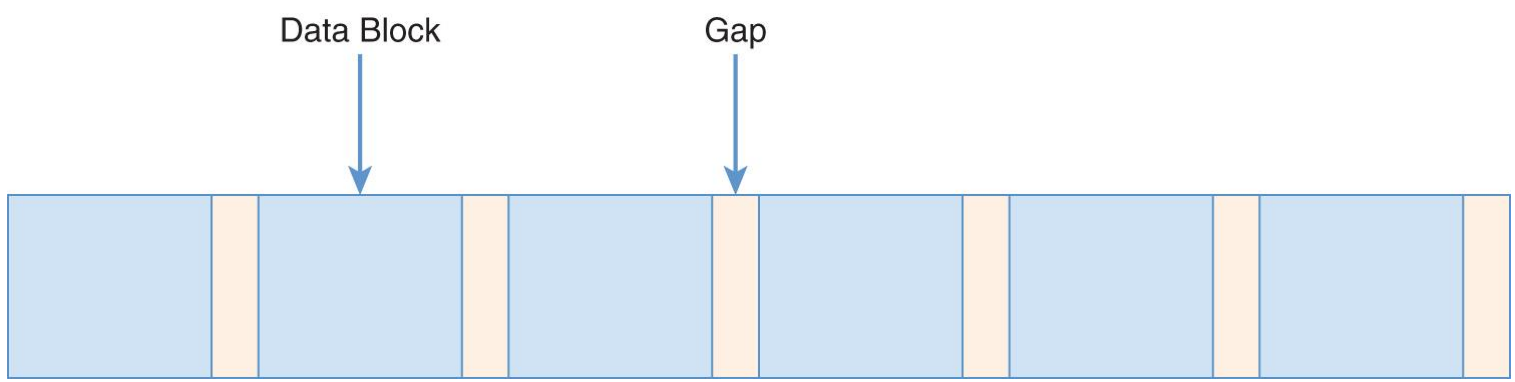

Figure 1.1 Magnetic tapes store data in sequential blocks.

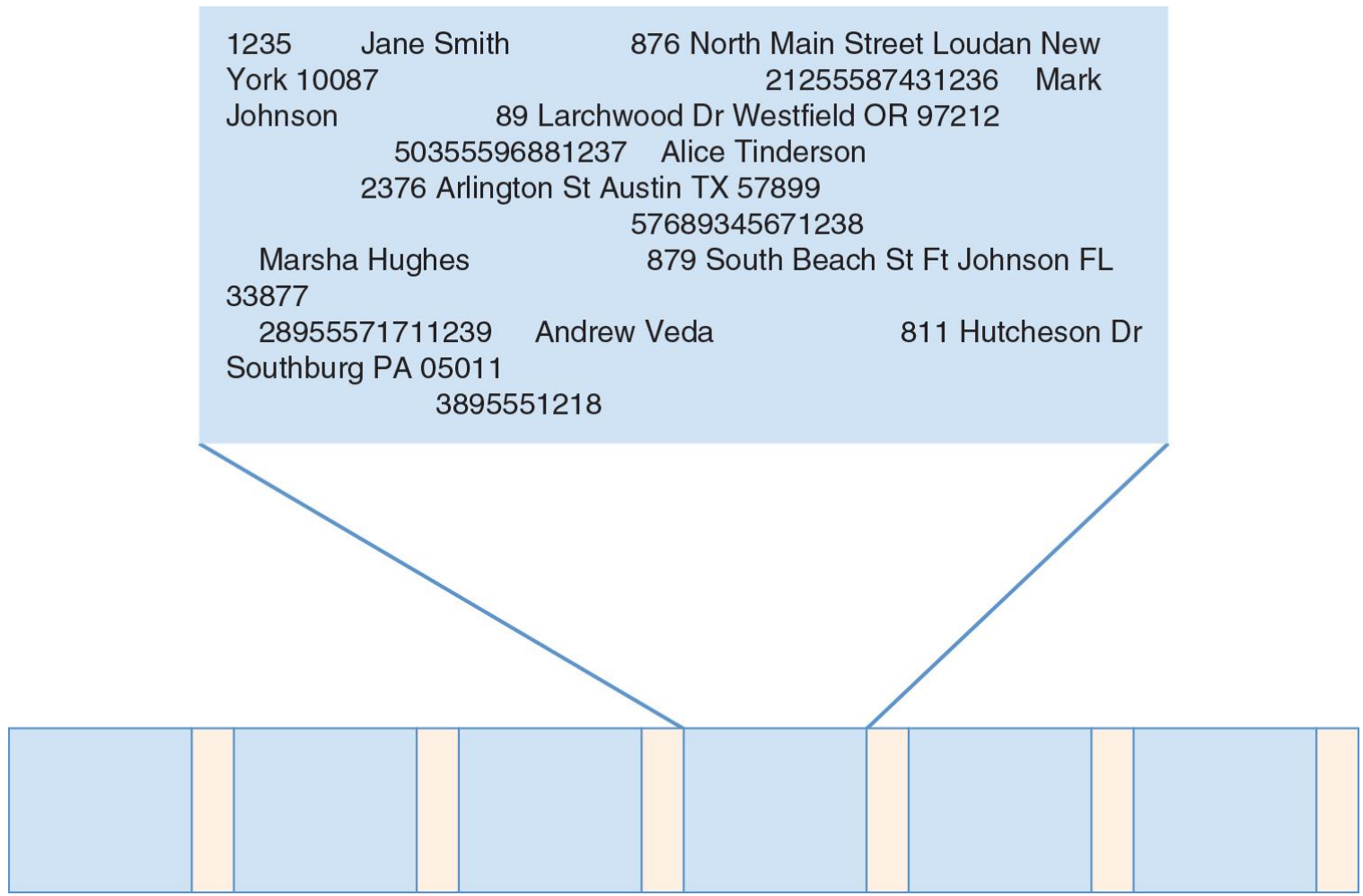

Figure 1.2 A block is a chunk of data read by tape or disk drive in a single read operation.

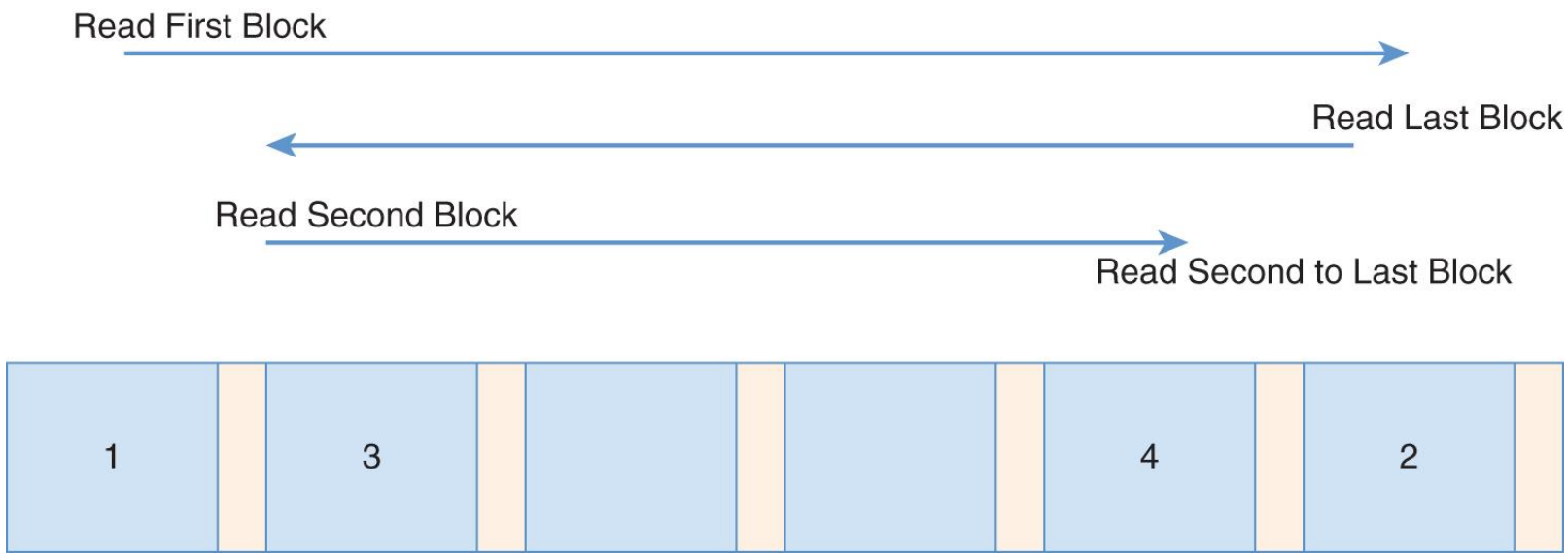

Figure 1.3 Random access to blocks on tape can take more time than sequential access because there can be more tape movement relative to the amount of data read.

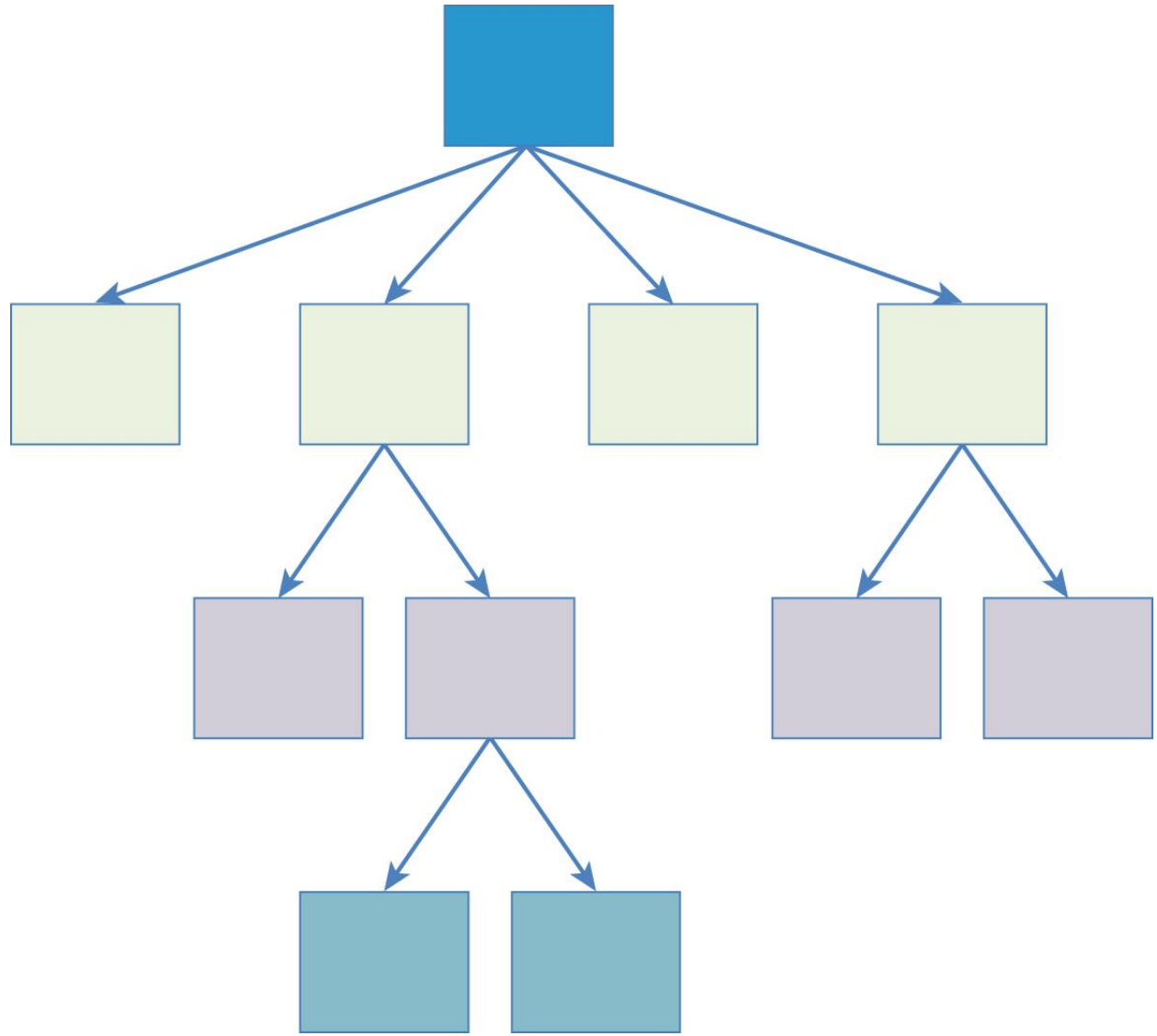

Figure 1.4 The hierarchical model is organized into a set of parent-child relations.

