Review Article GU J Sci 38(2): 706-729 (2025) DOI: 10.35378/gujs.1467890

Gazi University

Journal of Science

http://dergipark.gov.tr/gujs

An in-depth Examination of Logical Data Models Utilized in Data Storage Systems to Facilitate Data Modeling

Ahmet Arif AYDIN *

Inonu University, Engineering Faculty, Computer Engineering Department, Malatya, Türkiye

Highlights

- This paper focuses on logical data models of data storage systems.
- Advantages, disadvantages and trends in data storage systems are presented.
- A consolidated overview of the characteristics of logical data models is provided.

Article Info

Received: 13 Apr 2024
Accepted: 29 Dec 2024

Keywords

Data models
Data modeling
Data storage systems
Information systems
NoSQL

Abstract

Data is a crucial asset in the current era of big data. Organizations collect, store, and analyze data at different scales, velocities, types, and structures to aid their decision-making. Database management systems (DBMS) also play a key role in properly storing large amounts of data. Understanding data models and selecting the appropriate database are essential for achieving scalable storage and efficient query performance. The motivation and main purposes behind this work are to present important characteristics of prominent logical models of data storage systems in one place in order to accomplish the following goals: First, providing a detailed guide on logical data models of DBMS, starting from legacy ones to modern contemporary systems, all in one place; secondly, presenting a consolidated and comparative overview of the characteristics of logical data models for researchers, database designers, and developers of data-intensive systems to guide them in selecting the appropriate data storage system for data modeling tasks; and lastly, presenting an overview of popular data storage systems and their data models to illustrate current trends in DBMS.

1. INTRODUCTION

In this big data era, almost all organizations have been voluntarily involved in the generation of data in diverse formats, speeds, and sizes. Globally well-known companies such as Google, Twitter, Facebook, LinkedIn, and Microsoft manage continuously increasing petabytes of structured, semi-structured, and unstructured data for a variety of purposes [1]. Regarding the infographic provided by DOMO Company [2], every minute of a single day, enormous amounts of data are generated by a variety of sources [3]. In 2006, Clive Humby declared, “data is called the new oil” [4] because it contains hidden, valuable insights [5] for organizations to survive in the job market. Moreover, gleaning useful information and insights out of the new oil does not come for free; thus, organizations must deal with the numerous challenges involved in collecting, storing, and analyzing stages of big data processing [6].

Choosing an appropriate DBMS out of hundreds [7] and developing a reliable, scalable, and effective database design which matches user demands are crucial and challenging. To handle these challenges, database designers play a crucial role in developing a feasible design to meet domain requirements and user needs. This requires domain knowledge, internalizing the data model capabilities of data storage systems, and applying data modeling capabilities in design. Moreover, efficiently utilizing a proper data model is not a recent challenge; it was identified in the 1960s during the development of pioneer database management systems [8], and even today, this challenge has not been completely overcome because of big, large, fast, and various types of data such as numerical, text, image, audio, video, or signal.

* e-mail: arif.aydin@inonu.edu.tr

Data modeling is one of the key demanding capabilities in big data era since it deals with carefully designing feasible conceptual and logical models for capturing user requests, transforming a logical model to a physical data model to store data in a scalable way, and later efficiently retrieving data for a variety of analysis purposes [9]. Due to the importance of data modeling, this paper aims at presenting state-of-the-art logical data models ranging from pioneer DBMS to most contemporary data storage systems all in one place, to provide a comprehensive guide for database management system researchers, to support the data modeling responsibilities of database designers and developers of data-intensive systems, and to provide data model features and capabilities to aid in the selection of the appropriate database. This paper's main contribution to the literature lies in its comprehensive presentation of all prominent logical data models in one location. Although previous research works since the 1960s have only focused on a portion of logical data models, as shown in Table 1, this research work presents features, characteristics, and capabilities of all prominent logical data models in one place to guide researchers, database designers, and developers of data-intensive systems in selecting an appropriate data storage system for their data modeling tasks.

This work is structured as follows: Section 2 offers background information by introducing concepts and terminology. Section 3 of this paper includes the methodology and a discussion of relevant work on the logical data models presented. Section 4 discusses the historical context, impacts, and significant aspects of logical data models in DBMS. Section 5 contains a discussion that presents current trends, recent breakthroughs, and advancements in the field of DBMS. Section 6 provides a conclusion by summarizing the study's contributions.

2. BACKGROUND: CONCEPTS AND TERMINOLOGY

This section presents definitions of important terms utilized throughout the paper. These terms are “data model”, “ACID”, “BASE”, “structured”, “semi-structured”, “unstructured” and “NoSQL”. Presenting definitions of these terms is important to clarify how they are defined in the state-of-the-art works in the context of database management systems. After presenting these terms, categories of data models are presented.

2.1. Definitions

In [10], “data” was explained as “the real word ideas that is conceived by people as entities and their attributes” and then the authors defined data model as “a method of logically organizing such data according to the relationships found among data.”

In [11], the “data model” term was presented as “a collection of data structure types, operators to apply data types, and general integrity rules.” Also, according to Codd, defining a data model without operators and integrity rules is “like trying to understand the way the human body functions by studying anatomy but omitting physiology.” As a result, operators and integrity rules are critical components of a data model for determining behaviors to manipulate data for requested queries [12].

In [13], the “data model” was described as “a framework of concepts that can be used to express the miniworld semantics that is incorporated into a database system defines.” According to Dittrich, the data model comprises “basic data types, constructors for composed data types, operators for CRUD (Create, Read, Update, and Delete) operations, implicit and explicit consistency constraints. In [14], the “data model” was defined as “a set of concepts that can be used to describe structure of and operations on a database” and the meaning of structure was stated as “the data types, relationships, and constraints that define the template of that database.”

In [15], the “data model” was explained as “the structure of a database, a collection of conceptual tools for describing the real-world entities to be modeled in the database and the relationships among these entities.”

In [16], the “data model” was described as “a collection of constructs for describing and structuring applications in the database,” and according to Worboys, a data model includes translatable information for

developers to implement user requests, and for users it is a general feature representation of the demanded system.

According to the authors [17,18], a data model provides a high-level view, and it can be considered as an abstraction or a blueprint of a database. To sum up, regarding the preceding definitions, a data model is comprised of knowledge about supported data types, features of data structures, relationships among defined entities, basic operations to manipulate data, and a set of rules to ensure the integrity and consistency of data.

In addition, "structured, "semi-structured," and "unstructured" terms are also significant in the context of data storage systems since these terms provide information about how data will be stored [6]. The "structured" term indicates data has a strict structure and the fields, types, and size are already defined. In the context of database management systems, this term is generally utilized for data stored in RDBMS. For example, in an e-commerce web site customer data, purchase information, or product information data can be stored in RDBMS as structured data and then later accessed by SQL. Moreover, relational databases were developed by using ACID (Atomicity, Consistency, Isolation, Durability) properties; these features make RDBMS more suitable for storing atomic, structured, and not complex data. ACID features are critical for financial, banking, and enterprise applications to enable immediate consistency, speed, security, and integrity.

The "semi-structured" term indicates there are a diverse number of key value pairs, and the values for each key can be different in terms of size and type. For instance, Twitter data can be considered semi-structured data since each tweet can have a different size, entity, and content. Therefore, we can model "semi-structured" data using XML, JSON, and BSON.

The "unstructured" term indicates that data does not have a consistent structure, such as email or log data. Therefore, "semi-structured" and "unstructured" data are not suitable to store in RDBMS. To handle the complexity of storing "semi-structured" and "unstructured" data, NoSQL technologies have emerged.

The NoSQL term is known as "Not Only SQL" [19], which means there exists an adjustable and flexible schema that enables the elimination of strict schema rules in the relational data model [20]. To eliminate strict schema rules of the relational data model and to provide flexible schema for storing large, heterogeneous, and complex data, NoSQL technologies have emerged at the top the BASE features (Basically Available, Soft State, and Eventual Consistency). BASE features aim to relax the restrictions of ACID properties and provide flexible schema for storing large amounts of data in diverse formats and sizes.

2.2. Categories

This section provides categories of data models. In the context of DBMS literature, data models are generally classified under the following categories: conceptual (high-level), logical (representational), and physical (low-level) models [14, 21-23]. These categories are crucial throughout the entire database design process. Figure 1 illustrates when complexity, details, and reliance rise, abstraction reduces while transitioning from a conceptual model to logical and physical models. A conceptual model serves as an input for logical models, whereas a logical model serves as an input for a physical model.

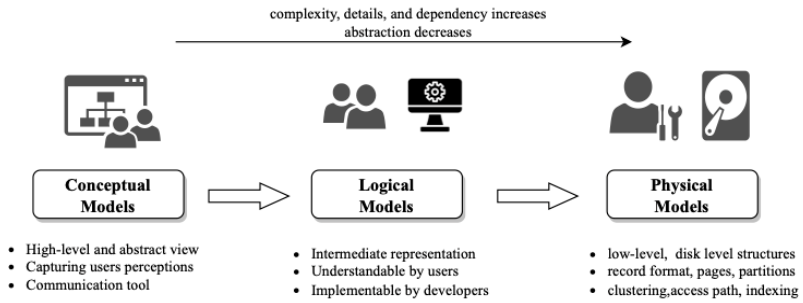


Figure 1. A High-level view of data models

Conceptual (semantic) models can be utilized as a collaboration tool to create a database-independent abstraction [24]. The role of the conceptual model is vital to capturing users' perceptions since conceptual models enable users to be included in the design process for the purpose of understanding their needs, minimizing misunderstandings, and creating a communication bridge among users and developers [25]. Regarding domain requirements and user demands, a conceptual data model provides a high-level visual representation to demonstrate information about relations, characteristics, semantics, and features of real-world entities which are planned to be stored in a DBMS [26]. For this purpose, Entity Relationship (ER) model can be utilized to create visual representations to present users' real-world demands [27]. Also, Unified Modeling Language (UML) is a widely used in many fields to visualize, build, identify, and document real-world conceptual aspects [28].

Logical (representational) data models are also utilized as a communication tool among users and developers because they are comprehensible by users and executable by developers. Moreover, a logical data model can be as an intermediate representation since it allows for including new or revised user demands after forming a conceptual data model. Because of the capabilities, the role of a logical data model is critical because it allows for the correction of any potential errors, the incorporation of new user requests, and the possibility of resolving misunderstandings before creating physical designs based on information obtained from a conceptual model [29]. One of the main purposes of this paper is to present features of logical data models to support data modeling efforts; therefore, section 4 is entirely devoted to presenting detailed explanations, visual representations, and characteristics of logical data models.

Physical data models are low-level models which define how data will be stored on a disk by using the underlying file system of the operating system. In addition, the physical data model deals with storing data in files (pages) on hard disk, creating indexes, utilizing data structures such as B+ tree, and representing access paths [14,21]. Conceptual and logical data models aim at including users in the design process since both provide an abstract, high-level representation of database design; a physical data model, on the other hand, requires low-level design and is therefore more understandable and manageable by database administrators and developers than users.

