[DM-01-001] Spatial Database Management Systems

Abstract

A spatial database management system (SDBMS) is an extension, some might say specialization, of a conventional database management system (DBMS). Every DBMS (hence SDBMS) uses a data model specification as a formalism for software design, and establishing rigor in data management. Three components compose a data model, 1) constructs developed using data types which form data structures that describe data, 2) operations that process data structures that manipulate data, and 3) rules that establish the veracity of the structures and/or operations for validating data. Basic data types such as integers and/or real numbers are extended into spatial data types such as points, polylines and polygons in spatial data structures. Operations constitute capabilities that manipulate the data structures, and as such when sequenced into operational workflows in specific ways generate information from data; one might say that new relationships constitute the information from data. Different data model designs result in different combinations of structures, operations, and rules, which combine into various SDBMS products. The products differ based upon the underlying data model, and these data models enable and constrain the ability to store and manipulate data. Different SDBMS implementations support configurations for different user environments, including single-user and multi-user environments.

Keywords: database, DBMS, relational data models

Author & citation

Nyerges, T. (2021). Spatial Database Management Systems. The Geographic Information Science & Technology Body of Knowledge (1st Quarter 2021 Edition), John P. Wilson (ed.). DOI: 10.22224/gistbok/2021.1.11.

This Topic is also available in the following editions: DiBiase, D., DeMers, M., Johnson, A., Kemp, K., Luck, A. T., Plewe, B., and Wentz, E. (2006). Basic Data Structures. The Geographic Information Science & Technology Body of Knowledge. Washington, DC: Association of American Geographers. (2nd Quarter 2016, first digital)

Explanation

1. Introduction to Spatial Database Management Systems
2. Example DBMS and Spatial DBMS Software
3. Examples of Enterprise Spatial DBMS

1. Introduction to Spatial Database Management Systems

Spatial database management systems, both software and hardware sub-components, organize data for inventorying and querying databases, conducting spatial analysis, and creating map visualizations within an integrated manner for managing large data stores

(Yeung and Hall 2007). Database management is a subset of a larger category of technology called data management technology. Data are managed using two types of computer-based files, physical files and logical files. A physical file is a collection of records managed by the operating system software as stored on disk; a data file being different than a database file. A logical file is a collection of records managed by application software, most fundamentally database management system software. Many logical files can be combined into a physical file. One advantage for using logical files is the increase in access speed to individual data elements, as opening a physical file takes considerable time in contrast to accessing individual elements within a logical file. When data are organized into physical files to be managed, we call this 'data file management' (or simply file management). When we use logical data files organized within physical files, we call this database management. When a logical file is the same as a physical file then the file is called a 'data file'. When multiple logical files are included in a physical file then we refer to the file as a database file (Rigaux, Scholl and A. Voisard 2002).

Spatial database management adds the spatial aspect (dimensions of space) to database management (Shekhar and Chawla 2003). Database management software is designed specifically with a spatial aspect in mind, as three dimensions of physical space are core to existence. These three dimensions are managed (stored and retrieved) in a special manner in data management software, making spatial database management software an enhanced-type of data management software based on the data model design.

A data model is essentially a design framework for a data management system. One of the most comprehensive definitions of a data model was provided by Edgar Codd (1980) ten years after he developed the design of relational data models (Codd 1970). Codd's interest stemmed from clarifying the logical character of a data model, as opposed to its physical implementation; as such, the general concept of data model is not restricted to any particular approach to data management. From a database design perspective, a more common and popular understanding of data model is that it defines the structure and intended meaning of data (West 2011, p. 5). However, Codd's (1980, p. 112) more comprehensive view characterizes a data model as consisting of three components: 1) a collection of data structure types (the building blocks of any database that conforms to the model) for describing data; 2) a collection of operators or inferencing rules for manipulating data, which can be applied to any valid instances of the data types listed in (1), to retrieve or derive data from any parts of those structures in any combinations desired; 3) a collection of general integrity rules for validating data, which implicitly or explicitly define the set of consistent database states or changes of state or both -- these rules may sometimes be expressed as insert-update-delete rules. Herein, the Codd (1980) framework is used to describe SDBMS due to its completeness, whereas the West (2011) interpretation and all others like it provides just the first third of the framework.

Data logical structures such as tables, objects, attribute fields and relationships as descriptions of data are implemented as physical data storage structures with data access mechanisms for primary and foreign keys. Basic data types such as integers (e.g., 1, 2, 3) and real numbers (e.g., 1.1, 1.2, and 1.3), and/or character strings (e.g., 'text string') are extended into spatial data types (e.g., points, polylines and polygons), and are used to form spatial data structures for data storage.

Basic DBMS operations for manipulating data include data creation (C), retrieval (R), update (U), and delete (D), referred to as the CRUD suite of operations. Logical operations and

their physical implementation are used to derive logical structures and store them in terms of storage structures.