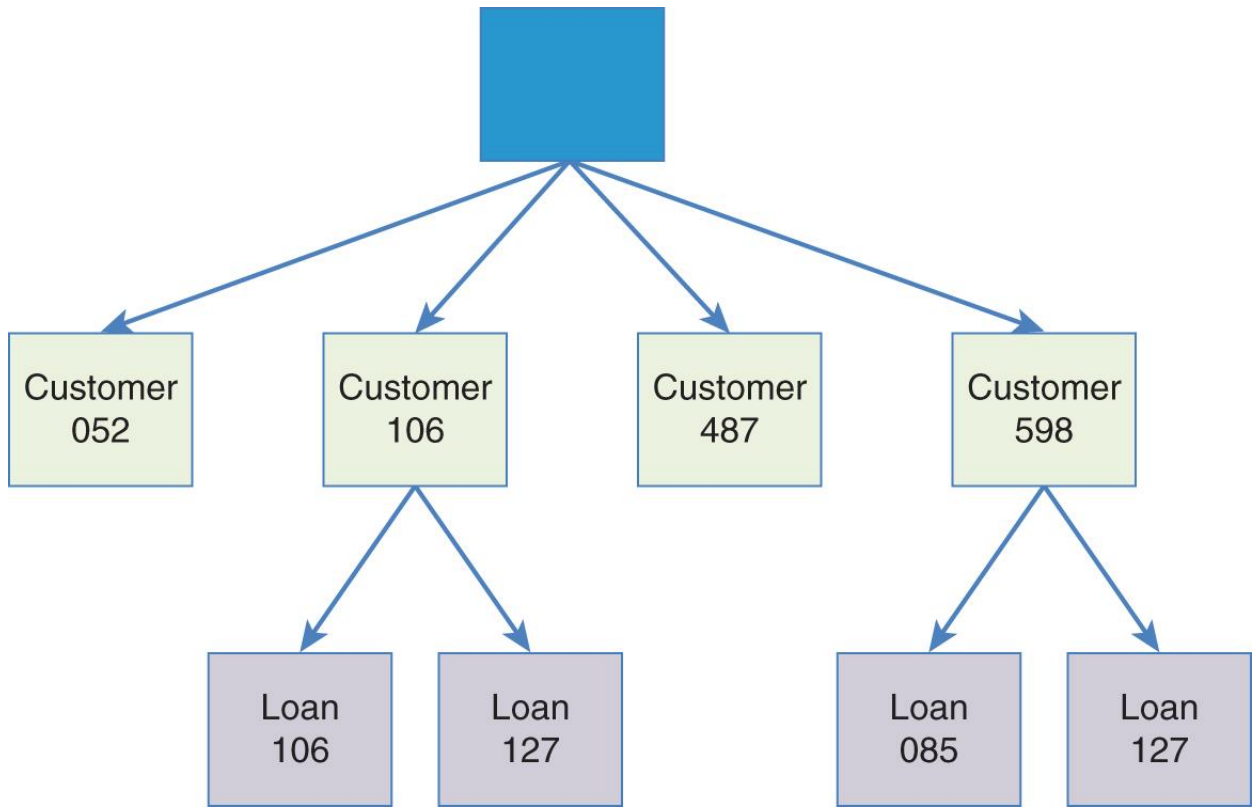

Figure 1.5 A hierarchical data model for a loan management database.

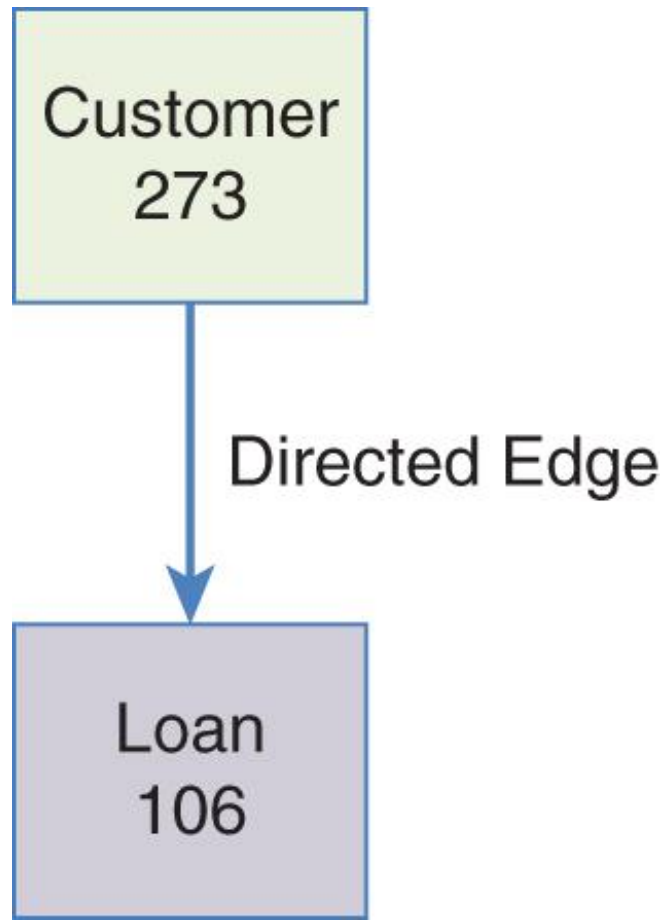

Figure 1.6 A parent-child relationship is represented by a directed edge.

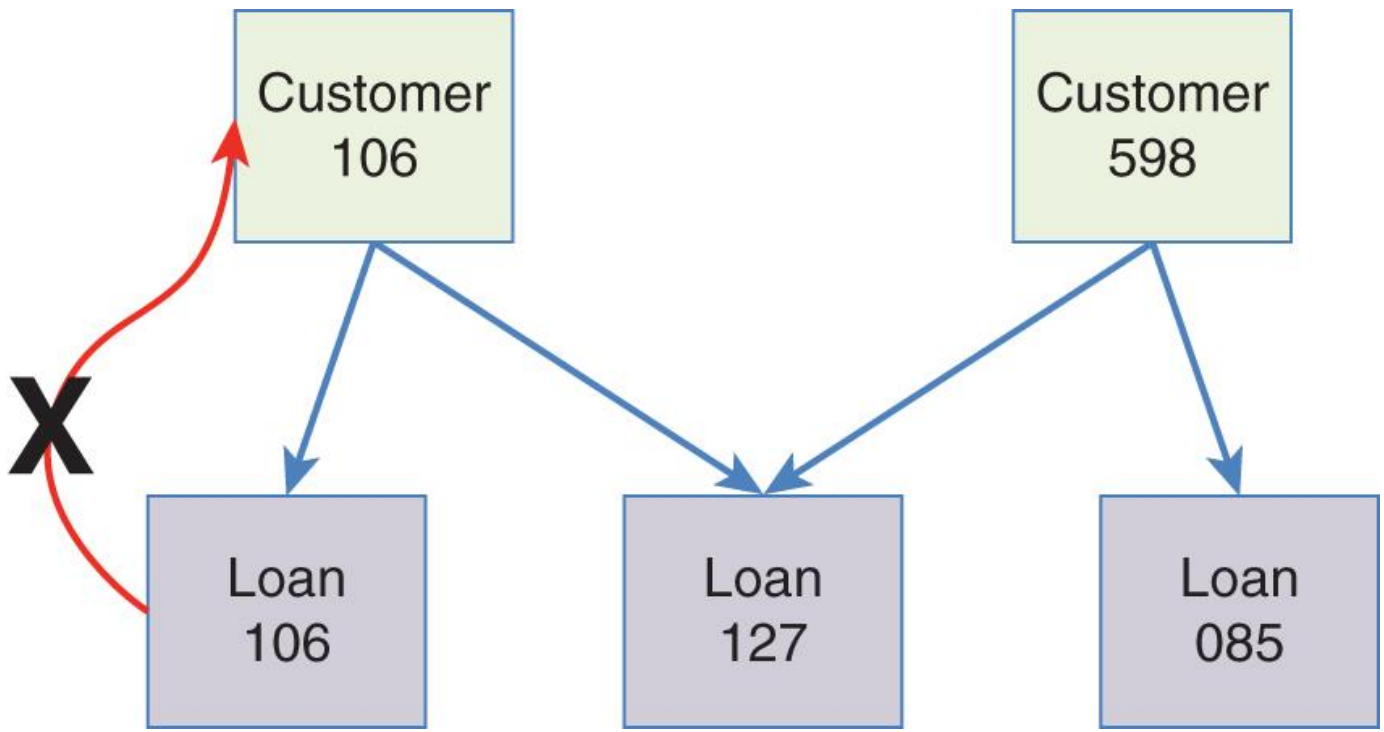

Figure 1.7 This graph has cycles and, therefore, is not a directed acyclic graph and not a model of a network data management system.

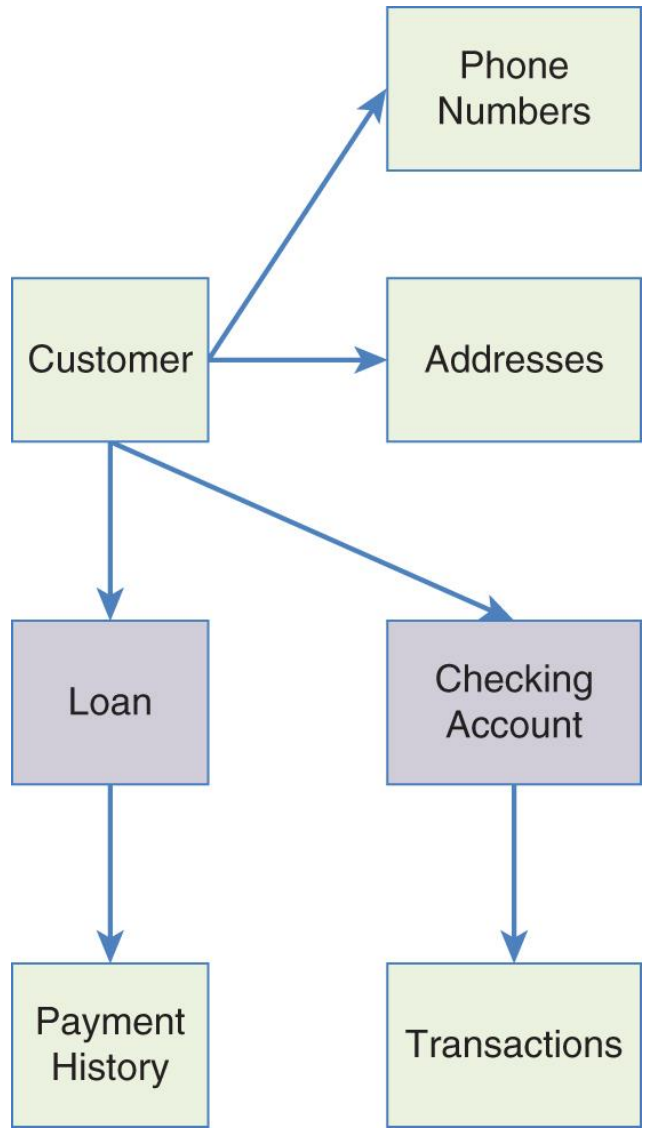

Figure 1.8 A simple network schema shows which entities can link to other entities.

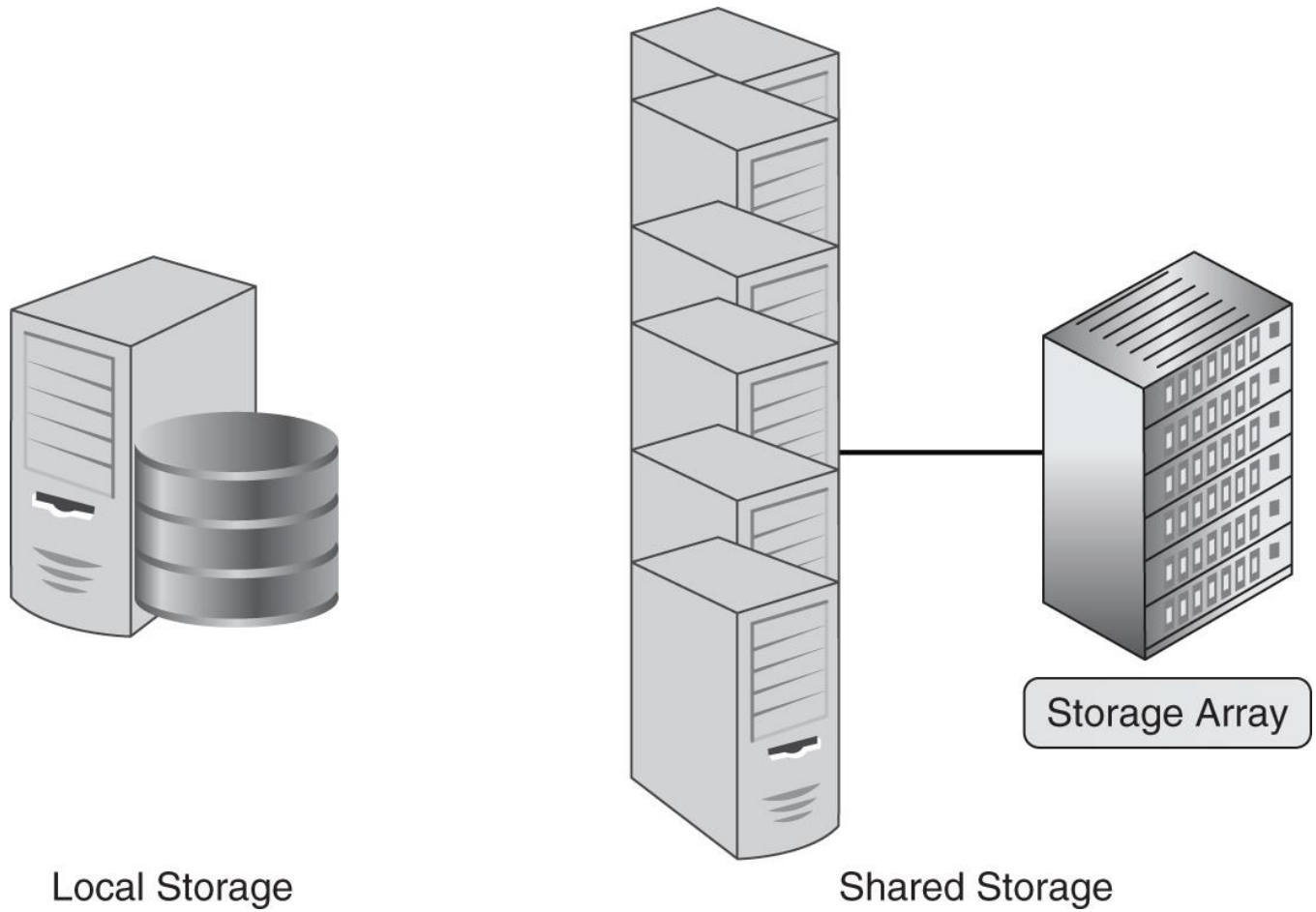

Figure 1.9 Local storage versus shared network storage.

