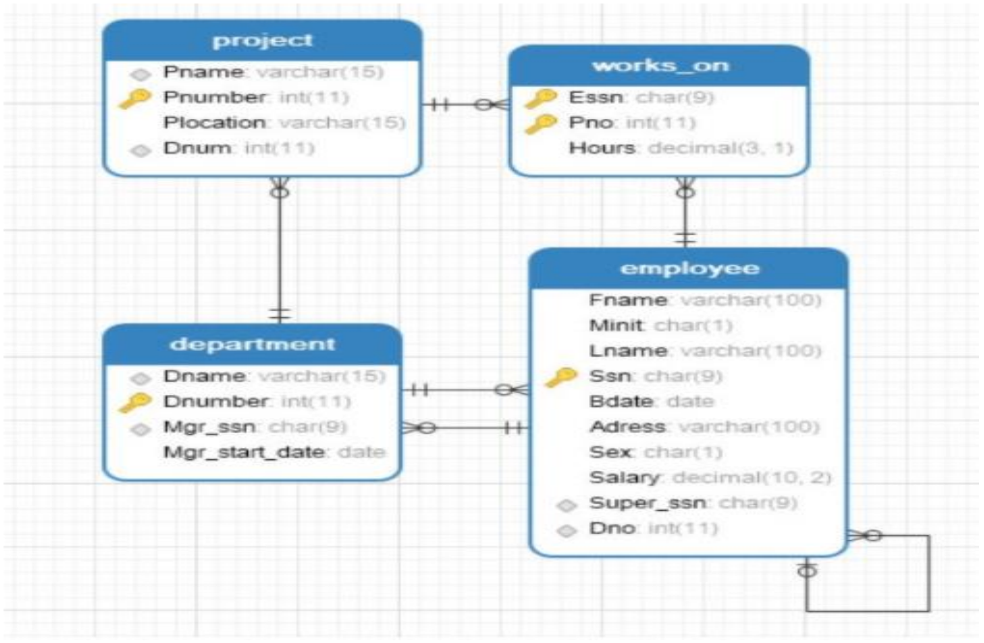
Types of Database

RELATIONAL DATABASE(RDBMS)

What is Relational Database?

The relational database is a database design format that allows the data to be stored in different tables according to certain criteria and to ensure data integrity by establishing relationships between these tables. Although the database is fully designed to meet the needs of the project, there are some basic rules for determining the relationship between database tables and ensuring normalization.

The following example is important to understand the relationship between the tables.



The data are divided into tables according to clear criteria instead of being kept in a single table. When examining the tables, we can reach the following results.

- These relational database is about a company
- It has many departments and workers in these departments
- In addition, these departments carry out various projects and the workers are works on these projects
- Relationships can be between concrete entities such as workers and departments, or between abstract entities such as project and workers

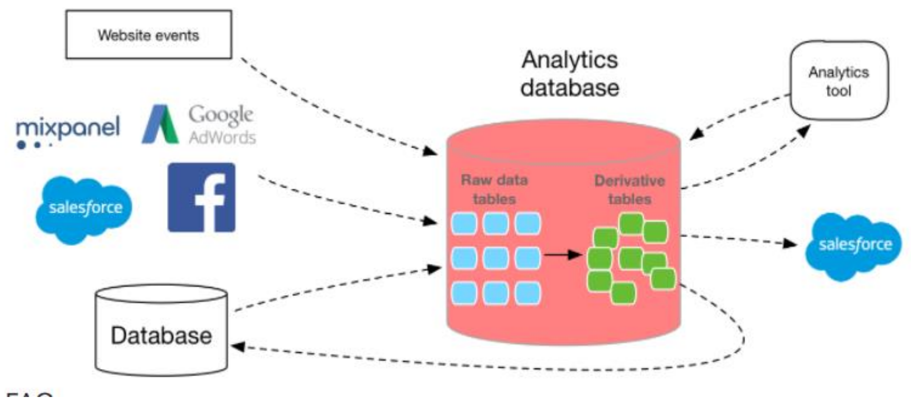
Primary key is the key to separate a record in the table from other records. The SSS can be a primary key because there cannot be two employee with the same SSN. However, the employee name cannot be a primary key because there may be more than one employee with the same name.

Foreign key is the key that correspond to the primary key field in the main table.

ANALYTICAL DATABASE

What is Analytical Database?

An Analytical Database is a type of database built to store , manage and consume big data. Optimized for processing advance analytics that involves highly complex queries on terabytes of data and complex statistical processing, data mining, and NLP(natural language processing. Examples of these are Vertica (acquired by HP),Aster Data(acquired by Teradata), Greenplum(acquired by EMC), and so on.



