256

Mondal J, Deshpande A (2012) Managing large dynamic graphs efficiently. pp 145–156

Morris H, Liao H, Padmanabhan S, et al (2008) Bringing Business Objects into Extract-Transform-Load (ETL) Technology. IEEE, pp 709–714

Naedele M (2003) Standards for XML and Web services security. Computer 36:96–98

Nance C, Losser T, Iype R, Harmon G (2013) NOSQL VS RDBMS - Why There Is Room For Both. SAIS 2013 Proceedings 27 7

Nimis J, Armbruster M, Kammerer M (2014) Zukunftsfähiges datenmanagement durch hybride lösungen – ein entwurfsmusterkatalog zur integration von sql- und nosql-datenbanken. In: jähnert j, förster c (eds) technologien für digitale innovationen: interdisziplinäre beiträge zur informationsverarbeitung. springer fachmedien wiesbaden, Wiesbaden, pp 19–42

O'Brien T (2015) 'Accounting' for data quality in enterprise systems. Procedia Comput Sci 64:442–449. https://doi.org/10.1016/j.procs.2015.08.539

Panetto H, Zdravkovic M, Jardim-Goncalves R, Romero D (2015) New perspectives for the future interoperable enterprise systems. Comput Ind 79:47–63. https://doi.org/10.1016/j.compind.2015.08.001

Panetto H, Zdravkovic M, Jardim-Goncalves R et al (2016) New perspectives for the future interoperable enterprise systems. Comput Ind 79:47–63. https://doi.org/10.1016/j.compind.2015.08.001

Paré G, Trudel M-C, Jaana M, Kitsiou S (2015) Synthesizing information systems knowledge: a typology of literature reviews. Inf Manage 52:183–199. https://doi.org/10.1016/j.im.2014.08.008

Parent C, Spaccapietra S, Zimányi E (2009) modularity in databases. In: stuckenschmidt H, parent C, spaccapietra S (eds) modular ontologies. springer berlin heidelberg, berlin, heidelberg, pp 113–153. https://doi.org/10.1007/978-3-642-01907-4_6

Pernul G (1994). Database Security. In: advances in Computers (Vol. 38, pp. 1–72). Elsevier. https://doi.org/10.1016/S0065-2458(08)60175-8

Plattner H (2009) A common database approach for OLTP and OLAP using an in-memory column database. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. Association for Computing Machinery, New York, NY, USA, pp 1–2

Pokorný J,(2015) Graph databases: their power and limitations. Springer, pp 58–69
Politou E, Michota A, Alepis E et al (2018) Backups and the right to be forgotten in the GDPR: an uneasy relationship. Comput Law & Secur Rev 34(1247):1257. https://doi.org/10.1016/j.clsr.2018.08.006
Radulović B, Radosav D, Malić M (2016) The application of nosql mongodb in developing the epr system for managing human resources. Int'1 J Comput, Commun Instrum Eng (IJCCIE) 3:2349–1469
Remenyi D, Williams B, Money A, Swartz E (1998) Doing research in business and management: an introduction to process and method. Sage
Robinson I, Webber J, Eifrem E (2015) Graph databases: new opportunities for connected data. O'Reilly Media, Inc.
Romero D, Vernadat F (2016a) Enterprise information systems state of the art: past, present and future trends. Comput Ind 79:3–13. https://doi.org/10.1016/j.compind.2016.03.001
Romero D, Vernadat F (2016b) Future perspectives on next generation enterprise information systems. Comput Ind. https://doi.org/10.1016/j.compind.2016.02.001
Rudolf M, Paradies M, Bornhövd C, Lehner W (2013) The graph story of the sap hana database. datenbanksysteme für business, technologie und web (btw), p 2037
Sadalage PJ, Fowler M (2013) NoSQL distilled: a brief guide to the emerging world of polyglot persistence. Pearson Education
Saldaña J (2021) The coding manual for qualitative researchers. SAGE Publications Ltd, Arizona State University, USA
Schaede C, Seifermann S, Metternich J (2018) Automated generation of CNC programs for manufacturing of individualized products. Procedia CIRP 72:1251–1257. https://doi.org/10.1016/j.procir.2018.03.064
Schaffner J, Bog A, Krüger J, Zeier A (2009) A hybrid row-column oltp database architecture for operational reporting. In: Castellanos M, Dayal U, Sellis T (eds) business intelligence for the real-time enterprise: second international workshop, birte 2008, auckland, new zealand, august 24, 2008, revised selected papers. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 61–74. https://doi.org/10.1007/978-3-642-03422-0_5
Schilling MA (2000) Toward a general modular systems theory and its application to interfirm product modularity. Acad Manag Rev 25(2):312. https://doi.org/10.2307/259016
D Seybold, J Domaschka (2017) Is distributed database evaluation cloud-ready Springer pp 100–108
Shapiro M, Miller E (1999) Managing databases with binary large objects. IEEE, pp 185.193
Sinha D, Roy R (2020) Reviewing cyber-physical system as a part of smart factory in industry 4.0. IEEE Eng Manage Rev 48:103–117. https://doi.org/10.1109/EMR.2020.2992606
Sokolova MV, Gómez FJ, Borisoglebskaya LN (2020) Migration from an SQL to a hybrid SQL/NoSQL data model. J Manage Anal 7:1–11. https://doi.org/10.1080/23270012.2019.1700401
Sridhar K, Johnson J (2018) Entropy aware adaptive compression for SQL column stores. Springer pp 90–104
Tavana M, Hajipour V, Oveisi S (2020) IoT-based enterprise resource planning: challenges, open issues, applications, architecture, and future research directions. Internet of Things. https://doi.org/10.1016/j.iot.2020.100262
Thakkar SS, Sweiger M (1990) Performance of an OLTP application on symmetry multiprocessor system. Proceedings of the 17th Annual International symposium on computer architecture - ISCA '90, 228–238. https://doi.org/10.1145/325164.325149
Thalheim B (2000) Entity-relationship modeling. springer berlin heidelberg, berlin, heidelberg. https://doi.org/10.1007/978-3-662-04058-4
Thomson A, Diamond T, Weng S-C et al (2014) Fast distributed transactions and strongly consistent replication for OLTP database systems. ACM Trans Database Syst (TODS) 39:1–39
Tolk A, Wang W (2009). The levels of conceptual interoperability model: applying systems engineering principles to M&S. pp 1–9.
Tongkaw S, Tongkaw A (2016) A comparison of database performance of MariaDB and MySQL with OLTP workload. IEEE, pp 117–119
Torres A, Galante R, Pimenta MS, Martins JBA (2017) Twenty years of object-relational mapping: a survey on patterns, solutions, and their implications on application design. Inf and Softw Technol 82:1–18. https://doi.org/10.1016/j.infsof.2016.09.009
ur Rehman, Yaqoob, Salah MHIK et al (2019) The role of big data analytics in industrial internet of things. Futur Gener Comput Syst 99:247–259
Vassiliadis P (2009) A survey of extract transform load technology. Inter J Data Warehouse Min 5(3):1–27. https://doi.org/10.4018/jdwm.2009070101