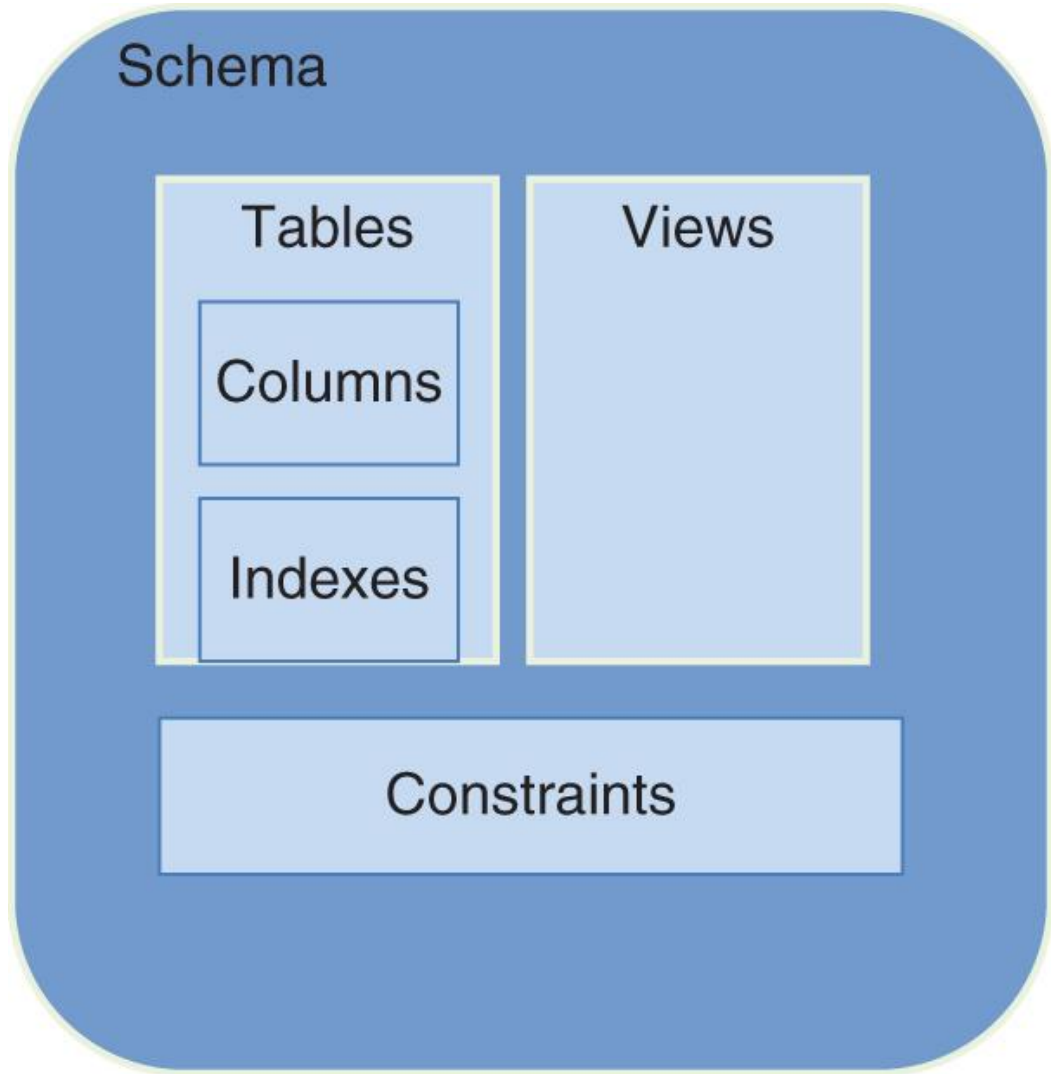

Figure 1.10 Data structures managed by a data dictionary.

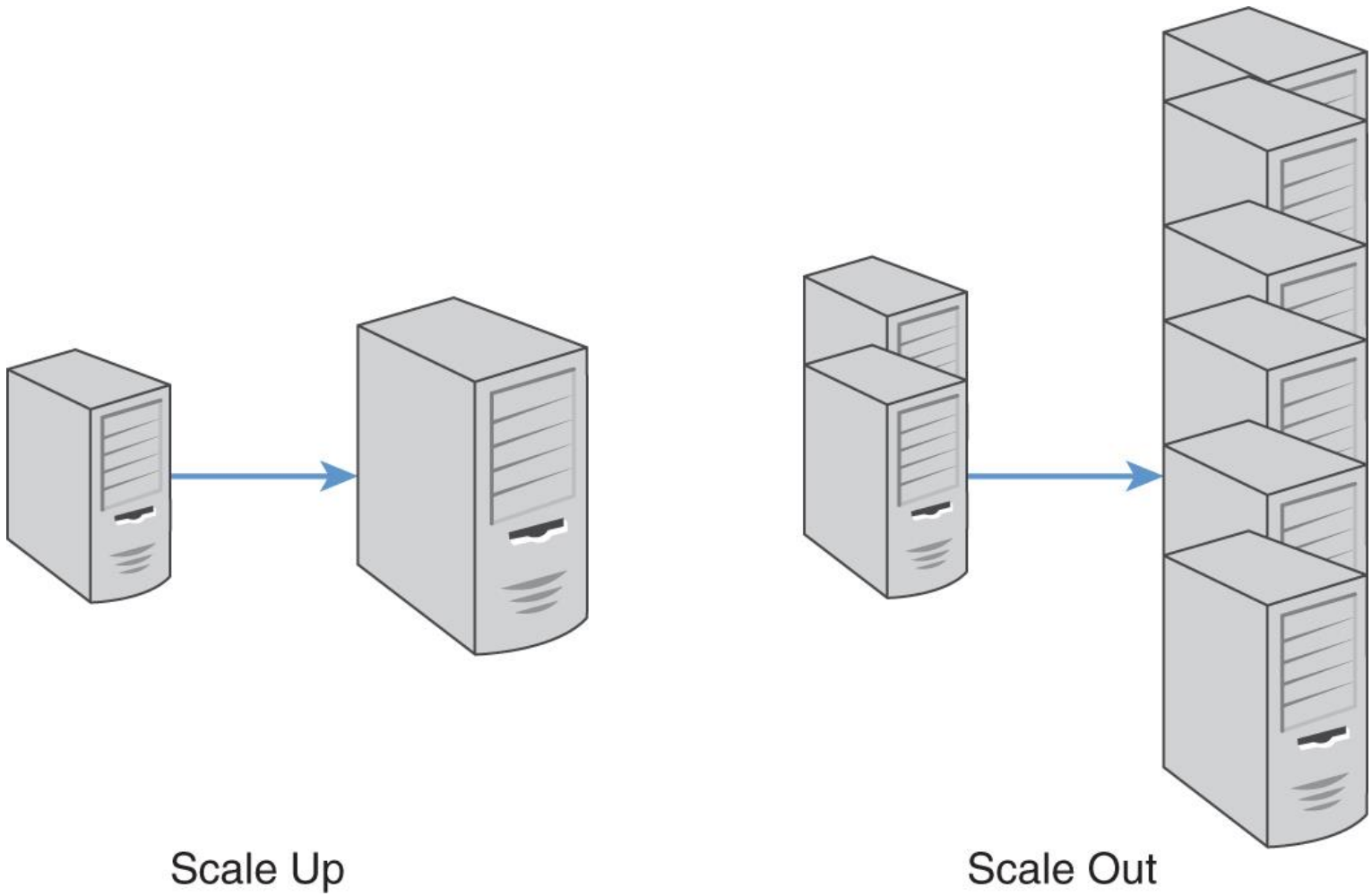

Figure 1.11 Scaling up versus scaling out.

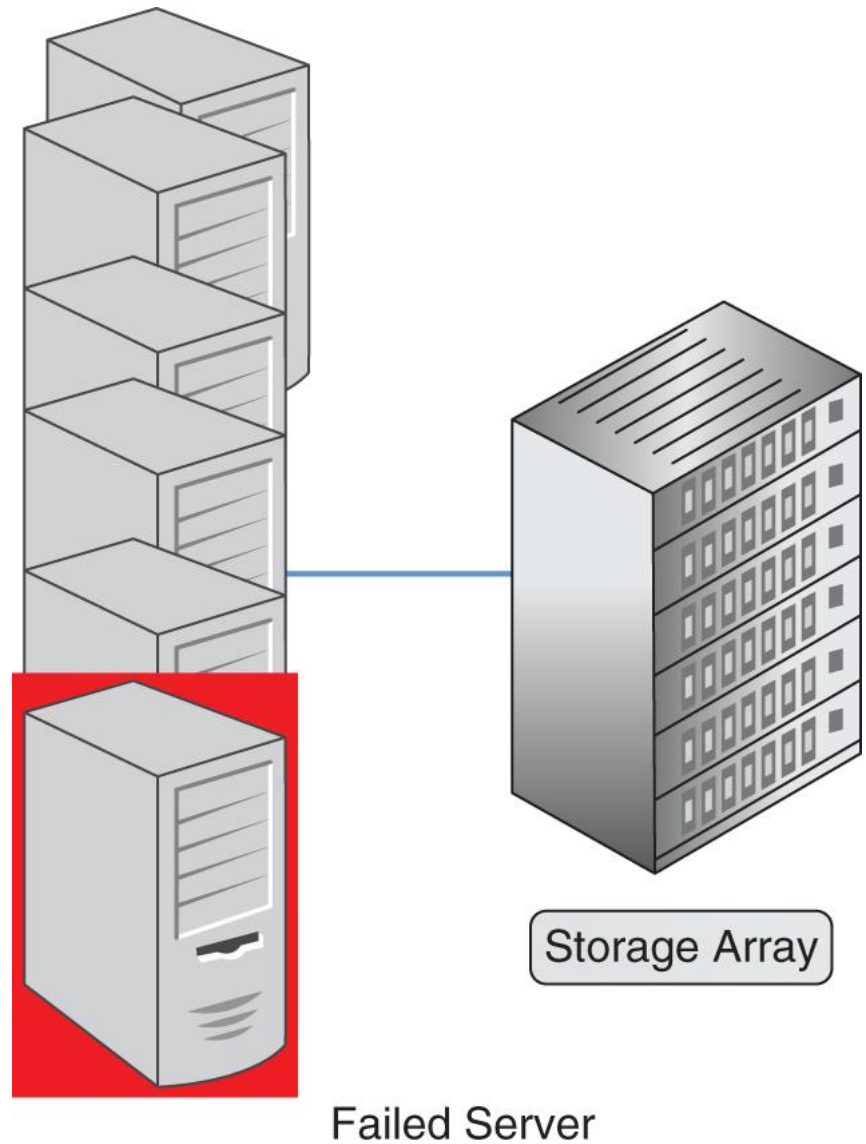

Figure 1.12 High-availability NoSQL clusters run multiple servers. If one fails, the others can continue to support applications.