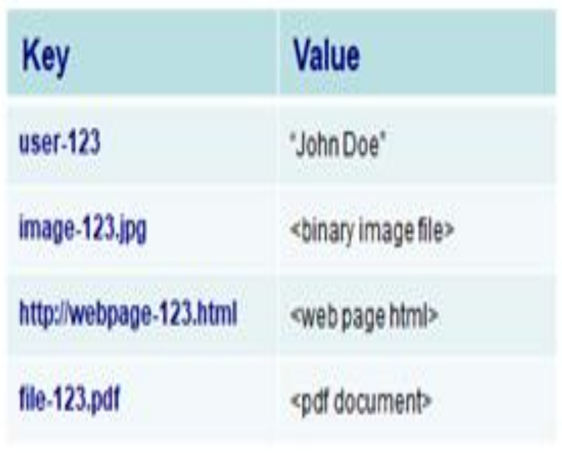
Examples of Analytical Database are

- Market data- historicaland volume data for financial markets for testing trading strategies.
- Transactional data — Historical transactions that can include purchasing patterns for improved marketing.
- Sensor data — Historical data from sensors that monitor situations like the weather.
- Natural language data — Study of social media posts for research purposes.
- Process data — Study of processes to better understand logistics and find bottlenecks.
- Machine data — Software and hardware-generated data from products to improve efficiency

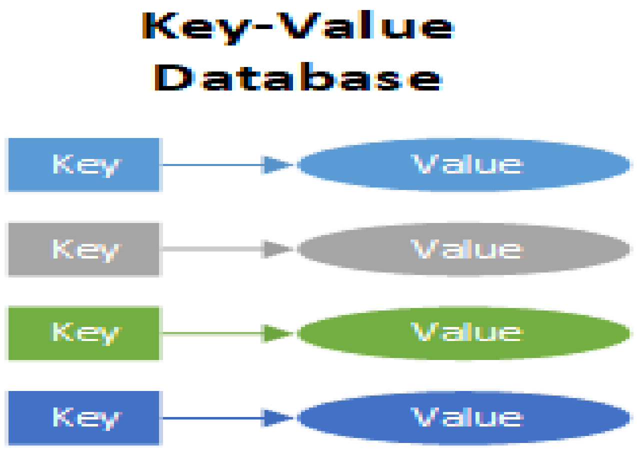
KEY VALUE

What is Key Value?

A Key Value also known as a key-value store, A key-value database is a simple database that contains a simple string (the key) that is always unique, and an arbitrary large data field (the value). Key-value stores have no query language, but they do provide a way to add and remove key-value pairs. Additionally, values cannot be queried or searched upon. Only the key can be queried. Within a key-value database, only three functions can be executed (put, get, delete).



Key	Value
user-123	'John Doe'
image-123.jpg	<binary image file>
http://webpage-123.html	<web page html>
file-123.pdf	<pdf document>



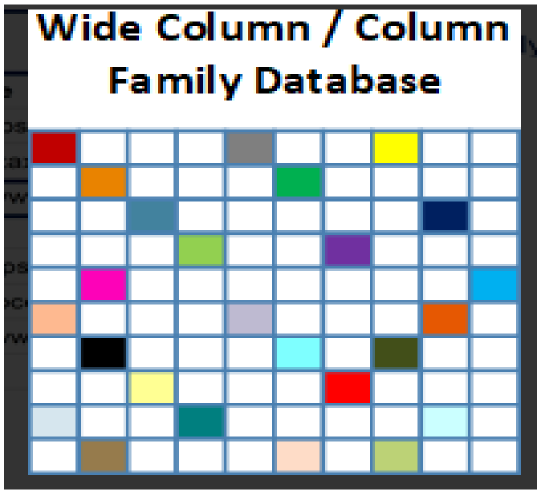
- **Put:** Adds a new key-value pair and updates the value if the key is already present.
- **Get:** Returns the value for any given key.
- **Delete:** Removes a key-value pair.

One of the benefits of the **key-value database** is that data of any data type can be stored in the value field including a binary large object (BLOB) value. Additionally, the key-value database is like a dictionary, as a dictionary has a list of words and each word has one or more definitions with various length. And like a dictionary, the key-value database is uniquely indexed by the key field. Thus, rapid retrieval of values can occur regardless of the number of records / items within the database. Key-value databases work in a very different fashion from traditional relational database management systems (RBMS). Traditionally RDBMS define data structures in the database as a series of tables containing fields with well-defined data types. In contrast, key-value databases treat data values as a single opaque collection, which may have different fields for every record. This offers considerable flexibility and more closely follows modern concepts like object-oriented programming. Because optional values are not represented by placeholders or input parameters, key-value databases often use far less memory to store the same database, which can lead to significant performance gains.