3. METHODOLOGY AND RELATED WORKS

In this study, a literature review is performed on ACM Digital Library, Google Scholar, Springer, Elsevier and IEEE Explore to find the most relevant related works by using combinations of "data models," "logical data models," and "conceptual data models" terms.

After the initial search through preceding academic databases 277 papers were identified. Then, 40 papers have been included in Table 1 based on the following inclusion criteria "*presenting features of logical data models of database management systems.*" Regarding the mentioned criteria, after manually checking the abstract, content, main contributions of those papers 18 papers from ACM, 10 papers from Google Scholar, 6 Papers from Springer, 4 papers from Elsevier and 2 papers from IEEE Explore have been chosen, carefully studied, and included in Table 1.

As shown in in Table 1, all related works only focus on a few important data models in their times. On the other hand, unlike the papers presented in Table 1, this review work provides features, characteristics, advantages and disadvantages of all presented data models which are Hierarchical, Network, Relational, Object-Oriented, Object-Relational, XML, RDF, Key-Value, Document, Wide-Column, Graph and Multi-Model. Furthermore, a minus sign (-) indicates that the cited paper does not include the specified category.

One of the primary objectives of this study is to consolidate all logical data models of DBMS into a single comprehensive source. Therefore, this review includes a total of 40 publications that aim to provide a comprehensive overview of data modeling efforts in the field of database management systems (DBMS). The purpose of this review is to assist researchers in the DBMS sector and help database designers in their data modeling tasks. The subsequent sections present each one with illustrative depictions.

Table 1. *Logical data models presented in related works*

Reference	Published In	Pioneer	Object-based	Contemporary	Multi-Model
[30]	Springer	R	-	RDF, KV, D, WC, G	MM
[29]	ACM	-	-	RDF, KV, D, WC, G	-
[31]	Elsevier	-	XML	-	MM
[32]	ACM	R	XML	RDF, KV, D, WC, G	MM
[33]	Springer	-	-	-	MM
[34]	Google Scholar	R	OO	RDF, KV, D, WC, G	MM
[35]	Google Scholar	R	OO	KV, D, WC, G	-
[36]	Google Scholar	R	-	RDF, KV, D, WC, G	-
[37]	Google Scholar	-	-	KV, D, WC, G	-
[38]	ACM	-	-	RDF, KV, D, WC, G	-
[39]	Elsevier	-	-	KV, D, WC, G	-
[25]	Elsevier	-	-	KV, D, WC, G	-
[40]	Springer	-	-	KV, D, WC	-
[23]	Google Scholar	-	-	KV, D, WC, G	-
[41]	Google Scholar	-	-		-
[42]	IEEE Explore	-	-		-
[43]	Elsevier	-	-		-
[44]	ACM	-	-	WC	-
[45]	Google Scholar	-	OO, OR	-	-
[46]	Google Scholar	-	OR	-	-
[47]	ACM	-	-	XML	-
[48]	Google Scholar	-	OO, OR	-	-
[49]	ACM	-	-	XML	-
[50]	Springer	-	-	XML	-
[51]	ACM	-	OO	-	-
[14]	ACM	H, N, R	OO	-	-
[52]	Google Scholar	-	OO, OR	-	-
[53]	ACM	-	OO, OR	-	-
[54]	ACM	-	OO	-	-
[55]	IEEE Explore	-		-	-
[56]	Springer	N, R		-	-
[57]	ACM	-		-	-
[58]	ACM	-		-	-
[13]	Springer	-		-	-
[11]	ACM	H, N, R	-	-	-
[59]	ACM		-	-	-
[10]	ACM		-	-	-
[60]	ACM	H, N	-	-	-
[20]	ACM	R	-	-	-
[26]	ACM	H, N	-	-	-

H: Hierarchical, N: Network, R: Relational, OO: Object-oriented, OR: Object relational, XML, RDF, KV: Key-Value, D: Document, WC: Wide-Column, G: Graph, MM: Multi-model, "-": Not Included

4. LOGICAL DATA MODELS: FROM LEGACY TO CONTEMPORARY

This section describes prominent logical data models of DBMS in detail, including historical impacts, capabilities, characteristics, and release year. Figure 2 provides the logical data models explained in the following subsections under four categories: pioneer, object-based, and contemporary and multi-model data models.

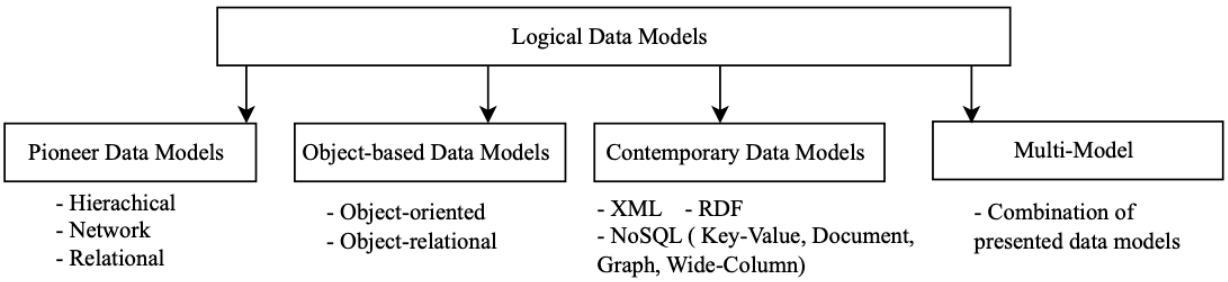


Figure 2. *Categories of logical data models*

4.1. Pioneer Data Models

In this section, hierarchical, network, and relational models are presented, respectively. These models can be considered "pioneer data models," since all data modeling efforts in the DBMS realm have been impacted by at least one of these models.

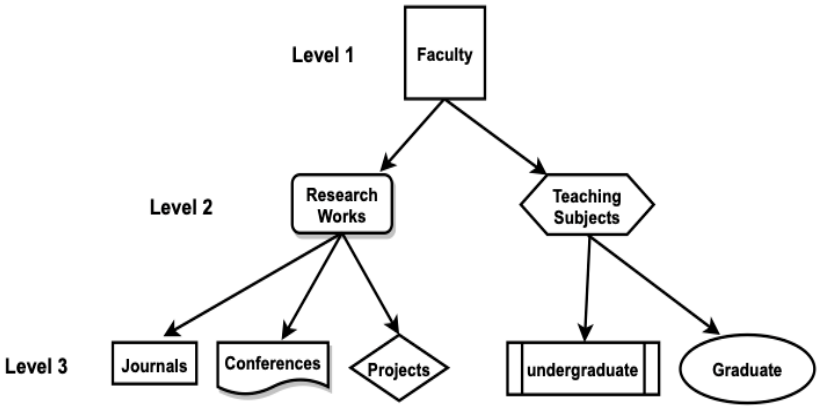


Figure 3. *An example of a hierarchical model representation*

A hierarchical data model is a tree-like structure, as illustrated in Figure 3. In a hierarchical model structure, each node is a different record type which is represented by a different shape. Each record type can include many members, and except for the root node, each level can have multiple record types. Also, each level in a hierarchical tree structure represents a set of related records which are children of a higher level and parents of a lower level (parent-child-relation). In addition, the hierarchical model restricts one node to have only one parent (one-to-one) and it allows one child to be the parent of many children (one-to-many). For example, the Research Works record type (intermediate node) is a member and child of the Faculty (root node), and parent of the Journals, Conferences, and Projects. Within a hierarchical tree structure, intermediate nodes are responsible for maintaining hierarchical relationships, whereas terminal nodes are used for storing data. The records that are displayed in level 3 in Figure 3 are referred to as terminal nodes. [10].

In 1968, the first commercially available database, Information Management System (IMS) was developed for NASA [61]. IMS was developed by IBM as the foundation for a hierarchical data model, and it is still used in government, health care, banking, and insurance databases [62]. The hierarchical model has several benefits, including a user-friendly model that is easy to grasp, the ability to effortlessly add or remove

records, the ability to retain links between records through parent-child relationships, support for efficient search operations through its tree structure, and its suitability for modeling hierarchical data. On the other hand, the hierarchical model comes with disadvantages such as data duplication due to only having one-to-one and one-to-many relationships and not supporting many-to-many relations; strict hierarchical ordering makes insert and delete operations very complex; deleting a parent cause losing all child records; and answering some queries requires navigating entire records stored in the database [10]. There are also limitations to the hierarchical model. For example, managing hierarchical tree structures is a complex responsibility, and its programmers must plan every detail to find the path, perform CRUD (Create, Read, Update, and Delete) operations, and manage stored data. Because of the mentioned disadvantages, the following data models were developed.

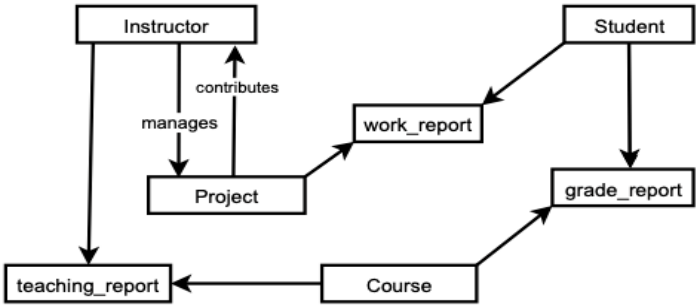


Figure 4. *An example illustration of a network model*

A network model represents data in a graph-like structure as shown in Figure 4. The network model incorporated all properties of the hierarchical model while introducing a new feature to allow a record to become a member of multiple sets (many-to-many relationship) [63]. Specifications and language (COBOL) of the network data model were defined by the Conference on Data Systems Languages Database Task Group (CODASYL DBTG) [64]; thus, the network model is also called the CODASYL DBTG model [60, 61]. The network model is comprised of two data types: records and sets [65]. A record type allows to group related data like Instructor, Student, Project, and Course, whereas a set allows for defining relationships between record types like teaching_report, work_report, and grade_report, as shown in Figure 4.

The first general-purpose DBMS, Integrated Data Store (IDS), was developed by Charles Bachman (1960-1964) and it established the basis for the network model. With this development, Charles Bachman received ACM's Turing award in 1973 [66]. Moreover, hierarchical and network models are called legacy data models since both were utilized first in the early DBMS respectively IBM's IMS and IDS [14] and also both data models are also called "navigational models" since in both cases, programmers are responsible for navigating through hierarchical and network data structures to perform CRUD operations on data records to answer user-demanded query requests. In addition, IDMS and HPIMAGE databases were developed by making use of the network data model [67].