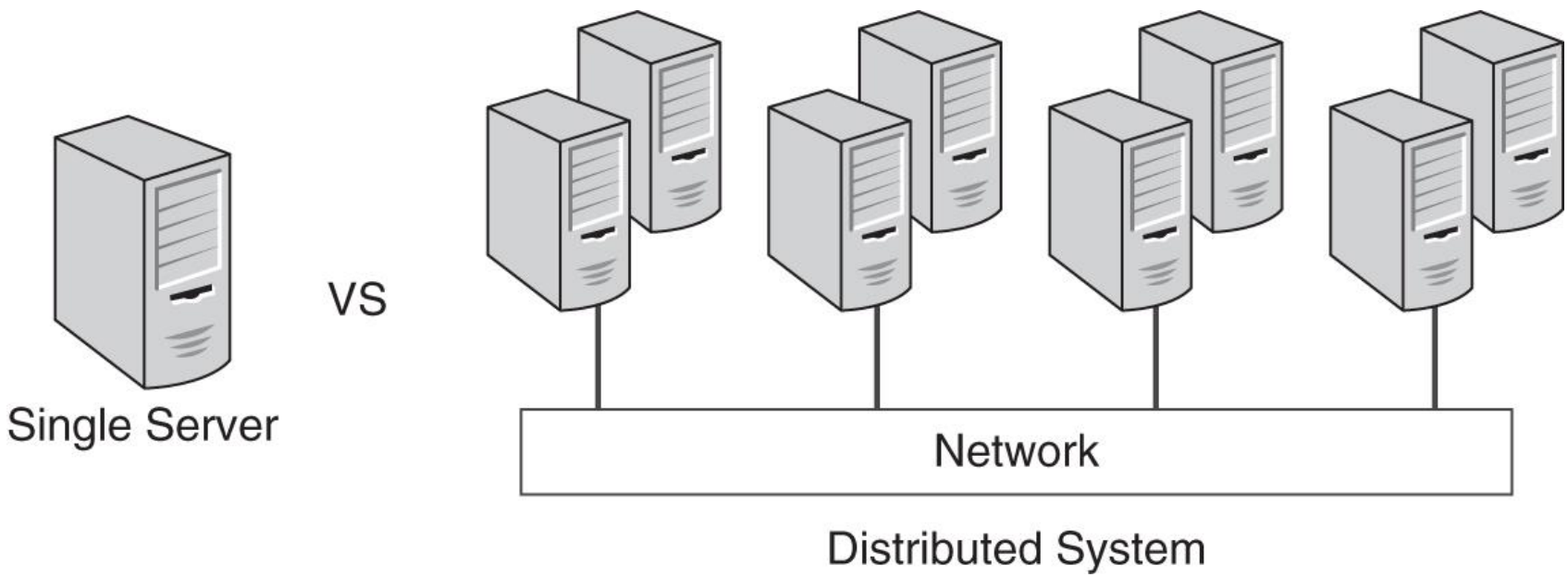

Figure 2.1 Single server versus distributed system.

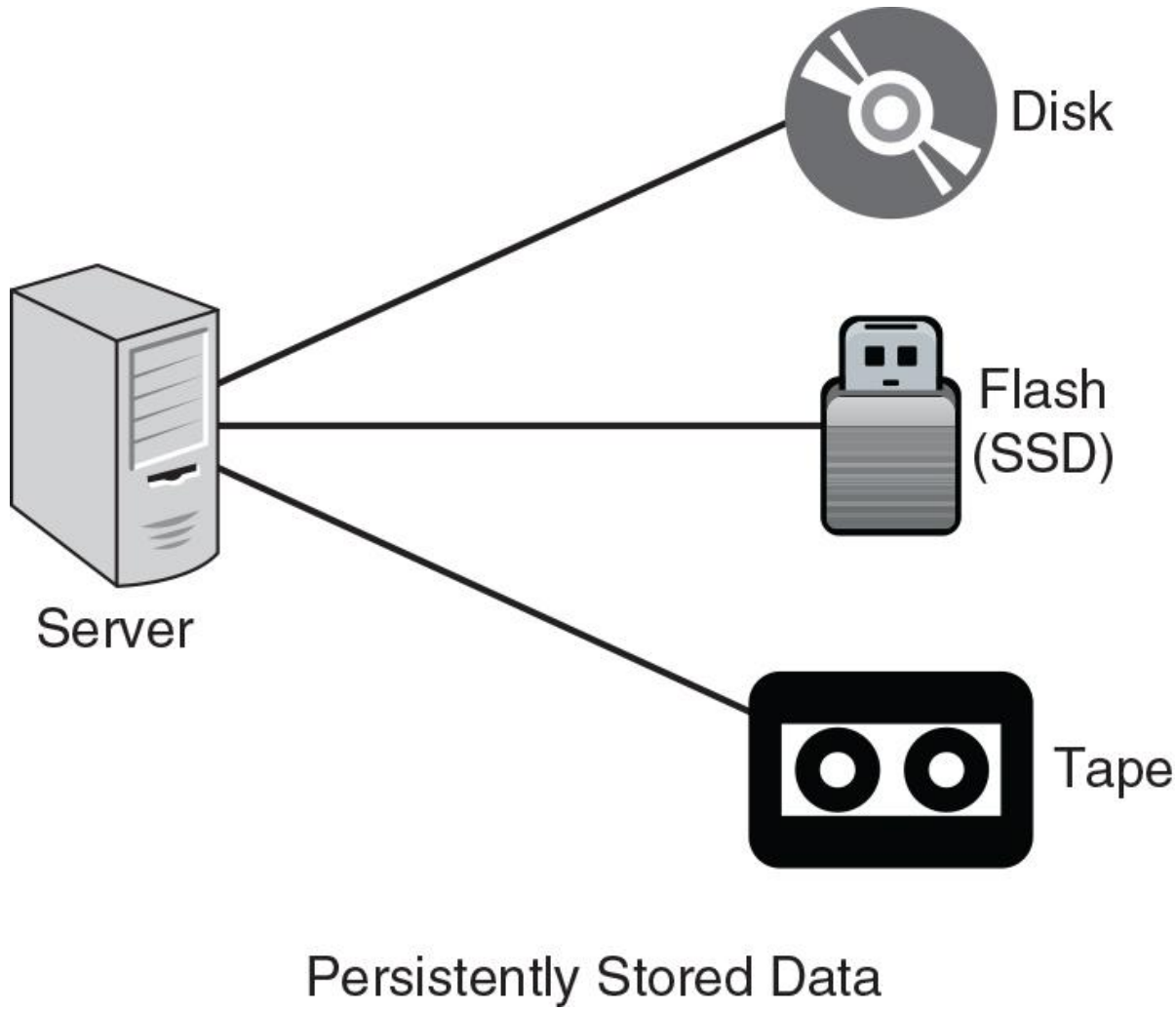

Figure 2.2 Persistently stored data is stored on disk, flash, or other long-term storage medium.

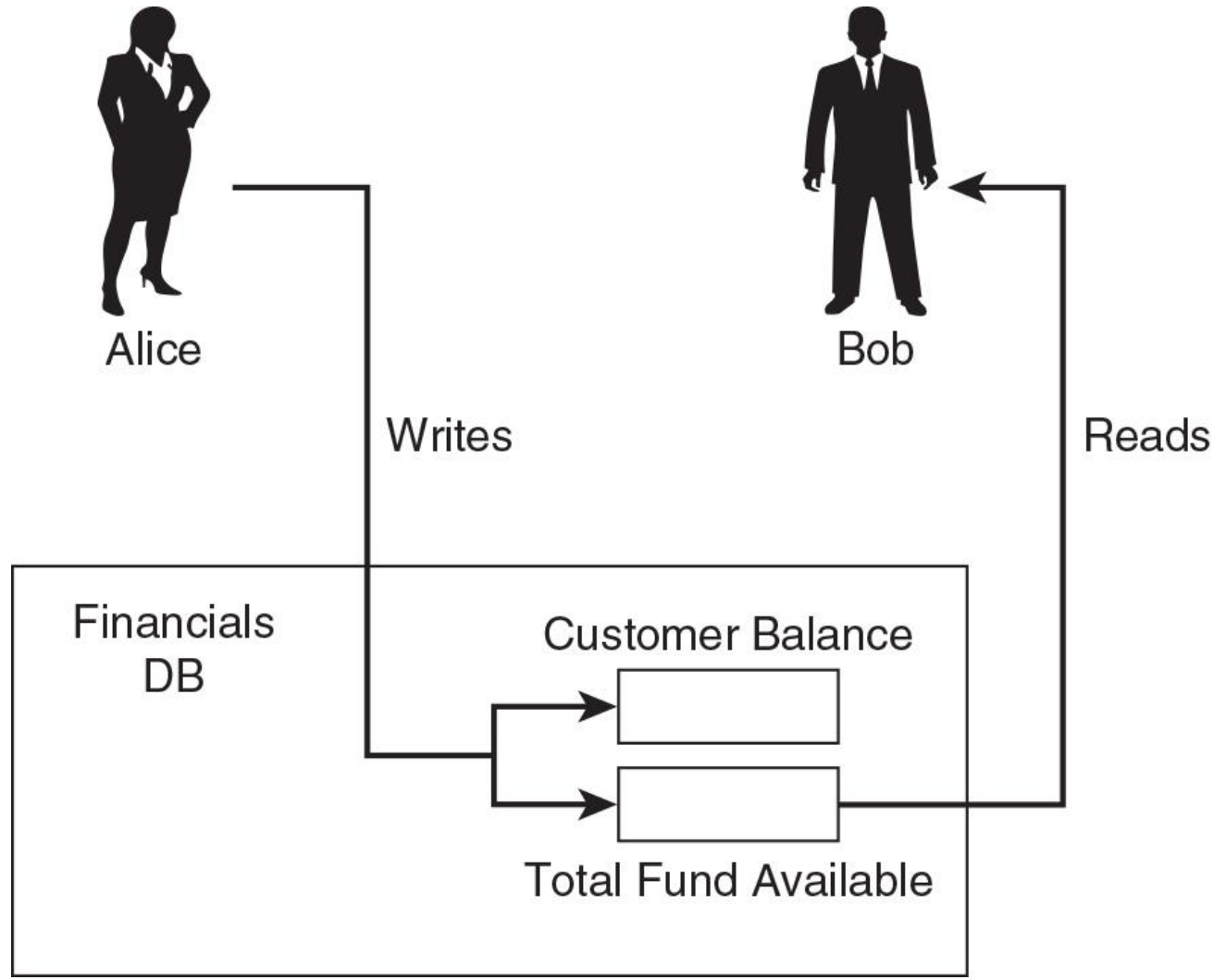

Figure 2.3 Data should reflect a consistent state.

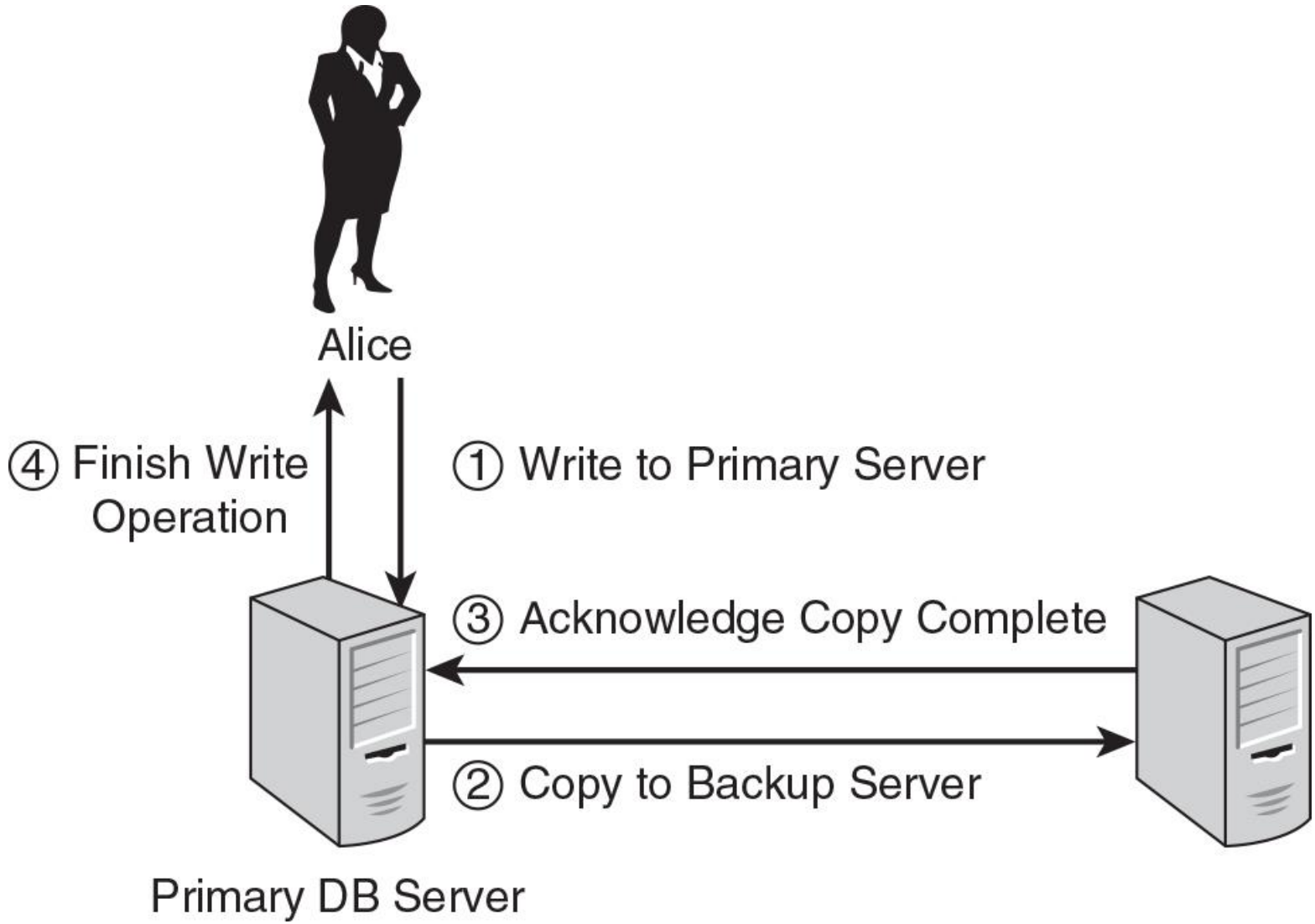

Figure 2.4 Two-phase commits are complete only when operations on both databases are complete.