Rules constitute the third component of a data model, hence the DBMS. DBMS rules protect against corruption of the data by validating data (hence data structures) during CRUD operations. Validity rules are critically important for establishing the veracity of databases, protecting against unintended changes by users. Atomicity, consistency, isolation, and durability is a set of properties in database transactions that are intended to guarantee data validity despite errors, e.g. power failures. A sequence of operations using these properties is called a "valid transaction."

In summary of the above characterization, three levels of data abstraction combined with the three components of a data model summarize the aspects of DBMS (see Table 1). Thus, all DBMS software implementations should contain explicit capabilities for three components 1) constructs, 2) operations, and 3) rules for all three levels of conceptual (meaning), logical (structure), physical (data formatting) levels of data abstraction, respectively.

Table 1. Levels of Abstraction and Components of DBMS Data Models

Levels of Abstraction	**Three Components**		
	Constructs for describing	**Operations** for manipulating	**Rules** for validating
Conceptual	... worldly features	... worldly processes	... features and processes
Logical	... data primitives of the database	... data primitives of the database	... data and operations on the database
Physical	... disk storage formats	... data stored as bytes and bits	... reads and writes to disk

In further clarification, many people understand data model as a collection of data categories and relationships (West 2011). As such, that interpretation is simply the first component offered by Codd (1980). However, it should be clear that operations on data structures plus rules for qualifying data structure and/or operations are essential in operational database management systems. Without the operations there is no ‘change’ in data being managed. Without the rules, the veracity of the data and operations can be easily called into question. With rules constituting the third component of a data model, DBMS rules protect against corruption of the data by validating data (hence data structures) during CRUD operations. Consequently, it is important to embrace the Codd (1980) interpretation for complete implementation and use of a SDBMS.

2. Example DBMS and Spatial DBMS Software

Logical data models underpin the designs of DBMS software. Consequently, these data models underpin implementations of Spatial DBMS package implementations.

2.1 Logical Data Models as DBMS Types

A variety of logical data model types for implementing DBMS exist, each type being a different implementation of a logical data language with a physical context. (See the GIS&T BoK entry for logical data model description.) To provide an idea of the most popular DBMS software systems across the world based on logical data models, DB-Engines (2020) maintains a website documenting general rank of popularity (using six criteria to form the ranks) among 300+ DBMS. The top-ten ranked DBMS and associated data models show that the relational model is the most popular (See Table 2).

Table 2. Top-ten Ranked DBMS of 363 Systems Ranked by DB-Engines-Website (December 2020)

DB-Engine Rank	DBMS Name	Data Models Supported* (but not all are SDBMS capable)
1	Oracle	Relational, Document Store, Graph, and RDF Store
2	MySQL	Relational and Document Store
3	Microsoft SQL Server	Relational, Document Store, and Graph
4	PostgreSQL	Relational and Document Store
5	MongoDB	Document Store and Search Engine
6	IBM Db2	Relational, Document Store, and RDF Store
7	Redis	Key-value Store, Document Store, Graph, Search Engine, Time Series
8	Elasticsearch	Search Engine and Document Store
9	SQLite	Relational
10	Cassandra	Wide column

* Definitions of the data models are provided in the text below.

The majority of DBMS available have been implemented based on the relational data model, or a derivation thereof, due to its long history of success as one can observe from the above table. This success and thus popularity is due to its simplicity of data storage for maintaining validity of database elements. However, there are many other DBMS implementations based on other logical data models as well because they offer richer data storage structures. The simpler the data structure storage, the more manipulation is needed to achieve an end result. With computers being faster over the decades, the rich data structure (non-relational) approaches have been gaining in popularity. The data model types described below appear in alphabetical order. There is no implied recommendation in the listing.

- **Graph** uses a data storage approach having nodes, links and properties. Nodes are units of data commonly constituting phenomena. Links are the relationships between the nodes. Properties are the characteristics of the nodes and relationships. The underlying logic is graph-theoretic which offers a rigorous approach to construction and retrieval. Operations can be performed on the node constructs to establish links as stored relationships. Rules guide the operations and structures to enhance validity of the nodes and links.
- **Document Store** uses a data storage approach wherein the primary unit is a document with direct access from document to document. Often thought of as a

graph approach, but the constructs can be adhoc in character, and do not necessarily involve the rigor of a graph-theoretic approach. This approach is often labeled as a 'NoSQL' approach, indicating that it is a non-relational approach.