In 1970, a new data modeling approach, the relational model, was introduced by Edgar F. Codd [20] for the purpose of addressing the mentioned limitations of hierarchical and network models [11]. The relational model introduced crucial features such as the data independence concept, which allows programmers to relax ordering, indexing, and access path dependencies; supporting application programmers not being affected during mandatory internal representation changes of stored data; and performing query requests declaratively, which allows programmers to relieve the querying burden and hide the details of querying steps from users. Moreover, the relational model addresses redundancy, provides an abstract structure view on the top of stored data, and enables dealing with integrity constraints, consistency, replications, distribution of data, security [14], algebraic, and join operations [68]. After a decade, the importance of relational model was understood, and then Relational Database Management Systems (RDBMS) became available for commercial purposes. With the development of the relational data model, Edgar F. Codd received ACM's Turing award in 1981 [12]. Some of the popular RDBMS are Oracle, Microsoft Server SQL, MySQL, PostgreSQL [69] and Informix. In addition, Figure 14 provides a popular list of RDBMSs.

The term of "relation" was used by Edgar F. Codd since the relational model was developed by using set theory. The relational model consists of data types, collections of relations, operations, and integrity constraints [20]. Each relation (table) has a degree (number of columns) and contains rows (tuples) which store number of attributes (columns) in supported data types such as integer, char, string, date, or Boolean. Each row in a relation must follow the exact order of schema and the data types given in schema. This means that each row is a unique tuple [27].

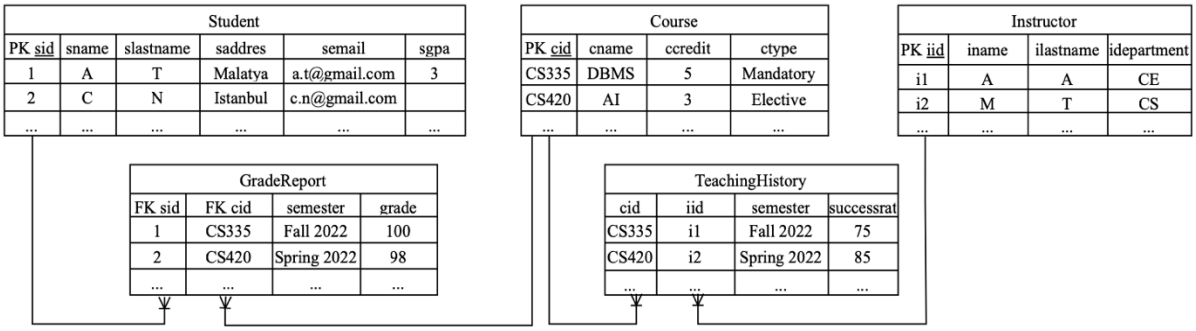


Figure 5. *An example illustration of a relational data model*

An example of a relational data model is presented in Figure 5. In this example, five relations (Student, Course, Instructor, GradeReport, TeachingHistory), degrees of each relation, primary key (PK) and foreign key (FK) constraints, and many-to-many relationships (TeachingHistory and GradeReport) are provided. For example, the GradeReport relation provides many-to-many relationships since many students can take many courses. Also, each row of GradeReport forms a unique tuple which comprises student id (sid), course id (cid), grade, and semester information. Furthermore, hierarchical (IBM's IMS) and network (IDS) database management systems were created as the basis for the documentation of hierarchical and network data models, while on the other hand, the relational database systems came out after the development of the relational model [11].

4.2. Object-Based Data Models

The object-oriented paradigm has deeply impacted the realm of software engineering and design with the development of early object-oriented programming languages such as Simula-67, Smalltalk, Eiffel, Trellis/Owl, Object LISP, and Objective C [54, 55]. Also, the object-oriented paradigm has significantly impacted the field of DBMS [70,71,51].

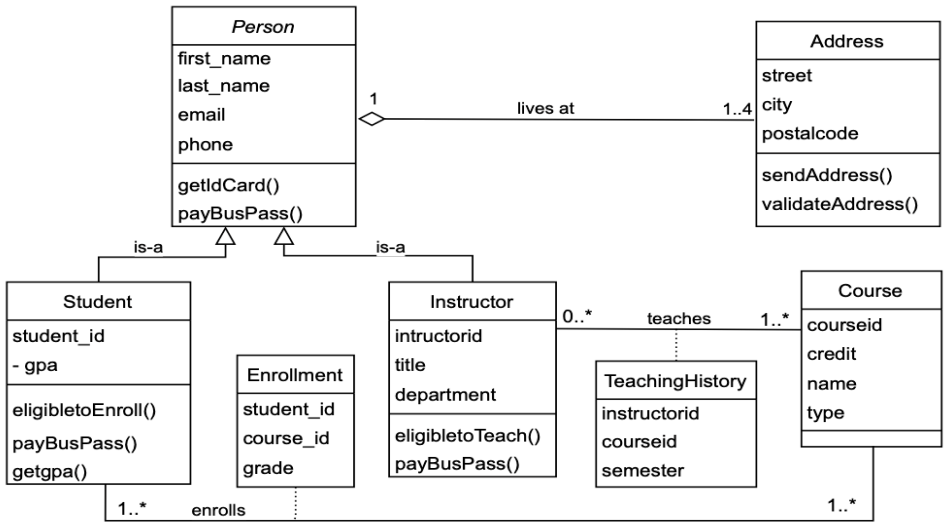


Figure 6. *An example illustration of an object-oriented model*

Object-oriented data model store data in terms of objects, properties of objects, and operations performed on objects, and an object-oriented data storage system must incorporate the following fundamental features: complex objects, object identity, types and classes, hierarchies (class or type), encapsulation, computational completeness, persistence, extensibility, concurrency, recovery, and querying [54,57,72,73]. Some of the well-known object-oriented databases are O2, Gemstone/S, ODE, ObjectDB, IRIS, ObjectDatabase++, Objectivity/DB, db4o, ObjectStore, ORION, and Versant, and Figure 14 (see section 5) provides a list of currently utilized object-oriented DBMS. Moreover, the following advantages were introduced by an object-oriented model: facilitating the task of modeling complex real-world entities, easily modeling complex data structures, incorporating all features introduced by object-oriented programming, providing extensibility to create new data types, and supporting fast data access by using pointers [74].

In Figure 6, an example of an object-oriented data model is presented. In an object-oriented model, a class may include state and behavior. Address class is a real-world entity which has street, city, and postalcode attributes that are considered to be states, and the sendAddress() and validateAddress() methods are regarded as behaviors to perform actions. The object-oriented data model provides many-to-many relationships among objects, and it provides pointers to access data. The many-to-many relationship between Student and Course classes, for example, necessitates the creation of the Enrollment class, and the TeachingHistory class is created to represent the many-to-many relationship between Instructor and Course. In Figure 6, inheritance is presented by the Student and Instructor classes since both classes are subclasses of the Person class, and it is read as "Student is a Person" and "Instructor is a Person." Specialization (child classes are more specific) and generalization (parent class is more general) are accomplished by inheritance.

All classes in Figure 6 can be considered abstract, and association is presented by lines among classes. Encapsulation is accomplished to include all related states (data) and behaviors (methods) in a class such as For behavior, the Student class contains student_id, gpa attributes for state and eligibletoEnroll(), payBusPass(), and getgpa() methods. Aggregation is presented among the Person and Address classes. Polymorphism allows for the creation of various behaviors in subclasses by overriding an existing method defined in the parent class. For example, the payBusPass() method is defined in the Person class, but its behavior is changed via override in the Student and Instructor classes. Information hiding mechanisms enable you to limit direct access to a class object. For example, the gpa attribute of the Student class cannot be directly accessed or modified from outside of the class, and it is shown by a minus sign. Aside from the benefits of the object-oriented data model, there are some drawbacks, such as the lack of a universal data model [74,75], lack of a standard query language, not existence of a mathematical foundation such as relational algebra and relational calculus [48], not providing closures to support nested queries, and not supporting join operations [73,76,77]. Due to the disadvantages of an object-oriented data model, research and development in data models were continued to fulfill various user and industry demands.

Another emerging data model is the object-relational (also known as extended-relational) model, which aims at extending the capabilities of the relational data model [52] with useful features of the object-oriented model such as user-defined data types (domain, structured types, distinct types, methods, stored procedures), inheritance, references (nested tables), and collections (array, list, set) [46,48,78]. Object-oriented features initially were introduced in SQL:1999 (also known as SQL3) and later updated in SQL:2008 (SQL/Foundation) [53]. Oracle8i [79], Sybase, DB2, Informix, Starburst, and PostgreSQL are considered as object-relational databases [80].

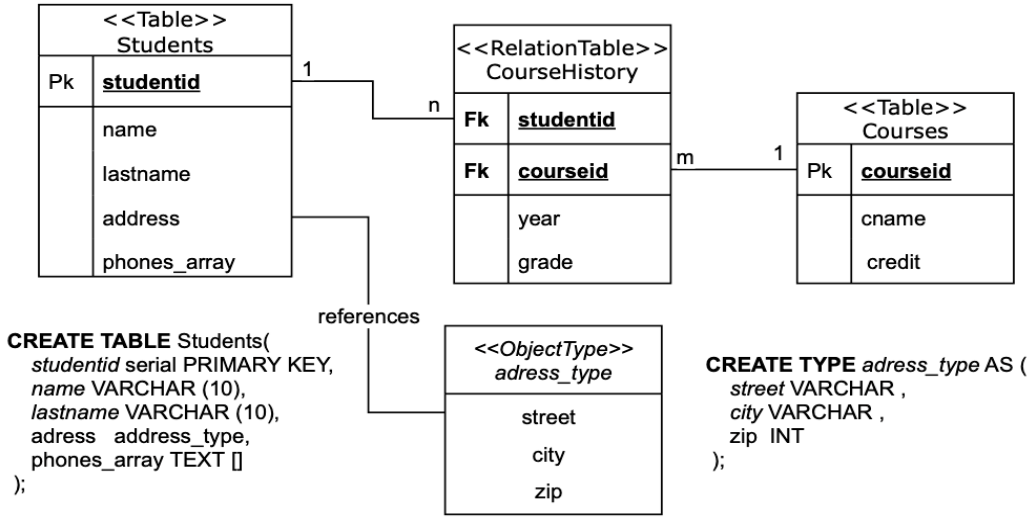


Figure 7. *An example of an object-relational model representation*

An example representation of the extended-relational model is presented in Figure 7. Beyond the capabilities of the relational data model (see section 4), the extended-relational model allows creation of complex objects with user-defined types such as address_type, which contains multiple user defined fields (street, city, zip) regarding to user demands. Also, it enables the use of containers such as a text array in the phones_array column shown in the CREATE TABLE Students command in Figure 7.

4.3. Contemporary Data Models

This section discusses contemporary logical data models such as XML, RDF, and NoSQL data models (Key-Value, Document, Wide-Column, Graph). Each modern data model will be detailed in the subsequent sections.

