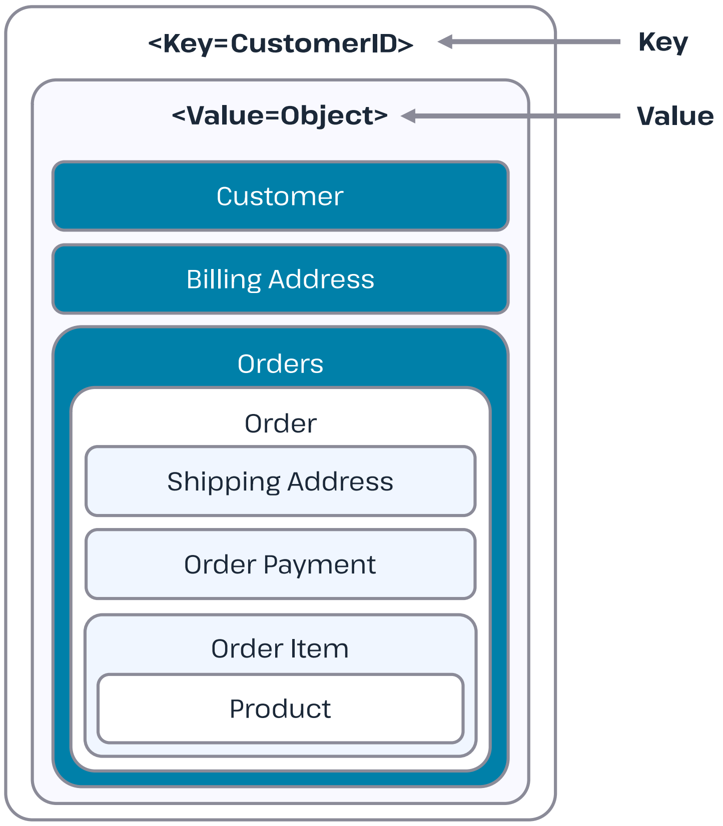
Data and Middleware Technologies

A **key-value store**, or key-value database, is a type of data storage software program that stores data as a set of unique identifiers, each of which have an associated value. This data pairing is known as a "key-value pair." The unique identifier is the "key" for an item of data, and a value is either the data being identified or the location of that data.

Key	Value
123	123 Main St.
126	(805) 477-3900



An example of a key-value store

The key could be anything, depending on restrictions imposed by the database software, but it needs to be unique in the database so there is no ambiguity when searching for the key and its value. The value could be anything, including a list or another key-value pair. Some database software allows you to specify a data type for the value.

In traditional relational database design, data is stored in tables composed of rows and columns. The database developer specifies many attributes of the data to be stored in the table upfront. This creates significant opportunities for optimizations such as data compression and performance around aggregations and data access, but also introduces some inflexibility.

Key-value stores, on the other hand, are typically much more flexible and offer very fast performance for reads and writes, in part because the database is looking for a single key and is returning its associated value rather than performing complex aggregations.

What does a key-value pair mean?

A key-value pair is two pieces of data associated with each other. The key is a unique identifier that points to its associated value, and a value is either the data being identified or a pointer to that data.

A key-value pair is the fundamental data structure of a key-value store or key-value database, but key-value pairs have existed outside of software for much longer. A telephone directory is a good example, where the key is the person or business name, and the value is the phone number. Stock trading data is another example of a key-value pair. In this case, you may have a key associated with values for the stock ticker, whether the trade was a buy or sell, the number of shares, or the price of the trade.

Key-value store advantages

There are a few advantages that a key-value store provides over traditional row-column-based databases. Thanks to the simple data format that gives it its name, a key-value store can be very fast for read and write operations. And key-value stores are very flexible, a valued asset in modern programming as we generate more data without traditional structures.

Also, key-value stores do not require placeholders such as “null” for optional values, so they may have smaller storage requirements, and they often scale almost linearly with the number of nodes.

Key-value database use cases

The advantages listed above naturally lend themselves to several popular use cases for key-value databases.

- Web applications may store user session details and preference in a key-value store. All the information is accessible via user key, and key-value stores lend themselves to fast reads and writes.
- Real-time recommendations and advertising are often powered by key-value stores because the stores can quickly access and present new recommendations or ads as a web visitor moves throughout a site.
- On the technical side, key-value stores are commonly used for in-memory data caching to speed up applications by minimizing reads and writes to slower disk-based systems. Hazelcast is an example of a technology that provides an in-memory key-value store for fast data retrieval.

Distributed key-value store

A distributed key-value store builds on the advantages and use cases described above by providing them at scale. A distributed key-value store is built to run on multiple computers working together, and thus allows you to work with larger data sets because more servers with more memory now hold the data. By distributing the store across multiple servers, you can increase processing performance. And if you leverage replication in your distributed key-value store, you increase its fault tolerance. Hazelcast is an example of a technology that provides a distributed key-value store for larger-scale deployments. The “IMap” data type in Hazelcast, similar to the “Map” type in Java, is a key-value store stored in memory. Unlike the Java Map type, Hazelcast IMaps are stored in memory in a distributed manner across the collective RAM in a cluster of computers, allowing you to store much more data than possible on a single computer. This gives you quick lookups with in-memory speeds while also retaining other important capabilities such as high availability and security.