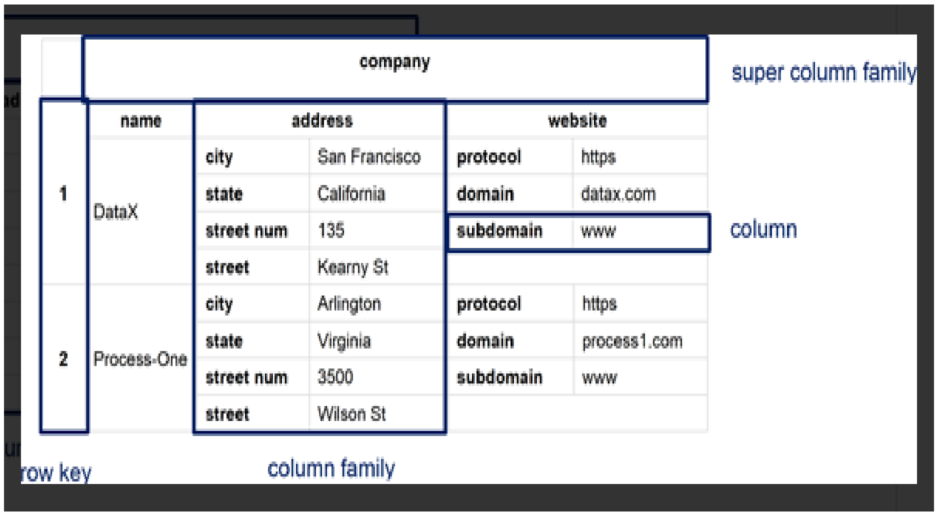
COLUMN-FAMILY

What is Column-Family Database?

The column family databases are NoSQL databases that store data in records with an ability to hold very large numbers of dynamic columns. Columns can contain null values and data with different data types. In addition, data is stored in cells grouped in columns of data rather than as rows of data. Columns are logically grouped into column families. Column families can contain a virtually unlimited number of columns that can be created at run-time or while defining the schema. And column families are groups of similar data that is usually accessed together. Additionally, column families can be grouped together as super column families.



The basis of the architecture of wide column / column family databases is that data is stored in columns instead of rows as in a conventional relational database management system (RDBMS). And the names and format of the columns can vary from row to row in the same table. Subsequently, a wide column database can be interpreted as a two-dimensional key-value. Wide column databases do often support the notion of column families that are stored separately. However, each such column family typically contains multiple columns that are used together, like traditional RDBMS tables. Within a given column family, all data is stored in a row-by-row fashion, such that the columns for a given row are stored together, rather than each column being stored separately.

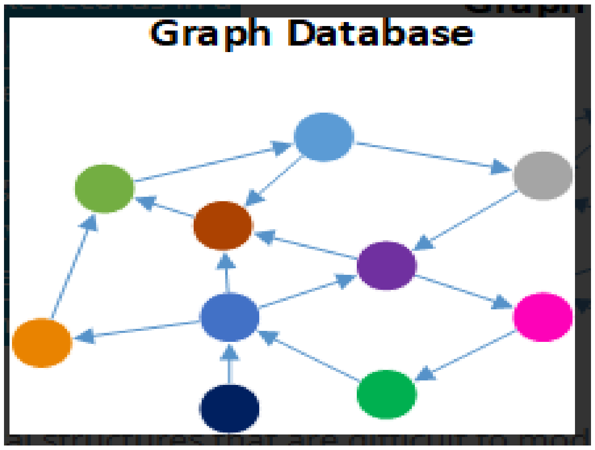


	company				
	name	address		website	
1	DataX	city	San Francisco	protocol	https
		state	California	domain	datax.com
		street num	135	subdomain	www
		street	Kearny St		
2	Process-One	city	Arlington	protocol	https
		state	Virginia	domain	process1.com
		street num	3500	subdomain	www
		street	Wilson St		

GRAPH

What is Graph Database?