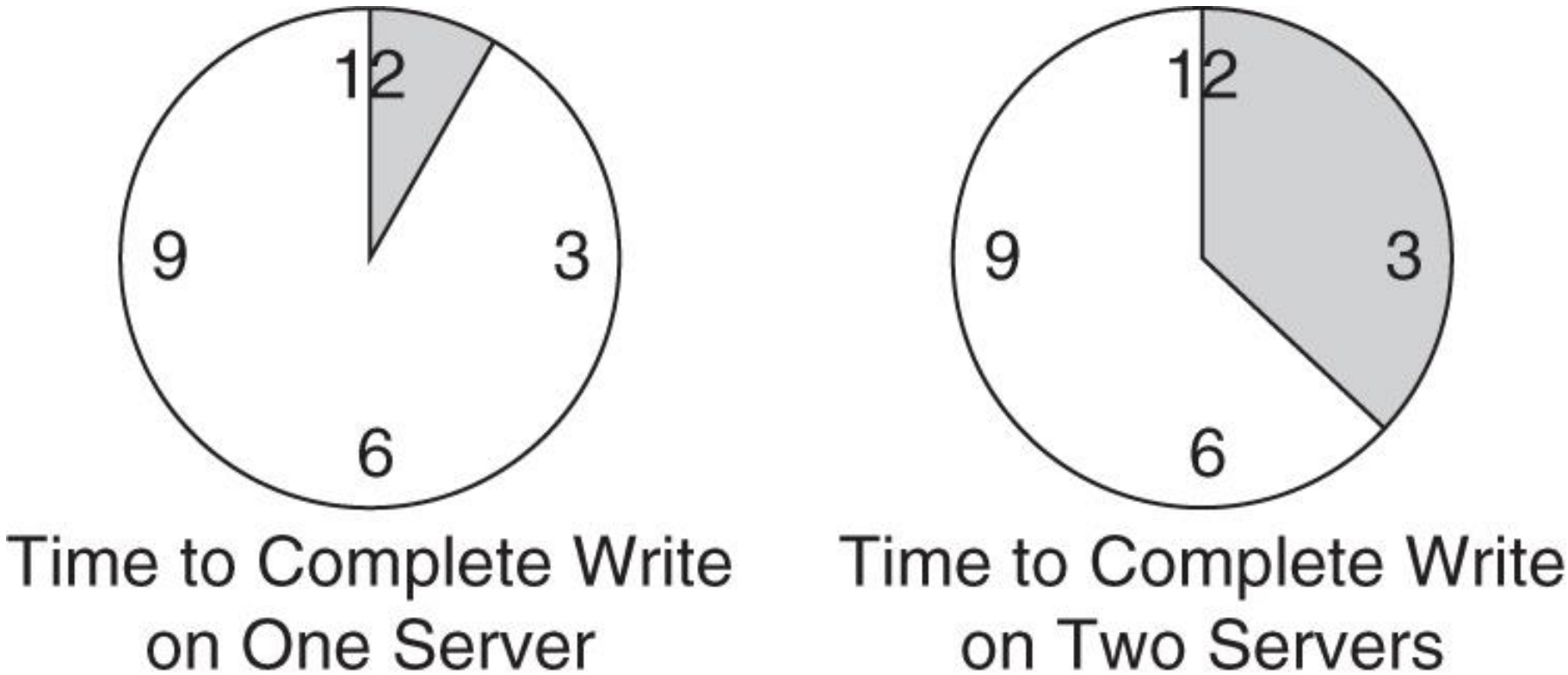

Figure 2.5 Consistency and availability require more time to complete transactions in high-availability environments.

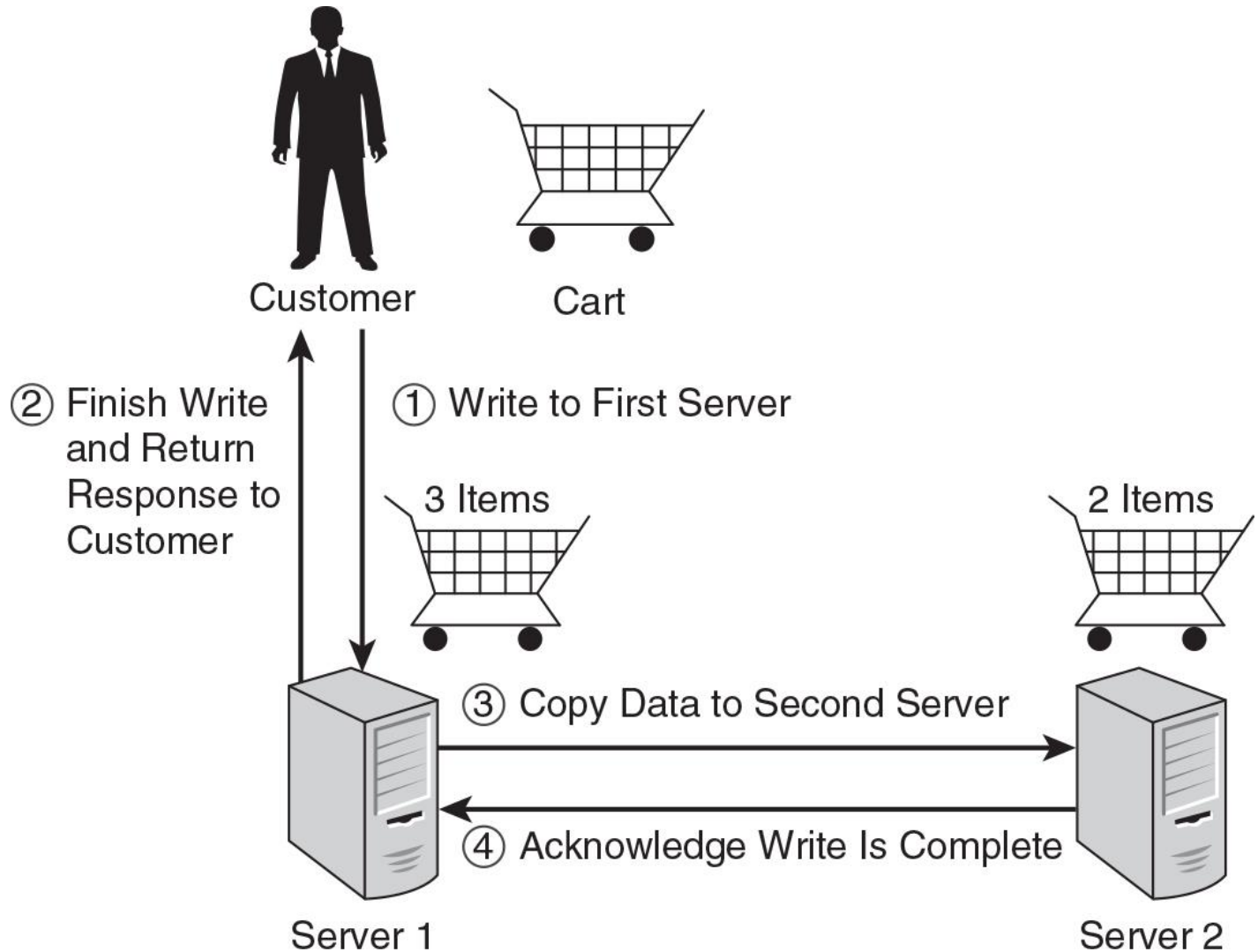

Figure 2.6 Data structures, such as shopping carts, can be inconsistent for short periods of time without adversely affecting system effectiveness. In this example, Server 2 is inconsistent with Server 1 until step 3 is complete.

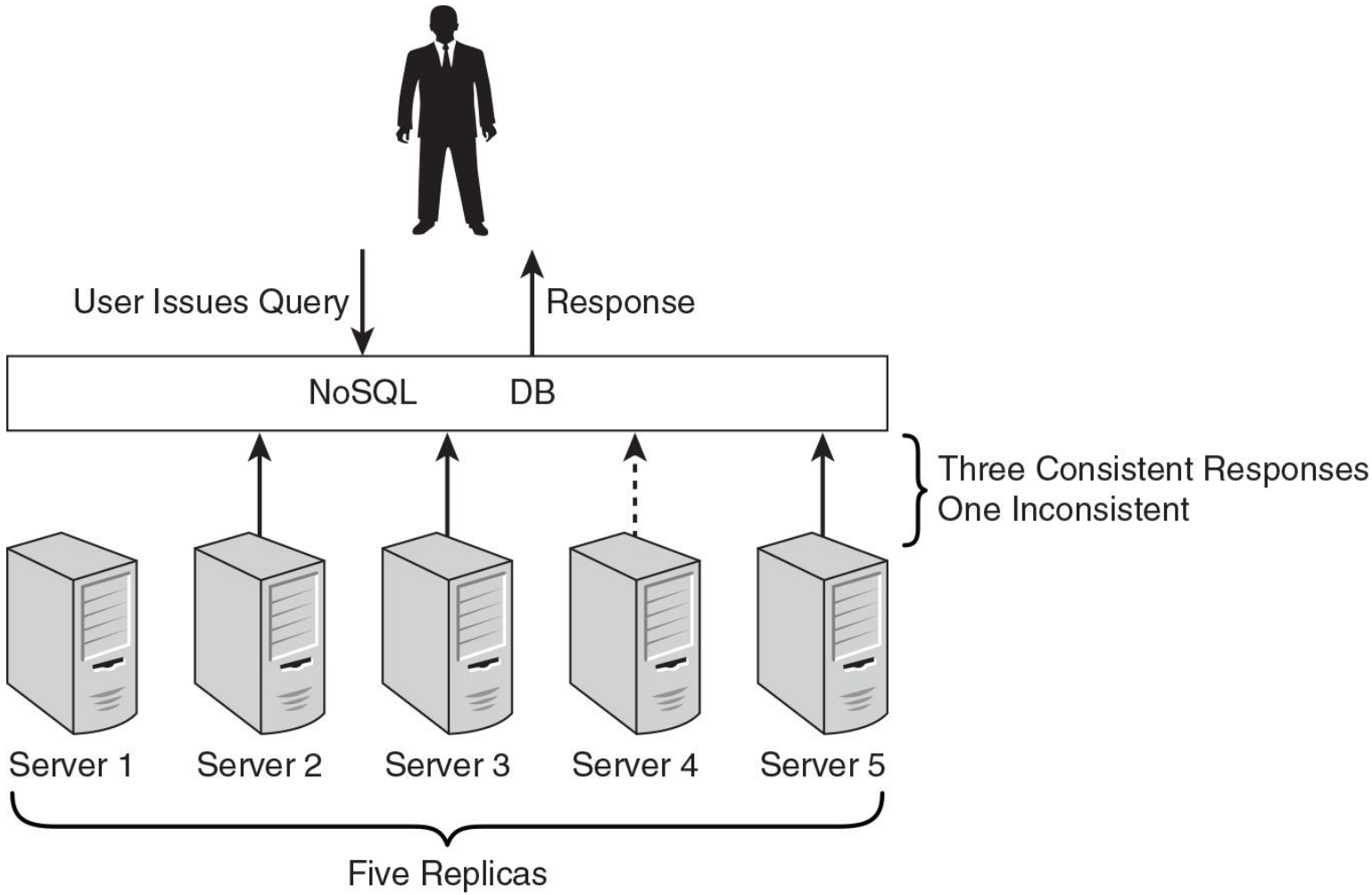

Figure 2.7 NoSQL databases can mitigate the risk of inconsistent data by having servers vote on the correct response to a query.