4.3.1. XML data model

The XML (Extensible Markup Language) data model [81] has emerged as a standard for the purpose of structuring, sharing, and exchanging semi-structured data over the web among databases and applications [50, 82,83]. The XML data model can be considered as a hierarchical tree structure [47,49]. The XML data model comprises elements, attributes, text, and document order [84,85] .

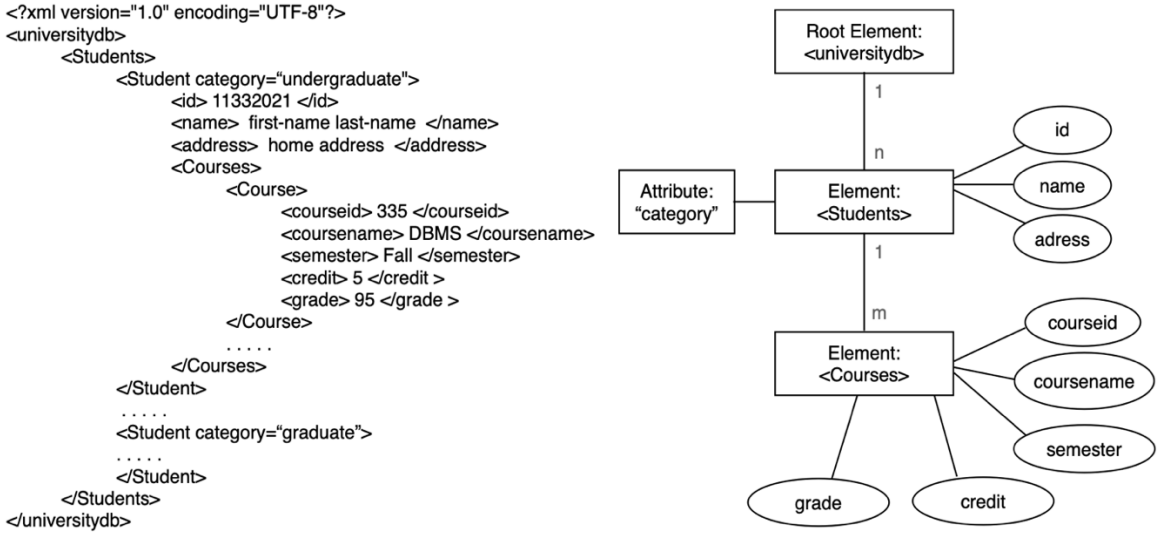


Figure 8. *An illustrative depiction of an XML data model*

An example of an XML data model is depicted in Figure 8. It requires a root element (<universitydb>) includes nested child elements (Students and Courses). Each element is required to be defined with a start (<Students>) and end (</Students>) tags within the parent element which ensures a well-formed and tree structured XML document. Also, elements in an XML document can have attributes like category, shown in Figure 8. The XML data model allows for creating a flexible tree structure that facilitates managing frequently changing, complex, and semi-structured data [83] such as tweet data. Native XML databases such as Sedna, BaseX, and eXist-db (see section 5 Figure 14) enable storing data as represented in an XML structure. Moreover, there are also exist NoSQL stores that support XML such as MarkLogic, ClusterpointServer, BagriDB, and OpenLink Virtuoso.

4.3.2. RDF data model

The Resource Description Framework (RDF) has been developed by the World Wide Web Consortium (W3C), and RDF 1.0 was released in 2004. RDF was initially developed for managing the metadata of IT resources; today, it is the standard for identifying and exchanging graph data. RDF is a directed graph which stores data as an object-based network. It employs the subject-predicate-object format to store meaning-related data facts. It represents data in the form of subject (a node), object (a node), and predicate triple components (also known as statements or RDF statements) which is an arc from subject to object [86]. It is adaptable, dynamic, and flexible, allowing for the linking of various forms of data. Using a required verb or predicate, it is possible to link any object to any other object. SPARQL, a SQL-like query language, is utilized to query RDF graphs [34,38].

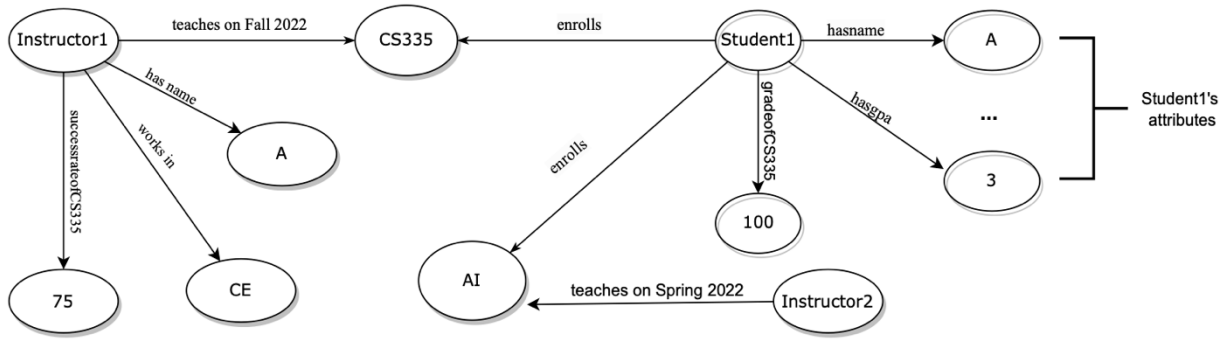


Figure 9. *An illustration of the RDF data model*

Figure 9 demonstrates how to display an RDF model by only displaying a portion of the objects from Figure 5. For example, Student 1 enrolls in CS335. Student 1 has a grade of 100 in CS335. RDF stores are a subcategory of graph data storage systems. However, RDF databases provide additional functionality beyond that of a typical graph database management system, such as indexing it for semantic search, providing text processing with the goal of constructing large knowledge graphs, enabling modeling and managing highly inter-connected data, and inferring new facts out of existing ones. Regarding DB-ENGINES [7], some of the most popular RDF stores out of twenty are Apache Jena-TDB, RDF4J, Strabon, and 4store. In addition, popular RDF stores used in various domains are presented in section 5.

4.3.3. NoSQL data models

The NoSQL term was initially introduced by Carlo Strozzi in 1998 and then re-introduced by Eric Evan to describe distributed data stores [42]. The NoSQL term stands for “Not Only SQL” and indicates there exists a flexible schema that is not restricted like the RDBMS schema structure [19]. Moreover, NoSQL data stores were initially introduced by the development of Google's Big Table [44] and then Amazon’s Dynamo [19] with the purpose of eliminating the restrictions of RDBMS, handling the challenges of large volumes of various data types, and providing a flexible schema to particularly store semi-structured, and unstructured data in diverse formats. Unlike relational databases, NoSQL storage systems aim at achieving horizontal scalability, distributing copies of data across machines to achieve high availability and performance, and

eliminating a single point of failure by allowing multiple nodes [9]. This section presents four categories of NoSQL data models that are key-value, document, wide-column, and graph models.

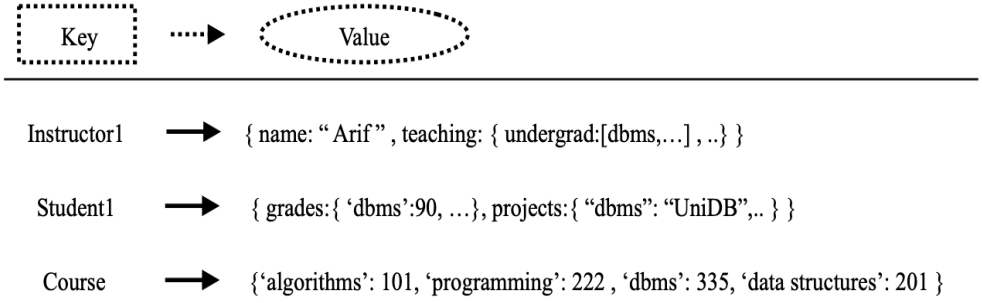


Figure 10. *An example illustration of a key-value model in Redis*

Key-value data model allows creation of collections of key-value pairs [23]. Keys can be specified by alpha-numeric identifiers and can be used like index [41] and values of associated keys can be any type of data structure (string, list, array, dictionary) supported by chosen key-value store [43,87]. Moreover, in a key-value model, search operations can only be performed on keys, not on associated values. Thus, database designers and developers must deal with the challenges of querying since there is no universal standard for querying data stored in key-value stores and each key-value store provides its own supported data structures for insert, delete, and lookup operations [19]. Key-value stores are also called in-memory stores, and one of the important characteristics of key-value stores is efficiently using both disk and memory. To accomplish this, for example, Redis allocates a user-defined amount of memory and keeps frequently used data in memory to perform fast answers for its users. Figure 14 (see section 5) provides popular in-memory data storage systems. Figure 10 provides a logical representation of a key-value data model with Redis example [88]. For example, Intructor1 is a key, and its associated value is a dictionary that contains "name" and "teaching" keys. NoSQL Redis provides a flexible schema that includes key-value pairs in nested form, such that the value of the "teaching" key is a dictionary that can also contain multiple key-value pairs without schema restriction, unlike RDBMS.

A document data model aims to store complex semi-structured data. A document can contain nested or an arbitrary number of key-value pairs in a flexible schema [89]. Each document can be stored in XML, JSON (JavaScript Object Notation) [90], YAML [23], or BSON (Binary JSON) formats [43]. A document store might contain any number of collections, depending on the available disk size of the hardware. A collection may contain an arbitrary number of documents, and each document has a unique identifier [41].

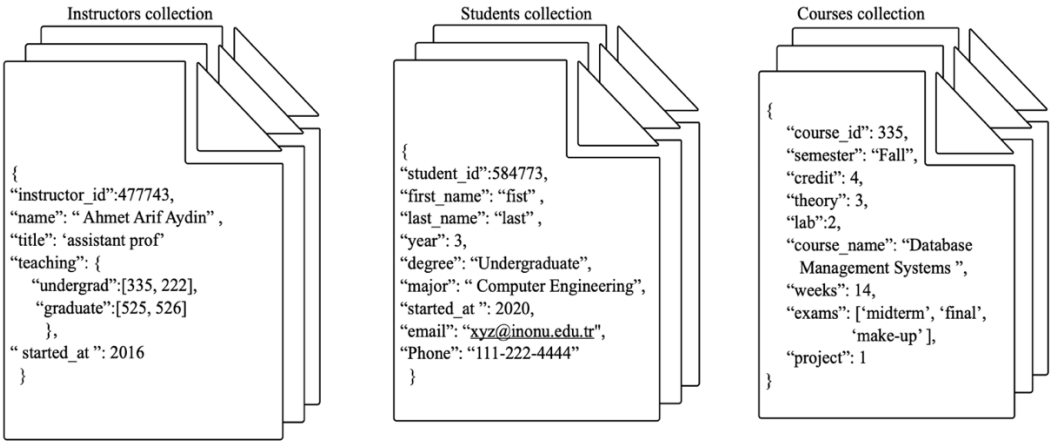


Figure 11. *An example representation of a document model in MongoDB*

Figure 11 illustrates a representation of a document model, specifically exemplifying a logical model of MongoDB [91], a document data store. Instructors, Students, and Courses collections are comprised of an

arbitrary number of documents which contain any number of key-value pairs based on user demands and database designers' data modeling expertise. Moreover, the document model provides flexible key-value pairs and nesting unlike restricted schema structure of RDBMS and it enables to store complex data and allows fully searchable keys and values with querying mechanism unlike key-value model [43]. In addition, popular document stores used in various domains are presented in section 5. The wide-column data model allows data to be stored in column families with an arbitrary number of row keys, each row key associated with an arbitrary number of column keys, and each column key pointing to a value [9,92]. Google's Big Table-inspired wide-column stores provides distributed and column-oriented storage [44]. Wide-column stores, also called column-family or extensible-record stores [41].