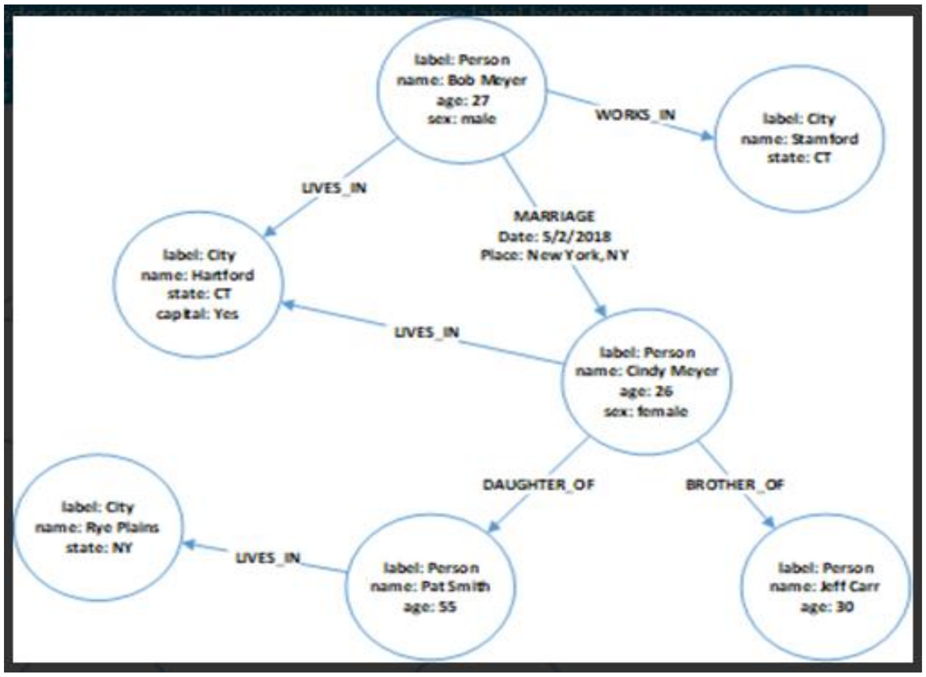
A graph database is a NoSQL database that organizes data as nodes, which are like records in a relational database, and relationships, which represent connections between nodes. Because the graph system stores the relationship between nodes, it can support richer representations of data relationships. Relationships are the key concept in graph databases, representing an abstraction that is not directly implemented in RDBMS or other NoSQL databases. Primarily, graph databases are applied in systems that share relationships between values, such as social networks, reservation systems, fraud detection, or customer relationship management systems. And graph databases address significant limitations of existing relational database management systems (RDBMS).



Graph databases contain four types of data fields (nodes, relationships, properties, & labels):

• **Nodes:** Objects that represent data entities or instances such as people, businesses, accounts, products or any other item to be tracked. They are roughly the equivalent of the record or row in a relational database, or the document in a document-store database. Each node contains several pieces of information that go together. For example, a single node might include a product name, description, price and product code. Another might have information about a customer, such as name and account number.
• **Relationships:** Objects that describe how the nodes relate to each other. Relationships represent the connections, edges, or lines between nodes to other nodes. A relationship connects two nodes and enables users to find related nodes. A relationship always has a source node and a target node that provides the direction of the arrow. Meaningful patterns can emerge when examining the connections and interconnections of nodes.
• **Properties:** Additional attributes of both nodes and relationships that are represented as additional key-value pairs. Properties store relevant data about the node or relationship with the entity it describes. Examples of priorities for a node with a label of person include name, age, address, & date of birth. Relationships usually have properties including time, distance, cost, rating or weights which are also stored as key-value pairs.
• **Labels:** Named graph construct that is used to group nodes into sets, and all nodes with the same label belongs to the same set. Many database queries can work with these sets instead of

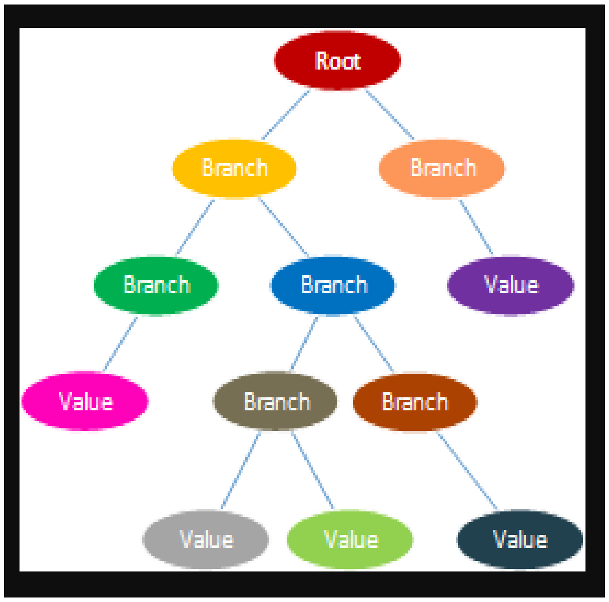
the whole graph, making queries easier to write and more efficient to execute. A node may be labeled with any number of labels, including none, making labels an optional addition to the graph.