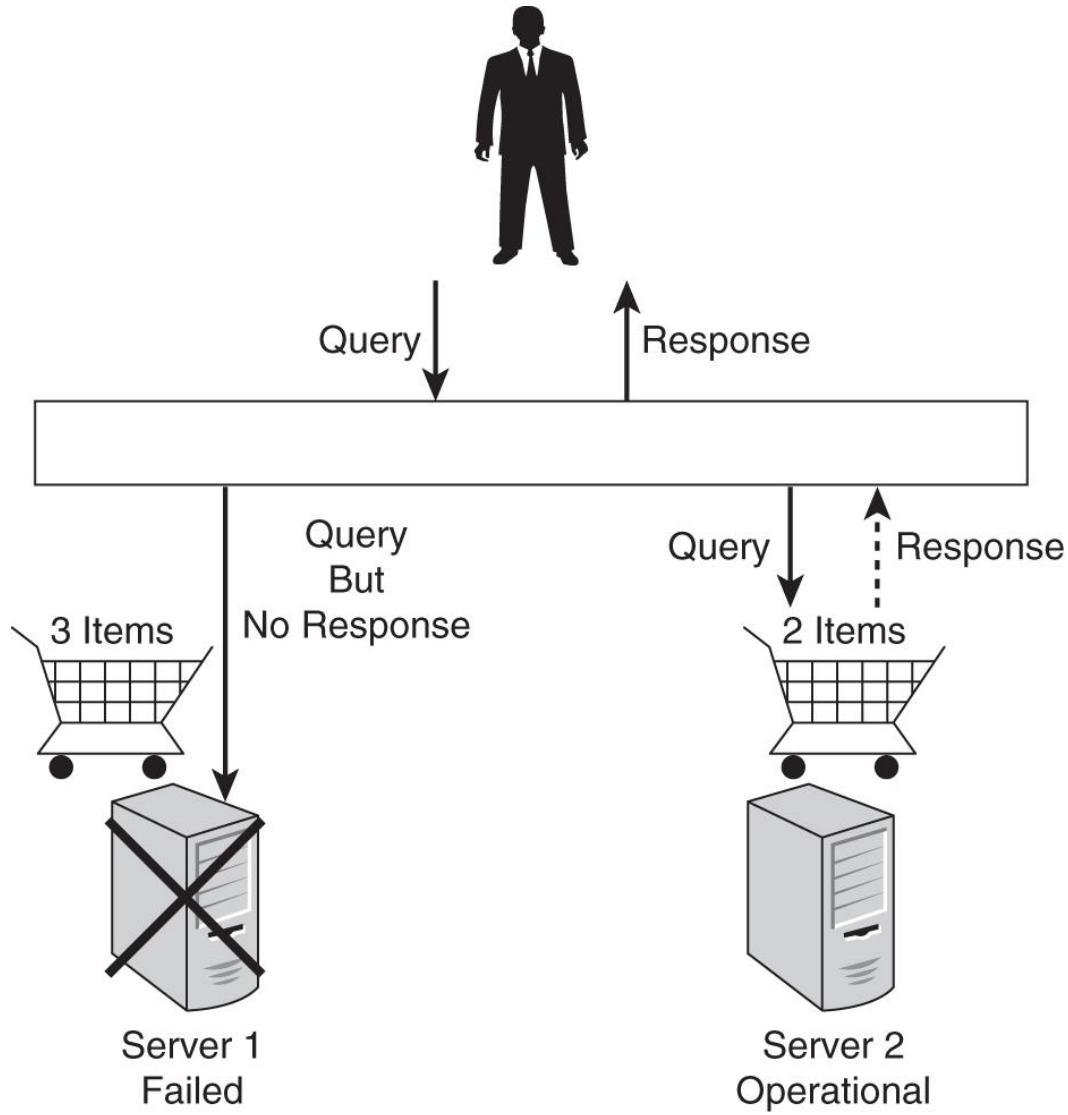

Figure 2.8 Data can be available but not consistent.

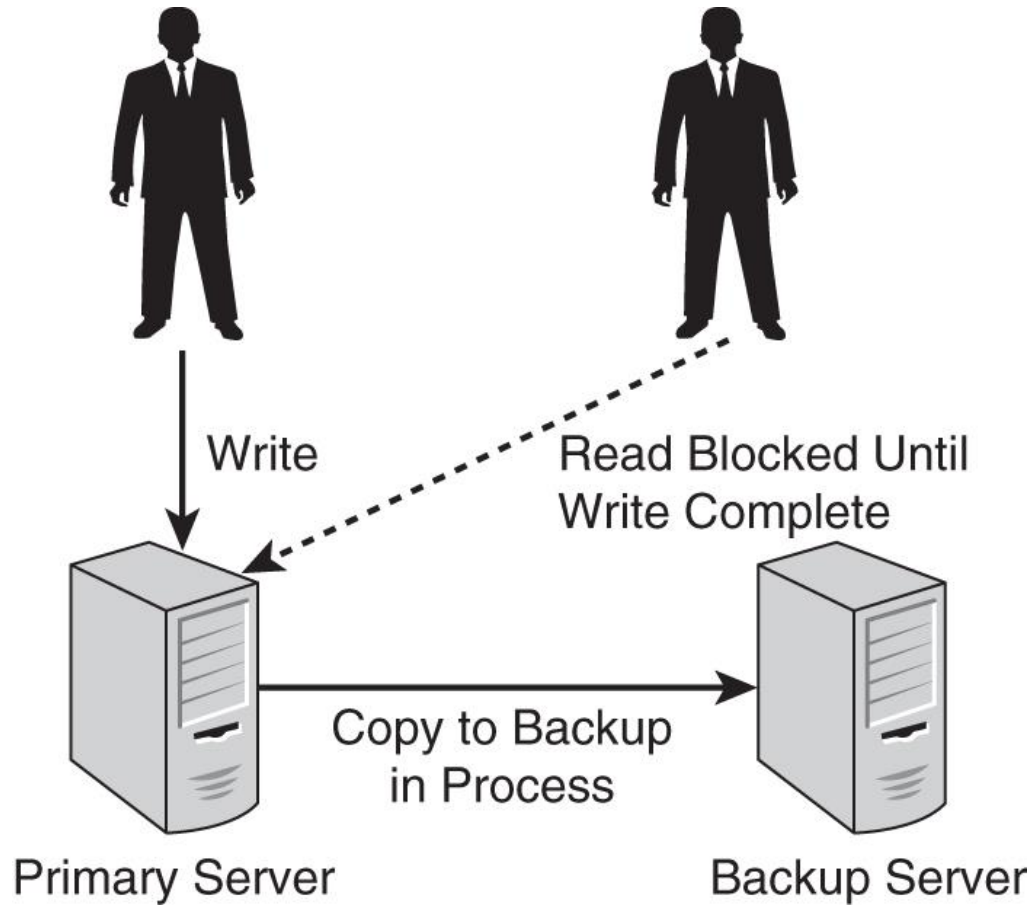

Figure 2.9 Data can be consistent but not available.

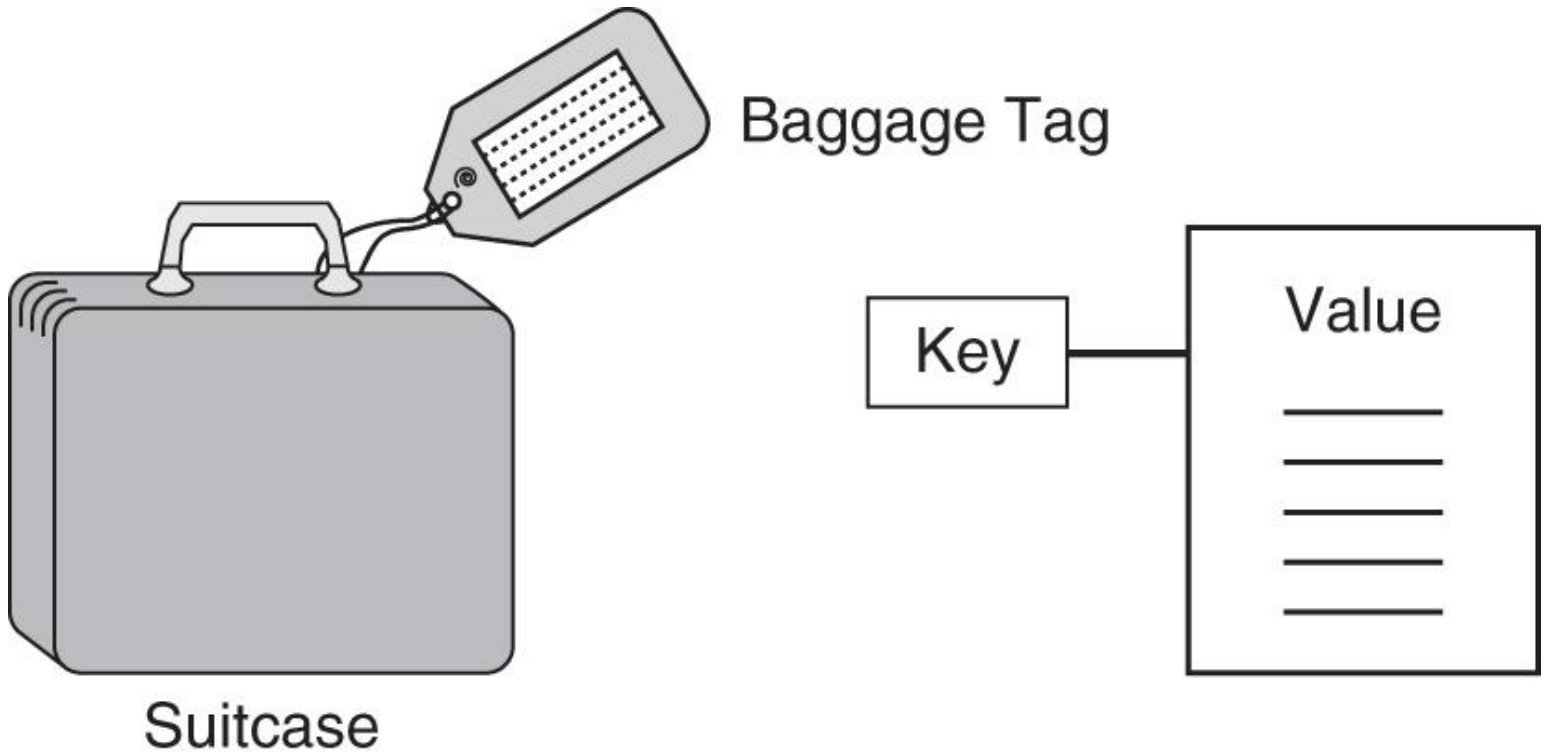

Figure 2.10 Airline tags for checked bags are analogous to keys used to store data in a key-value database.

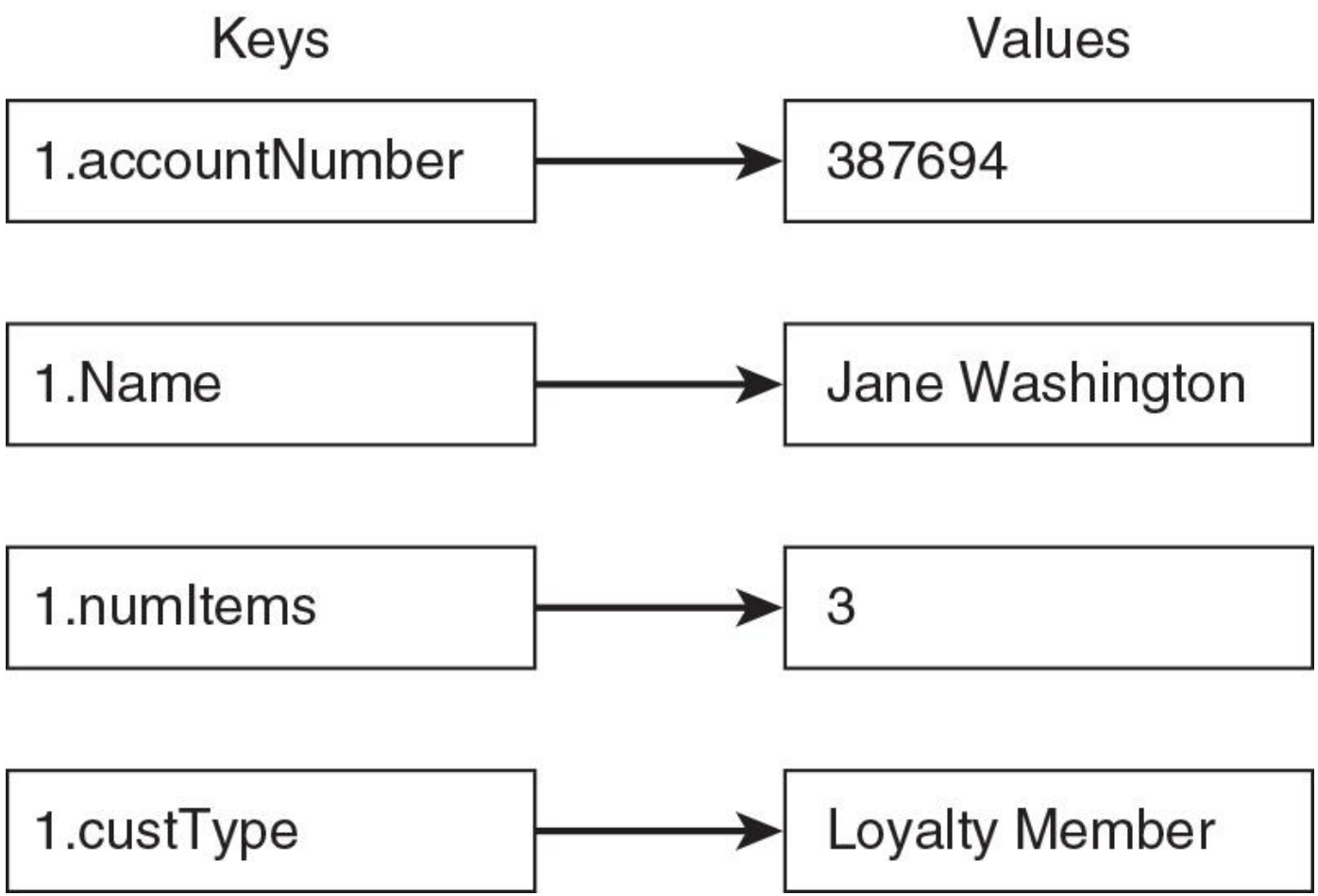

Figure 2.11 Key-value databases are modeled on a simple, two-part data structure consisting of an identifier and a data value.