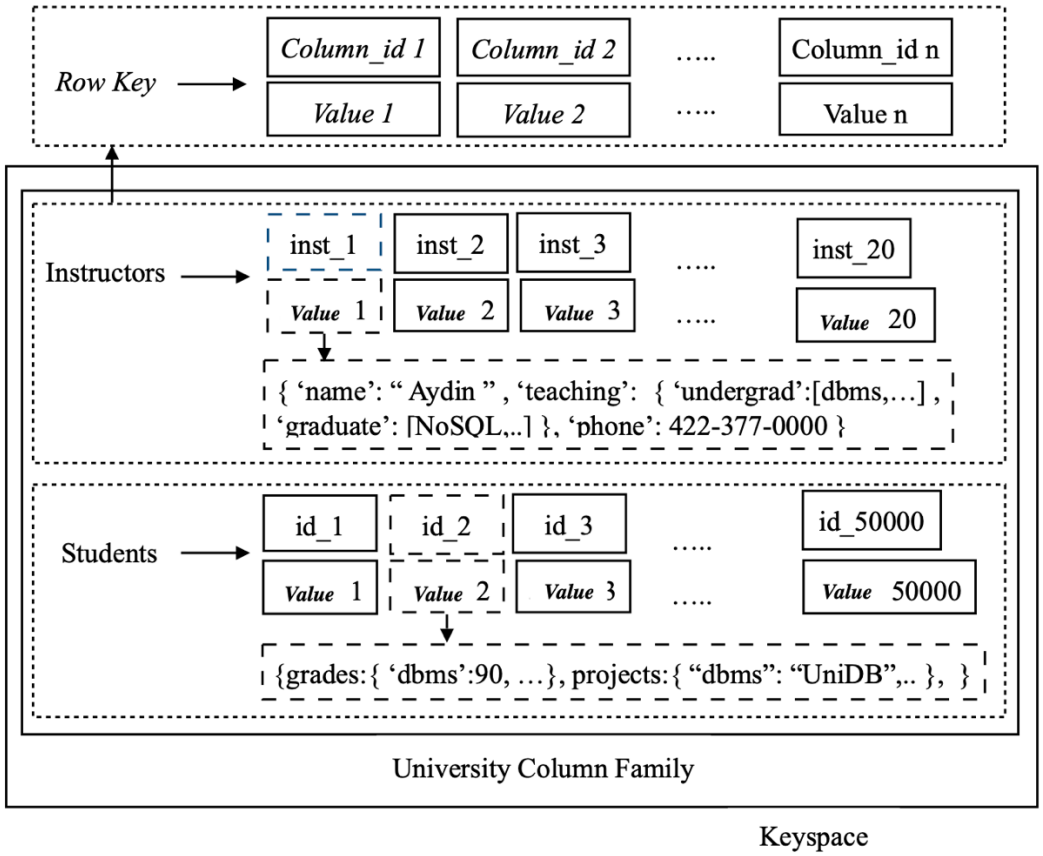


Figure 12. *An example of a wide-column model representation in Cassandra*

In Figure 12, a wide-column data model is presented with an Apache Cassandra's example [93]. Figure 12 presents a wide-column design version (Apache Cassandra) of the relational data model presented in Figure 5. In this design, Instructors and Students tables are modeled as a row in the University Column family, and row keys are, respectively, Instructors and Students. Each row key is associated with an arbitrary number of column keys, and values for each column key can be any supported data types in flexible schema. In a wide-column model, the number of column keys associated with each row key might be drastically different based on requirements and design. For example, while the Instructor's row key can have twenty column keys which are from inst_1 to inst_20, the Student's row key is associated with 50K column keys from id_1 to id_50000. Figure 12 shows an example of value for the Instructors [inst_1] and Students [id_2] column keys. Contemporary popular wide-column stores are presented in Figure 14 (see section 5).

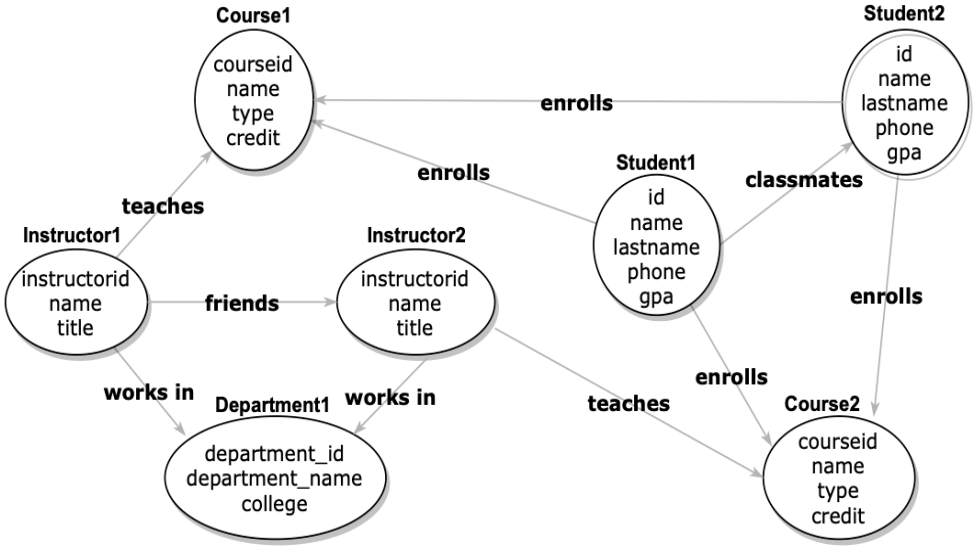


Figure 13. *An example illustration of a graph model in Neo4j*

Another emerging NoSQL data model is the graph model, which can be used to model real-world entities [22]. A graph data model is comprised of a network of nodes (real-world objects or entities), directed edges (relationships among entities), and properties (entity features represented via key-value pairs) [34]. In Figure 13 an example of a NoSQL Neo4j graph model is presented [94]. In Figure 13, each entity is defined as node, such as each course (Course1, Course2, …, Course n), each student (Student 1, Student 2, …, Student n) and each department (Department1, Department2, …, Department n). Also, each node in the graph model includes attributes of the related entity; for example, Course1 includes coursed, name, type, and credit attributes. Relations among nodes are denoted with directed edges, such as teaches relation among Instructor 1 and Course1. Moreover, graph data model is good for heavily linked data, and popular graph stores are presented in section 5.

4.3.4. Multi-model

Another important trend in DBMS is multi-model [30]. The main purpose of multi-model DBMS is to store data in backend by using more than one data model [33]. The combination of data models can be more than one data model which is presented up to this section. ArangoDB, OrientDB, Oracle, AlchemyDB are popular DBMS supports multi-model. For example, Oracle is considered a multi-model DBMS [6]. Its primary database model is relational, but in addition, Document, Graph, RDF, and Spatial DBMS data models are supported as secondary database models. Except for hierarchical and network data models, all categories presented so far have multi-model DBMS examples [32]. In Figure 14, although DBMS are presented with regard to their primary database model, DBMS having multi-model support are also presented.

5. DISCUSSION: ADVANTAGES, LIMITATIONS AND TRENDS IN DBMS

Since the early 1960s, DBMS have been crucial for storing vast amounts of diverse data. The popularity of DBMS has been consistently rising due to their efficient concurrent access, backup and crash recovery, security features, data independence, enhanced data integrity and consistency, reduced data redundancy, improved end-user productivity, efficient data management, and data modeling capabilities. Figure 14 illustrates significant advancements in the field of DBMS from 1964 to the current day, specifically focusing on the logical data models discussed in section 4. Figure 14 presents popular, open-source, commercial (commercial databases are depicted in sketched style), and actively utilized DBMS in various domains. Moreover, logical data models provide advantages as well as disadvantages when it comes to security considerations. Nevertheless, this paper does not work to discuss these specific security issues. This topic might be the subject of another study endeavor to concentrate on.

The network and hierarchical DBMS were pioneer developments in the history of DBMS, and these systems were successfully utilized in mission-critical systems. However, due to the following limitations which are complexity of performing CRUD operations and managing data in network and hierarchical models (see section 4), new data models were developed. Moreover, IBM's IMS [62] and IDMS [95] database management systems are still actively used in finance, healthcare, and manufacturing. Although managing data in these databases is very complex, these navigational databases can be adequate for using complete (stationary) or rarely changed datasets.