De Virgilio R, Maccioni A, Torlone R (2014) Model-driven design of graph databases. In: Eric Yu, Dobbie G, Jarke M, Purao S (eds) Conceptual modeling. Springer Int Publishing, Cham, pp 172–185. https://doi.org/10.1007/978-3-319-12206-9_14

vom Brocke J, Maaß W, Buxmann P et al (2018) Future work and enterprise systems. Bus Inf Syst Eng 60:357–366. https://doi.org/10.1007/s12599-018-0544-2

Vyawahare H, Karde PP, Thakare VM (2018) A hybrid database approach using graph and relational database. IEEE, pp 1–4

Wang Y, Wu S, Mao R (2020) Towards read-intensive key-value stores with tidal structure based on lsm-tree. IEEE, pp 307–312

Webster J, Watson RT (2002) analyzing the past to prepare for the future: writing a literature review. mis quarterly 26:xiii–xxiii. 10. 1.1.104.6570

Weichhart G, Molina A, Chen D, Whitman LE, Vernadat F (2016) Challenges and current developments for sensing, smart and sustainable enterprise systems. Comput Ind 79:34–46. https://doi.org/10.1016/j.compind.2015.07.002

Williams D, Liao X, Xue Y, Carin L, Krishnapuram B (2007) On classification with incomplete data. IEEE Trans Pattern Anal Mach Intell 29(3):427–436. https://doi.org/10.1109/TPAMI.2007.52

W Xu, Z Feng, Lo E (2016) Fast multi-column sorting in main-memory column-stores. pp 1263–1278

Yoder JW, Balaguer F, Johnson R (2001) Architecture and design of adaptive object-models. ACM SIGPLAN Notices 36(12):50–60. https://doi.org/10.1145/583960.583966

Yoon M, Jung J, Kang U (2018) Tpa: Fast, scalable, and accurate method for approximate random walk with restart on billion scale graphs. IEEE, pp 1132–1143

Zaniolo C (1984) Database relations with null values. J Comput Syst Sci 28:142–166

Authors and Affiliations

Benedict Bender[1] · Clementine Bertheau[1] · Tim Körppen[1] · Hannah Lauppe[1] · Norbert Gronau[1]

Clementine Bertheau
clementine.bertheau@wi.uni-potsdam.de

Tim Körppen
tim.koerppen@wi.uni-potsdam.de

Hannah Lauppe
Hannah.lauppe@wi.uni-potsdam.de

Norbert Gronau
Norbert.gronau@wi.uni-potsdam.de

[1] Chair of Business Informatics, Esp. Processes and Systems, University of Potsdam, 14482 Potsdam, Germany