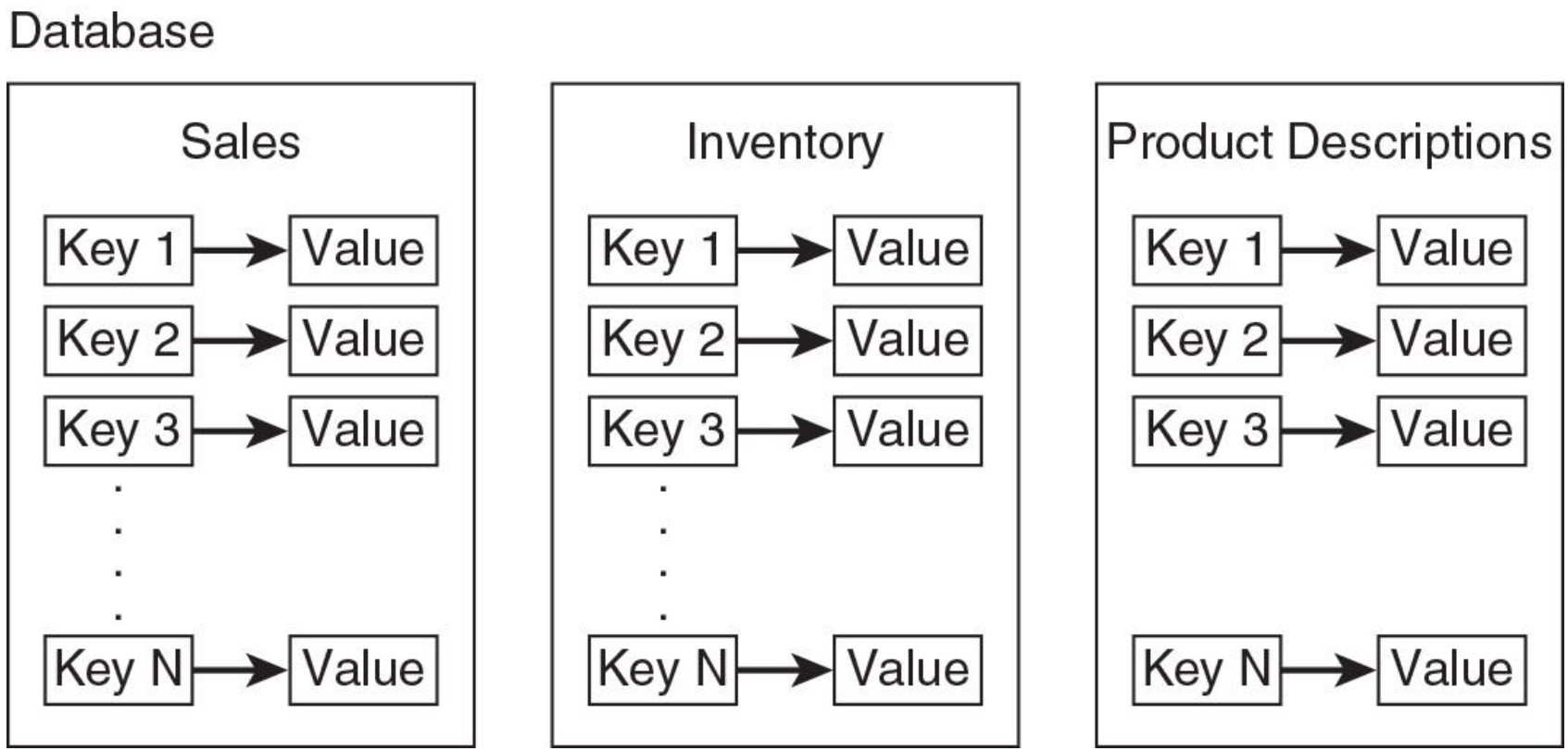

Figure 2.12 Key-value databases may support separate namespaces within a single database.

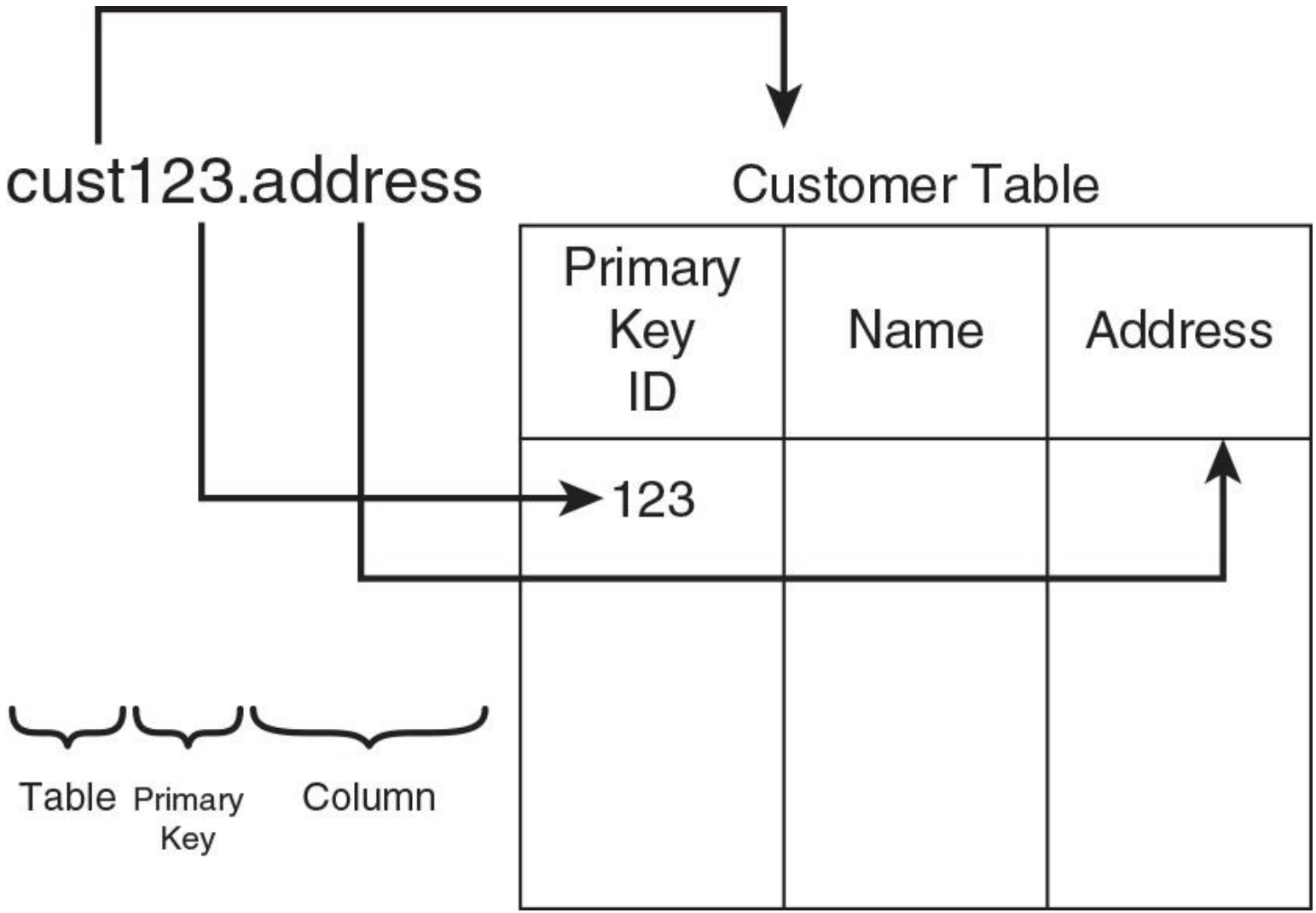

Figure 2.13 The key-naming convention outlined above maps to patterns seen in relational database tables.

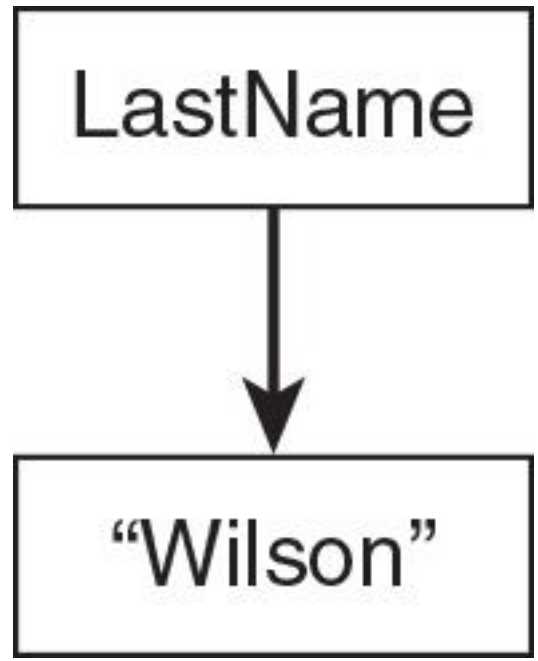

Figure 2.14 A column consists of a name and a value. In this example, the column is named lastName and has a value of “Wilson.”

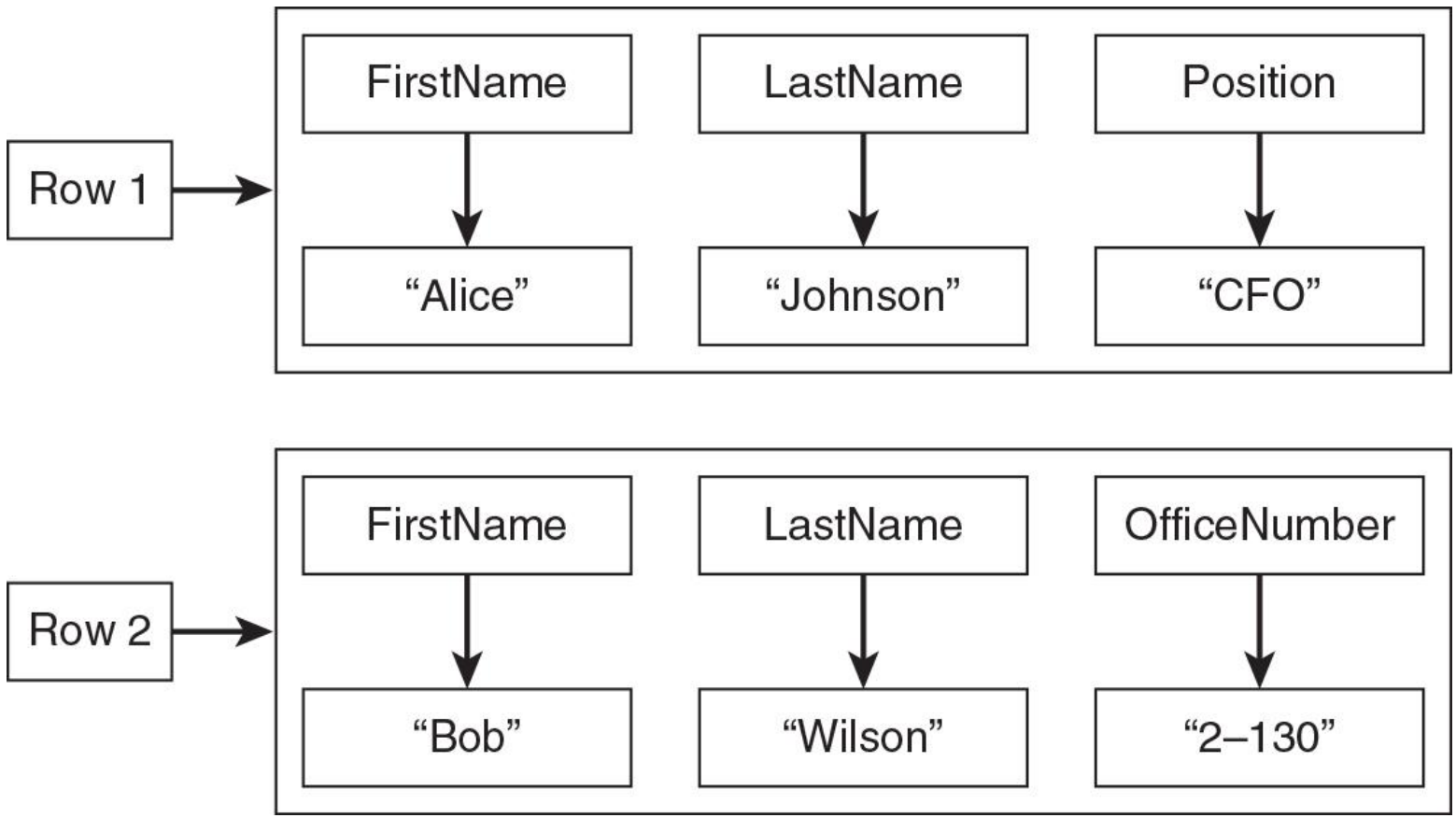

Figure 2.15 A row consists of one or more columns. Different rows can have different columns.