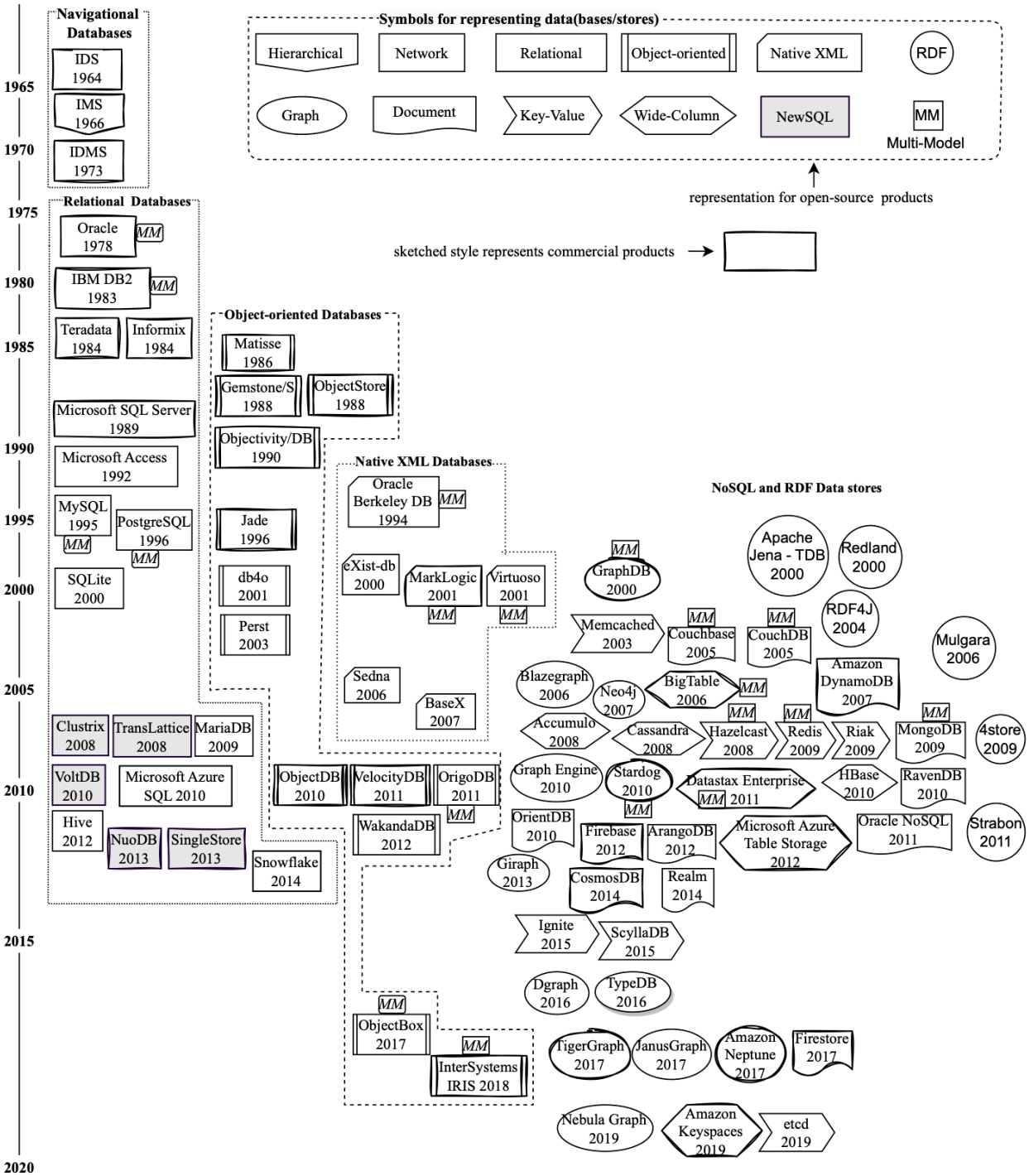


Figure 14. *Advancements in DBMS focusing on data model evolution over time*

In the 1980s, about a decade after it was created, people started to understand what the relational data model could do. Not only introducing advantageous features but also eliminating the challenges of hierarchical and network models significantly increased the popularity of the relational model. In fact, a solid

mathematical foundation of the relational model [20] was used as a medium of communication in the DBMS community to increase RDBMS usage. Also, development of a standard language for querying (SQL) in relational databases was radically supported by the RDBMS kingdom. According to DB-Engines [7], 166 RDBMS are currently in use in various domains. RDBMS is widely used in financial service in various companies since ACID features are well-suited for OLTP workloads. To give a real-world example, IMDB, the largest online database for films, film titles, actors, directors, screenwriters, and film agents, uses RDBMS to store about 10 million movie titles, film discussions, and rankings [96].

Object-oriented DBMS were developed specifically to facilitate the modeling of complex real-world entities and to incorporate features of object-oriented programming into the realm of DBMS. According to DB-Engines [7], 20 object-oriented DBMS are actively utilized, and Figure 14 provides thirteen of these. Due to the disadvantages of object-oriented databases presented in section 4, such as not having a standard and universally accepted data model, not providing a standard query language, and not existence of a mathematical foundation, their number has not increased as much as that of relational databases. With respect to historical usage and time span, like RDBMS, object-oriented databases are also actively used. For instance, Delta Air Lines, which operates flights to 326 destinations in 59 countries and employs nearly 80,000 people worldwide, built their new Crew ReRoute System using IgniteTech's ObjectStore system to handle large amounts of data and respond to inquiries quickly [97]. In addition, ObjectBox (2017) and InterSystems IRIS (2018) are recently developed, these systems are multi-model storage systems and both also support object-oriented data model [7].

Object-relational databases are not brand-new databases, and these enabled to utilize important features of an object-oriented data model (see section 4). For example, supporting complex data types such as JSON and HSTORE in PostgreSQL enables database designers to easily model frequently changing data. Thus, the inclusion of object-oriented features has been supported by the popularity of object-relational databases such as Oracle, MySQL, MSSQL, and PostgreSQL, which are the top four databases out of 421 DBMS actively used in various domains [7]. For example, Instagram utilize PostgreSQL for managing posts, tags, user information, and commands.

Another emerging storage technology in the DBMS domain is called NewSQL databases [19]. The NewSQL databases are also classified as RDBMS. In Figure 14, NewSQL databases are represented with gray backgrounds under relational databases that are VoltDB, NuoDB, SingleStore, and Clustrix [98]. The NewSQL technologies are getting more attention because they keep ACID features of relational databases and aim to include NoSQL capabilities [19]. The NewSQL movement also supports the kingdom of relational databases. Thus, RDBMS systems will continue to keep their popularity in the near future since almost all domains have been using RDBMS, and object-relational and NewSQL store technologies are also incorporating features of RDBMS [99]. For instance, VoltDB utilizes memory to achieve efficient data ingest and rapid, reliable processing while ensuring ACID compliance. Thus, the VoltDB NewSQL storage system is highly advantageous for executing instantaneous authorization, detecting real-time fraud, analyzing gaming data, and recommending customized online advertisements [100].

Native XML databases aim at facilitating managing the complexity of semi-structured data via a flexible, nested, and hierarchical data model. According to DB-Engines [7], currently seven native XML DBMS are actively utilized, and Figure 14 provides six native XML databases. Moreover, due to the importance and flexibility of modeling data with an XML structure, some databases such as Access, IBM's DB2, Informix, Matisse, MySQL, Oracle, and PostgreSQL have incorporated XML features [85]. These XML-enabled databases store data with their internal mechanisms and support creation of an XML data model externally. Furthermore, for the purpose of decreasing the restricting features of the relational model [101], XML data models also support the NoSQL paradigm.

RDBMS provides little or no support for storing large, complex, and diverse semi-structured and unstructured big data. ACID characteristics of RDBMS force centralized databases and do not enable replications of data in a distributed fashion. Also, scalability support of RDBMS is performed by scale up (vertical) that requires changing existing hardware with vendor-dependent expensive hardware. Therefore, to eliminate presented restrictions of RDBMS and to handle the challenges of big data, NoSQL data storage

systems have emerged. NoSQL storage technologies have been specifically introduced to deal with the following challenges: large volumes, variety of data types, and velocity of fast data, veracity, volatility, and value. After the development of pioneer NoSQL stores such as Big Table [44] and Amazon DynamoDB [102], popularity of NoSQL data stores has significantly increased. NoSQL data stores enable following advantageous features: flexible storage, horizontal scalability on commodity hardware, and distributed storage to increase availability and performance, and eliminating a single point of failure by providing BASE features. Flexible storage enables to store complex data types with flexible schema (Not only SQL) unlike restricted schema structure of RDBMS. Relaxing restrictions of ACID properties with providing BASE features (Basically available, Soft state, Eventual Consistency) [19]. Horizontal scalability enables to utilize commodity hardware that significantly decreasing the cost of vertical scaling. Distributed storage allowed to store multiple copies data in many nodes instead of central storage, no single point of failure supported availability fault tolerance, reliability, and ability to store data even in the presence of failures [38]. Thus, due to the advantages of NoSQL stores mentioned above, demand for migration from RDBMS to NoSQL technologies has increased [9]. The number of NoSQL technologies developed in last the 20 years also indicate the inclination toward NoSQL technologies.

According to DB-Engines [7], 70 key-value, 58 document, 41 graph, and 13 wide-column NoSQL stores are actively used in diverse domains. Key-value (in-memory) stores are generally utilized in real-time analytics to perform fast answers [103], keep temporary session information, and manage customer shopping cart and purchase operations on e-commerce web sites [104]. For example, Capital One Financial Corporation utilizing Redis to perform real-time data analysis billion of records. Adobe company is using Redis especially for high availability and advanced clustering options [88]. Document stores are suitable to store any types of large volumes of semi-structured and frequently changing user-generated data in a flexible JSON form, such as emails, logs, and large text-oriented documents [43]. A real-world example for utilizing MongoDB is Internet of Things (IoT) platforms. For example, Toyota making use of MongoDB to store data from IoT devices in Toyota Factory [91]. Wide-column stores are feasible to handle heavy reading and writing loads. Wide-column stores provide scalability, fault tolerance, and high availability with distributed storage without compromising performance [105]. For instance, entire tweet collection of X's (formerly known as Twitter) data since 2006 are stored, managed, and processed effectively via Apache Cassandra [9]. Also, Apple is using more than 75000 Apache Cassandra nodes with the purpose of achieving millions of reads/writes per seconds [93]. Graph stores are especially suitable for modeling relationships among entities such as social networks. Graph stores are generally used for providing recommendations for customers, representing and finding patterns, or figuring out fraud detection [42,43]. According to software engineers at eBay and Walmart, the Neo4j graph data store is recommended for analyzing consumer behavior, establishing connections between users and items, and delivering quicker real-time product suggestions [94].

According to DB-Engines [7], currently 21 RDF stores are actively utilized in various domains. RDF stores has advantageous for storing, managing and processing heavily linked data such as standard querying language SPARQL, efficient searching capabilities, evolving data from diverse sources, and supporting text analytics. Thus, RDF stores can be a good choice in real-life use cases for managing linked data, combining data from diverse resources, and managing social networks [86]. Moreover, the number of DBMS support multi-model has been increasing to handle variety challenge of big data. According to DB-Engines [7], currently 133 out of 421 DBMS are using multi-model. In Figure 14, multi-model DBMS are presented with MM tags.

Furthermore, choosing the right database for a job is crucial and challenging when developing the required database design for a target system because one size does not fit all. Thus, challenges pertaining to DBMS and database design are handling large amounts of data, choosing feasible data models for various domains, dealing with a variety of data types, effectively storing data in complex and diverse formats (structured, semi-structured, and unstructured data), providing efficient access for frequently accessed data, developing a feasible design via the selected data model, and handling frequently changing user demands and needs [8].

To handle the mentioned challenges, embracing the following suggestions can be beneficial. First, having domain knowledge to understand domain requirements and priorities, user demands, and data types that need to be stored. In order to support this goal, this paper presents features, advantages, disadvantages, and real-world use cases of internationally approved data models. Second, properly modeling and storing data while carefully and efficiently utilizing the data models of the target storage technology requires having knowledge of its logical data model and studying the best practices of experts in the DBMS domains. In section 5, concrete real-life example use cases are presented to support developers for their cases. Third, choosing a proper database or combination of multiple databases regarding user needs and domain requirements. In real-life data management scenarios, companies such as Apple, Netflix, Google, Facebook, and many others do not rely on only one data storage system; therefore, database designers and developers can utilize multiple data storage systems to accomplish various purposes in the big data era.