- **Key-value Store** uses a storage and access approach wherein data elements are the units of access with fine granularity. A key-value store is more general called a NoSQL approach indicating that it is a non-relational approach.
- **Object-oriented** uses a data storage and access for individual units about things in the world. The units have behaviors stored as methods containing the operations. Rules are used to constrain the behaviors of the objects.
- **Open Standards** use a data storage approach based on constructs promulgated by the Open Geospatial Consortium (OGC), wherein everyone one has access to information about storage and operations making it easier to integrate among data stores. The constructs tend to be simpler than other approaches for enhancing readability among software systems. Both vector and raster data types are included in the data structures. The vector geometry involves point, polyline and polygon geometry which stores features 'geometry only', i.e., no relationships among data elements are stored as part of the feature, and thus referred to as 'simple feature geometry' as part of the OGC data model documentation.
- **Relational** (commonly row-store indexing) uses a data storage approach wherein characteristics of phenomena are combined into relations (tables) and these relations can be manipulated using a relational algebra that constitute the operations. Rules constrain the combinations of characteristics and the operations on these combinations.
- **Relational Column Store** uses a data storage approach wherein the characteristic is the main access point, and all phenomena with that characteristic are manipulated rapidly. It uses a similar approach to relational, but the indexes are built on columns as opposed to rows.
- **Resource Description Framework (RDF)** stores use Internet-oriented formatting for implementation on the World Wide Web. The formats are designed to be read and understood by computers using the extensible make-up language (XML), wherein XML is a very common way to extend the hypertext mark-up language (HTML). XML extensions to html focus on 'content' as opposed to the 'format' focus of HTML.

Every spatial database management system makes use of spatial data types that are 'built-on-top' of general data types. The GIS&T Body of Knowledge physical data model entry offers the list of general data types.

2.2 Spatial DBMS Products

A Wikipedia (2020) page about spatial databases describes a wide variety of spatial DBMS products (https://en.wikipedia.org/wiki/Spatial_database). Below we categorize that list in terms of the principal logical data model used, as some software products support multiple data models. Again, we use alphabetical order to list the types as above, and within each data model category we alphabetize the software products, with no priority order intended. A December 2020 ranking of popularity as scored by DB-Engines website appears in parentheses, wherein NR is not ranked because DB-Engines ranks general DBMS only as opposed to more specific SDBMS. As such, the rank does not imply popularity of the SDBMS, only the DBMS used to host the SDBMS.

Graph

- Neo4j, a graph database that can build 1D and 2D indexes as B-tree, Quadtree and Hilbert curve directly in the graph (ranking = 19)
- AllegroGraph is a graph database that provides a novel mechanism for efficient storage and retrieval of two-dimensional geospatial coordinates for Resource Description Framework data; it includes an extension syntax for SPARQL queries (ranking = 166)

Document Store

- CouchDB a document-based database system that can be spatially enabled by a plugin called Geocouch (ranking = 36)
- Elasticsearch is a document-based database system that supports two types of geo data: geo-Point fields (lat/lon pairs) and geo-shape fields (points, lines, circles, poloygons, multi-polygons and others) (ranking = 8)
- GeoMesa is a cloud-based spatio-temporal database built on top of Apache Accumulo and Apache Hadoop; GeoMesa supports full OGC Simple Geometry Features and a GeoServer plugin
- MarkLogic, MongoDB, and RethinkDB support geospatial indexes in 2D (NR)
- RavenDB supports geospatial indexes in 2D (ranking = 85)

Key-value Store

- Redis with the Geo API (ranking = 7)
- Tarantool supports geospatial queries with RTREE index (ranking = 132)

Object-oriented

- Smallworld VMDS, the native GE Smallworld GIS database (NR)

Open Standards

- SpatialDB by MineRP, an open-standards spatial database with spatial type extensions used mostly within the mining industry (NR)

Relational

- Caliper extends the Raima Data Manager with spatial datatypes, functions, and utilities (ranking = 227).
- CartoDB, a cloud-based geospatial database on top of PostgreSQL with PostGIS (NR).
- Esri File geodatabase, plus support of single-user and multiuser relational geodatabases (NR).
- H2 supports geometry types and spatial indices as of version 1.3.173 (2013-07-28); an extension called H2GIS available on Maven Central gives full OGC Simple Features support (ranking = 49).
- IBM Db2 Spatial Extender can spatially-enable any edition of DB2, including the free DB2 Express-C, with support for spatial types (ranking = 6).
- IBM Informix Geodetic and Spatial DataBlade extensions auto-install on the use and expand Informix’s datatypes to include multiple stand coordinate systems and support for Rtree indexes. Informix datatypes can also be incorporated with time series data