DOCUMENT

What is Document Database?

A **document database**, also called a document store or document-oriented database, is a NoSQL database used for storing, retrieving, and managing semi-structured data. Unlike traditional relational database management systems (RDBMS), the data model in a document database is not structured in a table format of rows and columns. A document database uses documents as the structure for storage and queries. Subsequently a document database aggregates data from documents and stores the documents a searchable and organized format. The schema of document databases can vary, providing far more flexibility for data modeling than RDBMS. In this case, the term “document” may refer to a MS Word, MS Excel, MS PowerPoint or Adobe PDF document but is commonly a block of extensible markup language (XML) or javascript object notation (JSON) code and values. Instead of columns with names and data types that are used in RDBMS, a document contains a description of the data type and the value for that description. Each document stored within a document database can have the same or different structure.

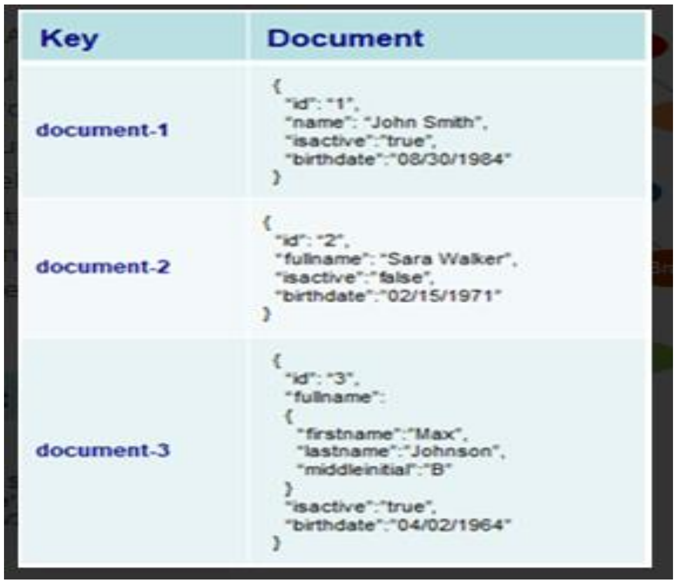


Document databases use a tree-like structure that begins with a root node. And beneath the root node, there is a sequence of branches, sub-branches, and values. Subsequently, each branch has a related path expression that shows to navigate from the root of the tree to any given branch, sub-branch, or value. Most document stores group documents together within document collections. And these collections are similar in look and feel to the directory structure in a Windows or UNIX/Linux file system. Document collections can be used to navigate document hierarchies, logically group similar documents, and to store business rules including permissions, indexes, and triggers. Additionally, collections can contain other collections.

Documents within document databases are identified using a unique key, which contains a simple identifier. The key usually contains either a string, a URI, or a path. And the key can be used to retrieve the document from the database. Typically, the database retains an index on the key to speed up document retrieval. And sometimes, the key is used to create or insert the document into the database.

A key advantage of a document database is that all values within the document are automatically indexed when a new document is insert into the database. That means that every value within the document can be searched upon. This also means that if a user knows any property of the document, all documents with the same property can be easily retrieved. And even if the document structure is complex, a document store search provides an easy way to select either an entire document or a sub-set of a document. Additionally, document database searches can tell the user whether the search item is included within a document as well as the search items exact location utilizing the document path.

To add additional types of data to a document database, there is no need to modify the entire database schema as is with a RDBMS. Data can simply be added by adding objects to the database. Further document databases utilize internal structure within documents in order to extract metadata that the database engine uses for further optimization and query performance. Unlike traditional RDBMS, some document databases prioritize write availability over strict data consistency. This ensures that writes will always be fast even if there is a failure in one portion of the hardware or network.



Key	Document
document-1	{ "id": "1", "name": "John Smith", "isactive":"true", "birthdate":"08/30/1984" }
document-2	{ "id": "2", "fullname": "Sara Walker", "isactive":"false", "birthdate":"02/15/1971" }
document-3	{ "id": "3", "fullname": { "firstname":"Max", "lastname":"Johnson", "middleinitial":"B" } "isactive":"true", "birthdate":"04/02/1964" }