6. CONCLUSION

Properly modeling, storing, and managing data in diverse formats, speeds, and sizes generated from various sources is a challenging task. Moreover, having knowledge of the logical data models of various data storage systems is paramount to accomplish proper data modeling, managing, and analyzing large amounts of data. Therefore, this research work aims to present all logical data models of data storage systems, beginning with historical data models and moving on to modern ones, with their features in one place. This research presents a big picture view of logical data models to establish a strong foundation for researchers in the field of DBMS. After presenting the capabilities, advantages, and limitations of data models, real-world use case examples for selected data storage systems are provided to assist the data modeling tasks of developers of data-intensive systems for selecting the most suitable database for a particular job from an extensive set. To conclude, one size does not fit all; thus, developers of data-intensive systems must internalize their development goals, capture benefits from best practices from real-world use cases, embrace the challenges of making use of multiple data storage systems in their development environment, and always design for change to fulfill never-ending requests.

CONFLICTS OF INTEREST

No conflict of interest was declared by the author.

REFERENCES

[1] Yaqoob, I., Hashem, I.A.T., Gani, A., Mokhtar, S., Ahmed, E., Anuar, N.B., and Vasilakos, A. V., "Big data: From beginning to future", International Journal of Information Management, 36: 1231–1247, (2016). DOI: https://doi.org/10.1016/j.ijinfomgt.2016.07.009

[2] https://www.domo.com/learn/infographic/data-never-sleeps-9. Access date: 29.07.2024

[3] Oussous, A., Benjelloun, F.Z., Ait Lahcen, A., and Belfkih, S., "Big Data technologies: A survey", Journal of King Saud University - Computer and Information Sciences, 30: 431–448, (2018). DOI: https://doi.org/10.1016/j.jksuci.2017.06.001

[4] Han, H., Yonggang, W., Tat-Seng, C., and Xuelong, L., "Toward Scalable Systems for Big Data Analytics: A Technology Tutorial", IEEE Access, 2: 652–687, (2014). DOI: https://doi.org/0.1109/ACCESS.2014.2332453

[5] Zhu, L., Bass, L., and Xu, X., "Data management requirements for a knowledge discovery platform", ACM International Conference Proceeding Series, 169–172, (2012). DOI: https://doi.org/10.1145/2361999.2362036

[6] Aydin, A.A., "A Comparative Perspective on Technologies of Big Data Value Chain", IEEE Access, 11: 112133–112146, (2023). DOI: https://doi.org/10.1109/ACCESS.2023.3323160

[7] https://db-engines.com/en/. Access date: 28.07.2024

[8] Kerpelman, C., Olle, T.W., Everest, G.C., Fry, J.P., Fuller, M.E., Hawes, M.K., Kay, A.J., Lefkovits, H.C., Systems, H.I., Mcgee, W.C., Metaxides, A., Laboratories, B.T., Olson, R.M., Corporation, C.D., Rich, M., Mathematics, E., Schubert, R.F., Chemical, B.F.G., Sibley, E.H., Stieger, W.H., Brass, C., Vorhaus, A.H., Weinert, A.E., Command, N., Support, and S., Young, J.W., "Introduction to " Feature Analysis of Generalized Data Base Management Systems ", Communications of the ACM, 14, (1971).

[9] Aydin, A.A., Anderson, K.M., "Data modelling for large-scale social media analytics: design challenges and lessons learned", International Journal of Data Mining, Modelling and Management, 12: 386-414, (2020). DOI: https://doi.org/10.1504/IJDMMM.2020.111409

[10] Tsichritzis, D.C., and Lochovsky, F.H., "Hierarchical Data-Base Management: A Survey", ACM Computing Surveys (CSUR), 8: 105–123, (1976). DOI: https://doi.org/10.1145/356662.356667

[11] Codd, E.F., "Data models in database management", Proceedings of the 1980 workshop on Data abstraction, databases and conceptual modeling, 112–114. ACM Press, New York, New York, USA, (1980). DOI: https://doi.org/10.1145/800227.806891

[12] Codd, E.F., "Relational Database: A Practical Foundation for Productivity", Communications of the ACM, 25: 109–117, (1982). DOI: https://doi.org/10.1145/358396.358400

[13] Dittrich, K.R., "Object-Oriented Database Systems: The Notion and the Issues", 1986 international workshop on Object-oriented database systems, 3–10, (1991). DOI: https://doi.org/10.1007/978-3-642-84374-7_1

[14] Navathe, S.B., "Evolution of data modeling for databases", Communications of the ACM, 35: 112–123, (1992). DOI: https://doi.org/10.1145/130994.131001

[15] Silberschatz, A., Korth, H.F., and Sudarshan, S., "Data Models", ACM Computing Surveys, 28: 105–108, (1996). DOI: https://doi.org/10.1145/234313.234360

[16] Worboys, M.F., "Relational Databases and Beyond", Geographical Information Systems: Principles, Techniques, Management and Applications, 373–384, (2005).

[17] Silberschatz, A., Korth, H., and Sudarshan, S., Database System Concepts, Sixth Edition, Mc Graw Hill, (2011).

[18] Ramakrishnan, R., Gehrke, J., Database Management Systems, (2003).

[19] Coronel, C., Morris, S., and Rob, P., Database Systems: Design, Implementation, and Management, Ninth Edition, Cengage Learning, (2011).

[20] Cattell, R., "Scalable SQL and NoSQL data stores", ACM SIGMOD Record, 39: 12–27, (2010). DOI: https://doi.org/10.1145/1978915.1978919

[21] Codd, E.F., "Relational Data Model of Data for Large Shared Data Banks", Communications of the ACM, 13: 377–387, (1970). DOI: https://doi.org/10.1007/978-1-4614-8351-9_7

[22] Elmasri, R., and Navathe, S.B., Fundamentals of Database Systems, Sixth Edition, (2011).

[23] Angles, R., and Gutierrez, C., "Survey of Graph Database Models", ACM Computing Surveys, 40: 1–39, (2008). DOI: https://doi.org/10.1145/1322432.1322433

[24] Ribeiro, A., Silva, A., and da Silva, A.R., "Data Modeling and Data Analytics: A Survey from a Big Data Perspective", Journal of Software Engineering and Applications, 08: 617–634, (2015). DOI: https://doi.org/10.4236/jsea.2015.812058

[25] Hull, R., and King, R., "Semantic Database Modeling: Survey, Applications, and Research Issues", ACM Computing Surveys (CSUR), 19: 201–260, (1987). DOI: https://doi.org/10.1145/45072.45073

[26] Storey, V.C., and Song, I.Y., "Big data technologies and Management: What conceptual modeling can do", Data & Knowledge Engineering, 108: 50–67, (2017). DOI: https://doi.org/10.1016/j.datak.2017.01.001

[27] Bachman, C.W., "Data structure diagrams", ACM SIGMIS Database, 1: 4–10, (1969). DOI: https://doi.org/10.1145/1017466.1017467

[28] Chen, P.P.S., "The Entity-Relationship Model—toward a Unified View of Data", ACM Transactions on Database Systems (TODS), 1: 9–36, (1976). DOI: https://doi.org/10.1145/320434.320440.

[29] Booch, G., "UML in Action", Communications of the ACM, 42: 26–28, (1999). DOI: https://doi.org/10.1145/317665.317672

[30] Vera-Olivera, H., Guo, R., Huacarpuma, R.C., da Silva, A.P.B., Mariano, A.M., and Maristela, H., "Data Modeling and NoSQL Databases-A Systematic Mapping Review", ACM Computing Surveys, 54: (2021). DOI: https://doi.org/10.1145/3457608

[31] Koupil, P., Hricko, S., and Holubová, I., "A universal approach for multi-model schema inference", Journal of Big Data, 9, (2022). DOI: https://doi.org/10.1186/s40537-022-00645-9

[32] Brahmia, Z., Hamrouni, H., and Bouaziz, R., "XML data manipulation in conventional and temporal XML databases: A survey", Computer Science Review, 36(100231): 1-13, (2020). DOI: https://doi.org/10.1016/j.cosrev.2020.100231

[33] Lu, J., and Holubová, I., "Multi-model Databases: A new journey to handle the variety of data", ACM Computing Surveys, 52: 1-38, (2019). DOI: https://doi.org/10.1145/3323214

[34] Liu, Z.H., Lu, J., Gawlick, D., Helskyaho, H., Pogossiants, G., and Wu, Z., "Multi-model Database Management Systems - A Look Forward", Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 16–29, (2019). DOI: https://doi.org/10.1007/978-3-030-14177-6_2

[35] Besta, M., Gerstenberger, R., Peter, E., Fischer, M., Podstawski, M., Barthels, C., Alonso, G., and Hoefler, T., "Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries", (2019). DOI: https://doi.org/https://doi.org/10.48550/arXiv.1910.09017.

[36] Chen, J.K., and Lee, W.Z., "An introduction of NoSQL databases based on their categories and application industries", Algorithms, 12: 1–16, (2019). DOI: https://doi.org/10.3390/a12050106

[37] Ali, W., Shafique, M.U., Majeed, M.A., and Raza, A., "Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics", Asian Journal of Research in Computer Science, 4: 1–10, (2019). DOI: https://doi.org/10.9734/ajrcos/2019/v4i230108

[38] Bathla, G., Rani, R., and Aggarwal, H., "Comparative study of NoSQL databases for big data storage", International Journal of Engineering & Technology, 7: 83-87, (2018). DOI: https://doi.org/10.14419/ijet.v7i2.6.10072

[39] Davoudian, A., Chen, L., and Liu, M., "A survey on NoSQL stores", ACM Computing Surveys, 51: 40: 1-43, (2018). DOI: https://doi.org/10.1145/3158661

[40] Corbellini, A., Mateos, C., Zunino, A., Godoy, D., and Schiaffino, S., "Persisting big-data: The NoSQL landscape", Information Systems, 63: 1–23, (2017). DOI: https://doi.org/10.1016/j.is.2016.07.009

[41] Gessert, F., Wingerath, W., Friedrich, S., and Ritter, N., "NoSQL database systems: a survey and decision guidance", Computer Science - Research and Development, 32:353–365, (2017). DOI: https://doi.org/10.1007/s00450-016-0334-3

[42] V, M., "Comparative Study of NoSQL Document, Column Store Databases and Evaluation of Cassandra", International Journal of Database Management Systems, 6: 11–26, (2014). DOI: https://doi.org/10.5121/ijdms.2014.6402

[43] Kaur, K., and Rani, R., "Modeling and Querying Data in NoSQL Databases", 2013 IEEE International Conference on Big Data, 1–7, IEEE (2013). DOI: https://doi.org/10.1109/BigData.2013.6691765

[44] Battersby, S., "How a 3000-year-old code unmasked a stellar cannibal", NewScientist, (1956). 216: 43–45, (2012). DOI: https://doi.org/10.1016/S0262-4079(12)63205-9

[45] Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R.E., "Bigtable: A distributed storage system for structured data", ACM Transactions on Computer Systems, 26: 4: 1-26, (2008). DOI: https://doi.org/10.1145/1365815.1365816

[46] Bertino, E., and Guerrini, G., "Object-Oriented Databases", Wiley Encyclopedia of Computer Science and Engineering, 1–15. Wiley (2008). DOI: https://doi.org/10.1002/9780470050118.ecse279

[47] Zendulka, J., "Object-Relational Modeling in the UML", Encyclopedia of Database Technologies and Applications, 421–426, IGI Global, (2005). DOI: https://doi.org/10.4018/978-1-59140-560-3.ch071

[48] Vianu, V., "A Web odyssey", ACM SIGMOD Record. 32: 68–77, (2003). DOI: https://doi.org/10.1145/776985.776999

[49] Lord, C., Gupta, S., The Evolution of Object Relational Databases, (2002).