support for tracking objects in motion (ranking = 30).
- Linter SQL Server supports spatial types and spatial functions according to the OpenGIS specifications (ranking = 309).
- Microsoft SQL Server has support for spatial types since version 2008 (ranking = 3).
- MySQL DBMS implements the datatype geometry, plus some spatial functions implemented according to the OpenGIS specifications, but different version offer different levels of support for spatial data types (ranking = 2).
- OpenLink Virtuoso supports SQL/MM, with significant enhancements including GeoSPARQL (ranking = 111).
- Oracle Spatial and Graph aid users in managing geographic and location-data in a native type within an Oracle database, potentially supporting a wide range of applications for spatial data (ranking = 112).
- PostgreSQL DBMS uses the spatial extension PostGIS to implement the standardized datatype geometry and corresponding functions (ranking = 4).
- SpatiaLite extends Sqlite with spatial datatypes, functions, and utilities (ranking = 9).
- Spatial Query Server from Boeing spatially enables Sybase ASE (NR).
- Teradata Vantage, is a data intelligence platform, deploys on-premises, to the cloud, or as a hybrid model. Vantage consists of various analytics engines on a core relational database, including its MPP engine, the Aster graph database, and a machine learning engine (ranking = 14).

Relational Column Store

- MonetDB/GIS extension for MonetDB adds OGC Simple Features to the relational column-store database (ranking = 123).
- SAP HANA is a multi-model in-memory data environment (ranking = 20).
- Vertica Place, the geo-spatial extension for HP Vertica, adds OGC-compliant spatial features to the relational column-store database (ranking = 32).

3. Examples of Enterprise Spatial DBMS

Several situations exist for user environments, including single-user, workgroup, enterprise, and consortium activities. Single-user SDBMS involves a single person at a time making use of a database environment. Workgroup database management activity involves multiple people performing database management on the same project records within a single unit (division) of an organization, that is, an intra-organizational same unit context. Enterprise database management activities involve multiple people performing databases management on the same project records within multiple units across an organization, that is, an organization-wide, but different unit context. Consortium database management activities involve multiple people performing data management on the same project records across organizations, that is, an inter-organizational context. In all multiple user contexts, conflicts with record access can occur when multiple users try to update the same database record at the same time. Those circumstances require record-locking capabilities, wherein record-locking protects users ‘stepping on’ one another changes, potentially resulting in database corruption. Enterprise SDBMS are among the most common types of data management implementations across the GIS industry.

By combining a list of DBMS supported by Esri and the ranked list from the DB-Engines

website we gain a sense of the popularity of a DBMS being used to host a GIS enterprise approach with the Esri geodatabase DBMS environment (See Table 3). Table 2 and 3 present world-wide lists. Only two of the DBMS solutions fell in rank from December 2019 to December 2020. This might indicate that SDBMS is on the rise world-wide.

Table 3. DBMS Support for Esri Geodatabase Data Model as an Enterprise Approach*

Esri DBMS-Compatibility**	Rank on DB Engines Website (363 DB Engines ranked Dec 2020)		Data Model(s) Listed on DB Engines Website, for each of respective DBMS
	December 2020	December 2019	
Oracle	1	1	Relational, Document Store, Graph, and RDF Store
Microsoft SQL Server	3	3	Relational, Document Store, and Graph
PostgreSQL	4	4	Relational and Document Store
IBM Db2	6	6	Relational, Document Store, and RDF Store
SQLite	9	11	Relational
Teradata Data Warehouse Appliance	14	15	Relational, Document Store, Graph, and Time Series
Microsoft Azure SQL Database	16	25	Relational, Document Store, and Graph
SAP HANA	20	20	Relational, Document Store, and Graph
IBM Informix	30	26	Relational, Document Store, and Time Series
Netezza Data Warehouse Appliance	34	33	Relational
ALTIBASE	142	147	Relational
Dameng	Nor ranked	Not ranked	Relational

* Relational data model is supported by Esri DBMS software. Other data models might be supported through customized software. Dameng is not compatible with the geodatabase data model.

** Esri's ArcMap (10.8) and ArcGIS Pro (2.7)

References

Codd, E. F. (1970). A Relational Data Model for Large Shared Data Banks. Communications of the ACM. 13(6), 377-387.

Codd, E. F. (1980). Data models in database management. ACM SIGMOD Record - Proceedings of the workshop on Data abstraction, databases and conceptual modelling, 11(2), 112-114.

DB-Engines (2020). DB-Engines Ranking. DB-Engines.

Rigaux, P., Scholl, M. and Voisard, A. (2002). Spatial Databases: With Application to GIS. San Francisco:Morgan-Kaufmann.

Shekhar, S. and Chawla, S. (2003). Spatial Databases: A Tour. New York: Pearson Higher Education.

West, M. (2011). Developing High Quality Data Models. San Francisco, CA: Morgan Kaufmann Publishers Inc.

Wikipedia contributors. (2023, December 13). Spatial database. In Wikipedia, The Free Encyclopedia. Retrieved 19:32, January 20, 2024.

Yeung, A. K. W. and Hall, G. B. (2007). Spatial Database Systems: Design, Implementation and Project Management, Springer, Dordrecht, Netherlands.