[50] Salminen, A., and Tompa, F.W., "Requirements for XML Document Database Systems", Proceedings of the ACM Symposium on Document Engineering, 85–94, (2001). DOI: https://doi.org/10.1145/502200.502201

[51] Wuwongse, V., Anutariya, C., Akama, K., and Nantajeewarawat, E., "A Data Model for XML Databases", Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 237–246, (2001). DOI: https://doi.org/10.1007/3-540-45490-X_28

[52] Zand, M., Collins, V., and Caviness, D., "A Survey of Current Object-Oriented Databases", ACM SIGMIS Database, 26: 14–29, (1995). DOI: https://doi.org/10.1145/206476.206480

[53] Cattell, R.G.G. Object Data Management: Object-Oriented and Extended Relational Database Systems, (1991).

[54] Stonebraker, M., Rowe, L.A., Lindsay, B.G., Gray, J., Carey, M.J., Brodie, M.L., Bernstein, P.A., and Beech, D., "Third-generation database system manifesto", ACM SIGMOD Record, 19: 31–44, (1990). DOI: https://doi.org/10.1145/101077.390001

[55] Bancilhon, F., and Kim, W., "Object-Oriented Database Systems: In Transition", ACM SIGMOD Record, 19: 49–53, (1990). DOI: https://doi.org/10.1145/122058.122063

[56] Kim, W., "Object-oriented databases: definition and research directions", IEEE Transactions on Knowledge and Data Engineering, 2: 327–341, (1990). DOI: https://doi.org/10.1109/69.60796

[57] Schek, H.-J., and Scholl, M.H., "Evolution of data models", Presented at the (1990). DOI: https://doi.org/10.1007/3-540-53397-4_35

[58] Bancilhon, F., and Kim, W., "Object-oriented database systems", ACM SIGMOD Record, 19: 49–53, (1990). DOI: https://doi.org/10.1145/122058.122063

[59] Heiler, S., Dayal, U., Orenstein, J., and Radke-Sproull, S., "An object-oriented approach to data management: why design databases need it", 24th ACM/IEEE conference proceedings on Design automation conference - DAC '87, 335–340. ACM Press, New York, New York, USA, (1987).

[60] Fry, J.P., Sibley, E.H., "Evolution of Data-Base Management Systems", ACM Computing Surveys (CSUR). 8:7–42, (1976). DOI: https://doi.org/10.1145/356662.356664

[61] Curtice, R.M., "Data base design using a codasyl system", Proceedings of the 1974 annual ACM conference, 473–480, (1974).

[62] North, K., Database Systems: The First Generation, World Wide Web Internet And Web Information Systems, (1986).

[63] https://www.ibm.com/products/ims. Access Date: 17.04.2024

[64] Taylor, R.W., and Frank, R.L., "CODASYL Data-Base management systems", Computing Surveys, 8: 67–103, (1976). DOI: https://doi.org/10.11499/sicejl1962.19.206

[65] CODASYL Data Description Committe, "CODASYL Data Description Language", Journal of Development, 155, (1974).

[66] Elmasri, R., and Navathe, S.B., Appendix E: An overview of Network Data Model. In: Fundementals of Database Systems 6th Edition, 1–24, (2010).

[67] Brown, A.S., Hirata, T.M., Koehler, A.M., Vishwanath, K., Ng, J., Pechulis, M.J., Sikes, M.A., Singleton, D.E., and Veazey, J.E., "Data Base Management for Hp Precision Architecture Computers", Hewlett-Packard Journal, 37: 34–48, (1986).

[68] Codd, E.F., "Extending the Database Relational Model to Capture More Meaning", ACM Transactions on Database Systems (TODS), 4: 397–434, (1979). DOI: https://doi.org/10.1145/320107.320109

[69] Stonebraker, M., and Kemnitz, G., "The POSTGRES next generation database management system", Communications of the ACM, 34: 78–92, (1991). DOI: https://doi.org/10.1145/125223.125262.

[70] Ketabchi, M.A., "Object-Oriented Data Models and Management of CAD Databases", IEEE, 123–128, (1986).

[71] Bertino, E., and Martino, L., "Object-Oriented Database Management Systems: Concepts and Issues", Computer (Long Beach Calif), 24: 33–47, (1991). DOI: https://doi.org/10.1109/2.76261

[72] Atkinson, M., DeWitt, D., Maier, D., Bancilhon, F., Dittrich, K., and Zdonik, S., "The Object-Oriented Database System Manifesto", Building an Object-oriented Database System: The Story of O2, 1–17, Morgan Kaufmann Publishers Inc., (1992). DOI: https://doi.org/10.1016/b978-0-444-88433-6.50020-4

[73] Schewe, K.D., and Thalheim, B., "Fundamental concepts of object oriented databases", Acta Cybernetica, 11: 49–83, (1993).

[74] Vossen, G., "On formal models for object-oriented databases", ACM SIGPLAN OOPS Messenger, 6: 1–19, (1995). DOI: https://doi.org/10.1145/219260.219262

[75] Hui, S.C., Goh, A., and Raphel, J.K., "Data modeling using an object-oriented logic language", Proceedings of IEEE Singapore International Conference on Networks/International Conference on Information Engineering: Communications and Networks for the Year 2000, SICON 1993, 2: 744–748, (1993).

[76] Yan, L., Ma, Z.M., and Zhang, F., "Algebraic operations in fuzzy object-oriented databases", Information Systems Frontiers, 16: 543–556, (2014). DOI: https://doi.org/10.1007/s10796-012-9359-8

[77] Alzahrani, H., "Evolution of Object-Oriented Database Systems", Global Journal of Computer Science and Technology, 16: 33–36, (2016).

[78] Urban, S.D., and Dietrich, S.W., "Using UML class diagrams for a comparative analysis of relational, object-oriented, and object-relational database mappings", SIGCSE Bulletin (Association for Computing Machinery, Special Interest Group on Computer Science Education). 21–25, (2003). DOI: https://doi.org/10.1145/792548.611923

[79] https://docs.oracle.com/cd/A87860_01/doc/appdev.817/a76976/toc.htm, Access date: 16.07.2024

[80] Dietrich, S.W., and Urban, S.D., Fundamentals of Object Databases: Object-Oriented and Object-Relational Design, (2010).

[81] https://www.w3.org/TR/2008/REC-xml-20081126/, Access date: 17.04.2024

[82] https://www.w3.org/XML/Datamodel.html, Access date: 16.04.2024

[83] Seligman, L., and Roenthal, A., "XML's impact an databases and data sharing", Computer (Long Beach Calif), 34: 59–67, (2001). DOI: https://doi.org/10.1109/2.928623

[84] Steegmans, B., Bourret, R., Cline, O., Guyennet, O., Kulkarni, S., Priestly, S., Sylenko, V., and Wahli, U., XML for DB2 Information Integration, (2004).

[85] http://www.rpbourret.com/xml/XMLDatabaseProds.htm#xmlenabled, Access date: 17.04.2024

[86] https://www.w3.org/TR/rdf-concepts/, Access date: 17.04.2024

[87] Ramzan, S., Bajwa, I.S., Kazmi, R., Amna, "Challenges in NoSQL-based distributed data storage: A systematic literature review", Electronics (Switzerland). 8: 1-29, (2019). DOI: https://doi.org/10.3390/electronics8050488

[88] https://redis.io/, Access date: 17.04.2024

[89] Hashem, H., and Ranc, D., "Evaluating NoSQL document oriented data model", Proceedings - 2016 4th International Conference on Future Internet of Things and Cloud Workshops, W-FiCloud 2016, 51–56, (2016).

[90] https://www.json.org/json-en.html, Access date: 16.04.2024

[91] https://www.mongodb.com/, Access date: 16.03.2024

[92] Aydin, A.A., "Incremental Data Collection & Analytics the Design of Next-Generation Crisis Informatics Software", PhD Thesis, University of Colorado Boulder, Boulder, USA, (2016).

[93] https://cassandra.apache.org/_/index.html, Access date: 16.03.2024

[94] https://neo4j.com/, Access date: 16.03.2024

[95] https://www.broadcom.com/products/mainframe/databases-database-mgmt/idms, Access date: 16.03.2024

[96] https://developer.imdb.com/non-commercial-datasets/. Access date: 29.07.2024

[97] https://ignitetech.ai/softwarelibrary/objectstore. Access date: 29.07.2024

[98] Ryan, J., Big Data Velocity in Plain English, (2019).

[99] Chaudhry, N., Yousaf, M.M.: Architectural assessment of NoSQL and NewSQL systems. Distributed and Parallel Databases, 38: 881–926, (2020). DOI: https://doi.org/10.1007/s10619-020-07310-1

[100] Ryan, J.: Oracle vs. NoSQL vs. NewSQL Comparing Database Technology. (2018).

[101] Truică, C.O., Apostol, E.S., Darmont, J., and Pedersen, T.B., "The Forgotten Document-Oriented Database Management Systems: An Overview and Benchmark of Native XML DODBMSes in Comparison with JSON DODBMSes", Big Data Research, 25, (2021). DOI: https://doi.org/10.1016/j.bdr.2021.100205

[102] Lourenço, J.R., Cabral, B., Carreiro, P., Vieira, M., and Bernardino, J., "Choosing the right NoSQL database for the job: a quality attribute evaluation", Journal of Big Data, 2 (18): 1-26, (2015). DOI: https://doi.org/10.1186/s40537-015-0025-0

[103] Kekevi, U., and Aydin, A.A., "Real-Time Big Data Processing and Analytics: Concepts, Technologies, and Domains", Computer Science, 7 (2): 111–123, (2022). DOI: https://doi.org/10.53070/bbd.1204112

[104] Srivastava, K., and Shekokar, N., "A Polyglot Persistence approach for E-Commerce business model", Proceedings - 2016 International Conference on Information Science, ICIS 2016, 7–11, (2017).

[105] Schram, A., and Anderson, K.M., "MySQL to NoSQL Data Modeling Challenges in Supporting Scalability", Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity - SPLASH '12, 191–202, (2012).