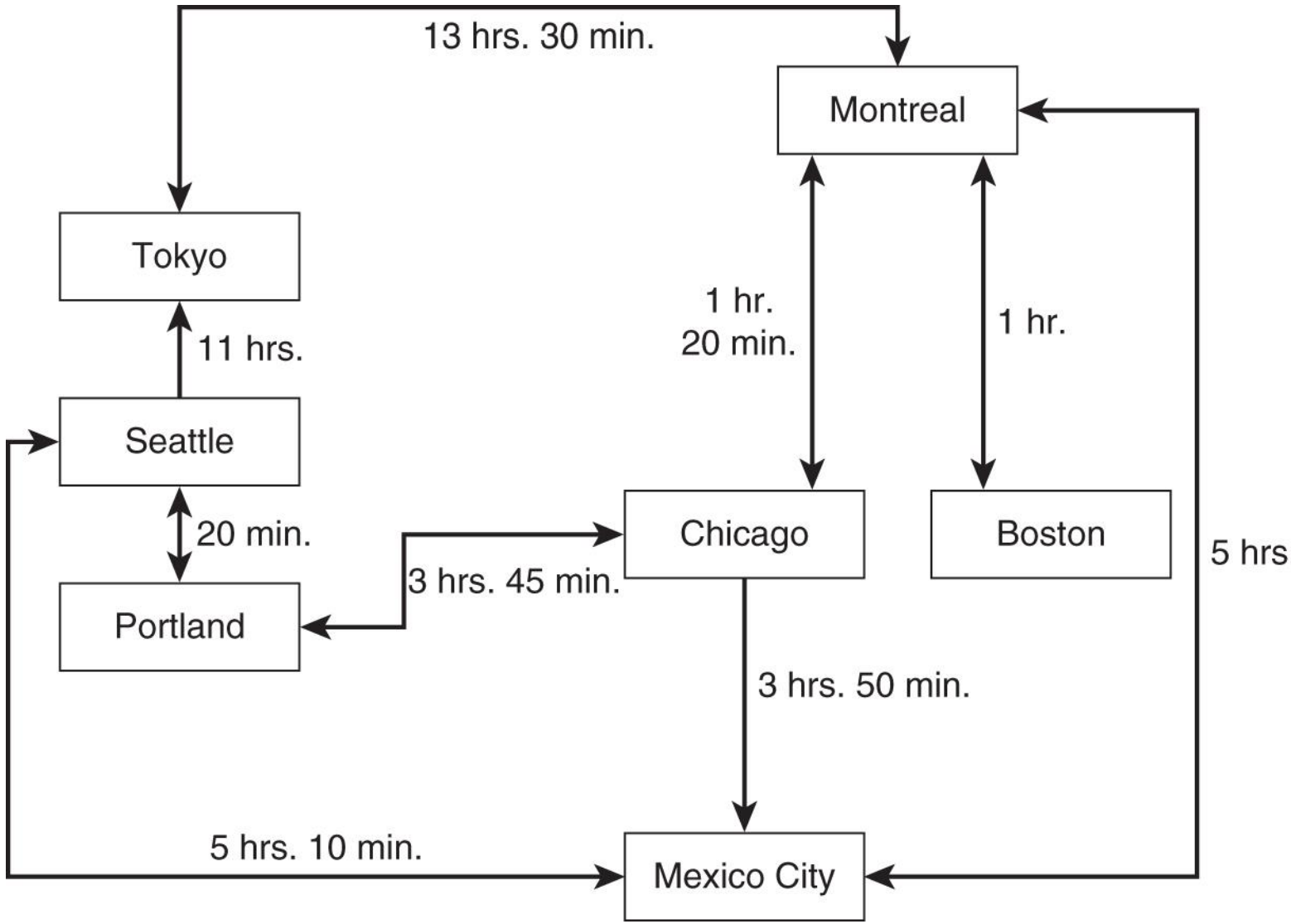

Figure 2.16 Properties of relationships or nodes store attributes about relations between linked nodes. In this case, attributes include flying times between cities.

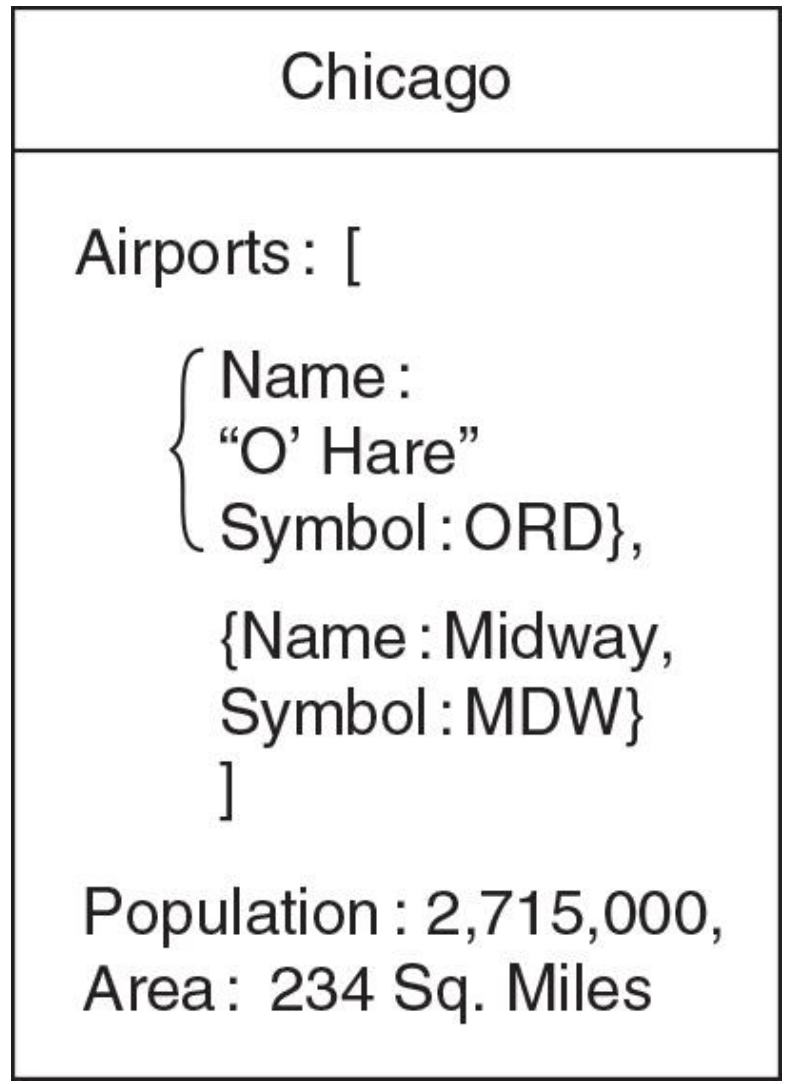

Figure 2.17 Nodes can also have attributes to describe the node. In this case, attributes include information about the airports in the city along with population and geographic area.

1	True
2	True
3	False
4	True
5	False
6	False
7	False
8	True
9	False
10	True

Figure 3.1 An array is an ordered list of elements. All elements are of the same type. The value of each element of the array is read and set by reference to its index.

‘Pi’	3.14
‘CapitalFrance’	‘Paris’
17234	34468
‘Foo’	‘Bar’
‘Start_Value’	1

Figure 3.2 An associative array shares some characteristics of arrays but has fewer constraints on keys and values.

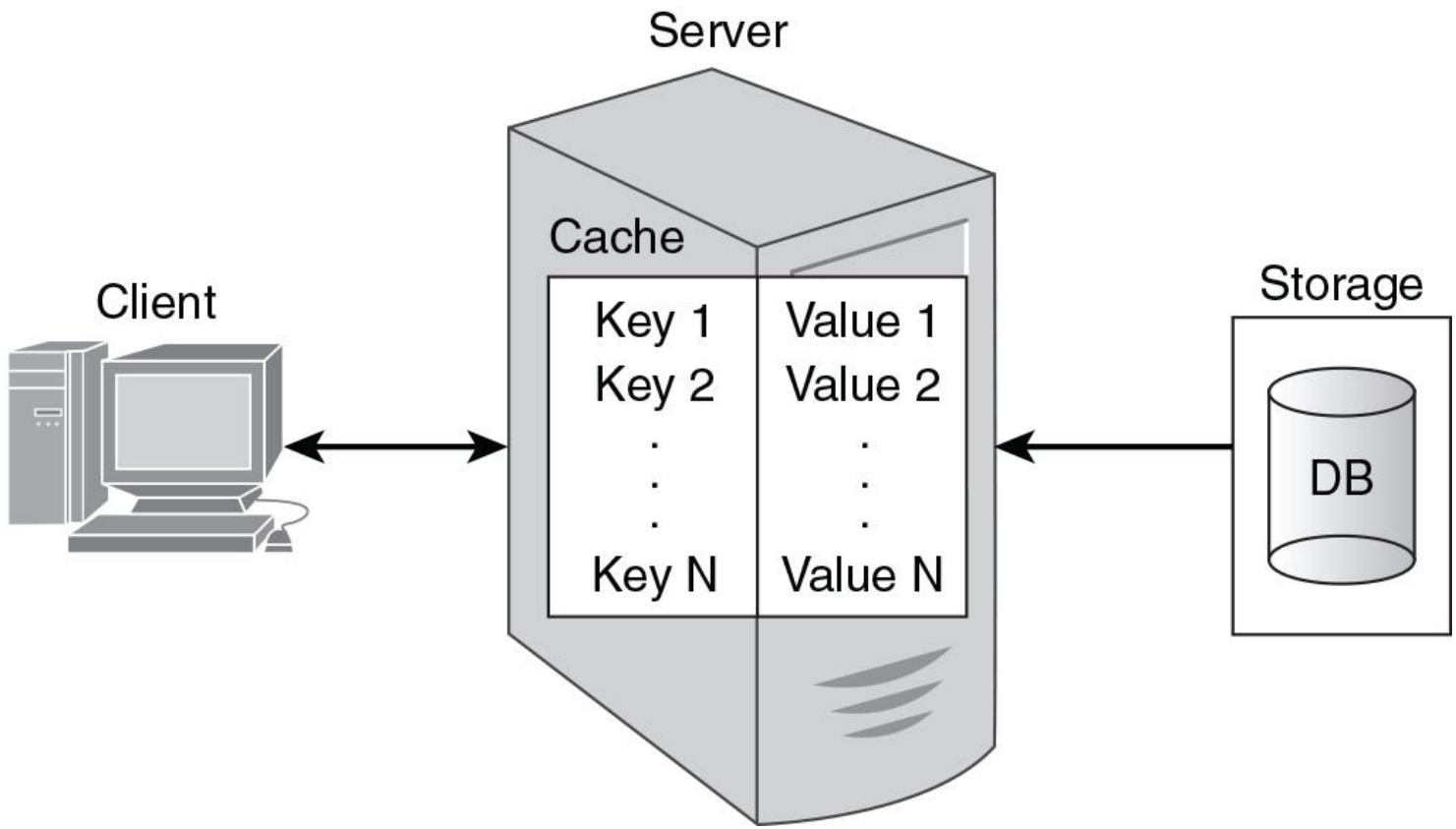

Figure 3.3 Caches are associative arrays used by application programs to improve data access performance.

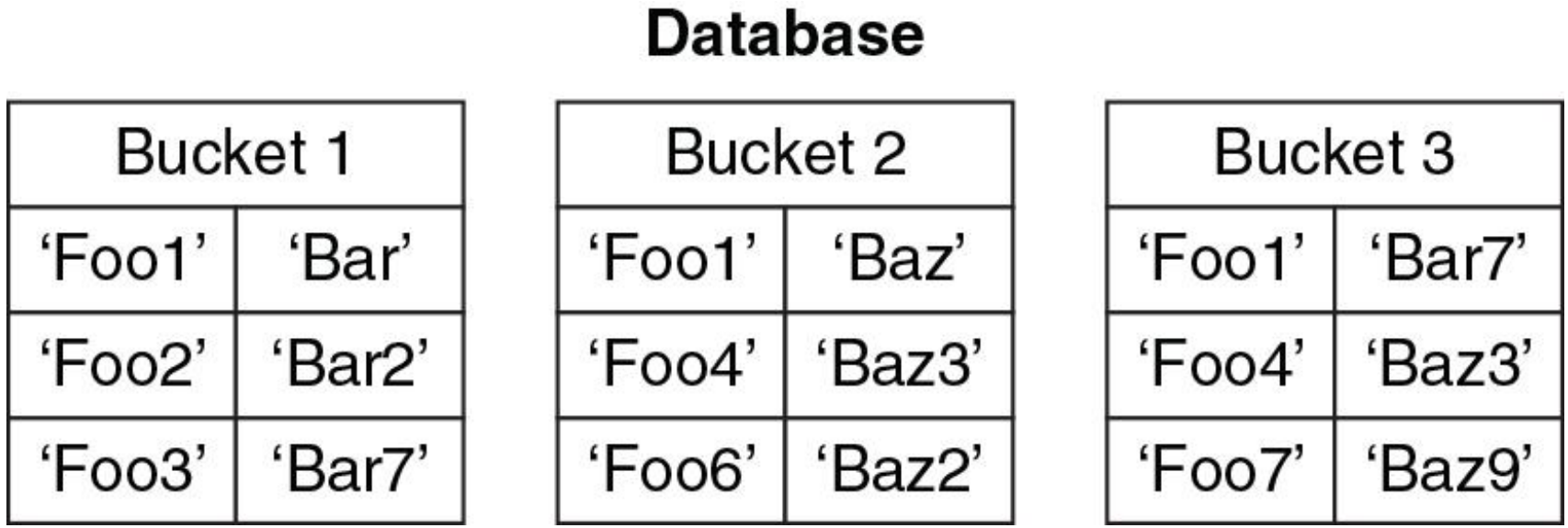

Figure 3.4 Keys of a key-value database must be unique within a namespace.

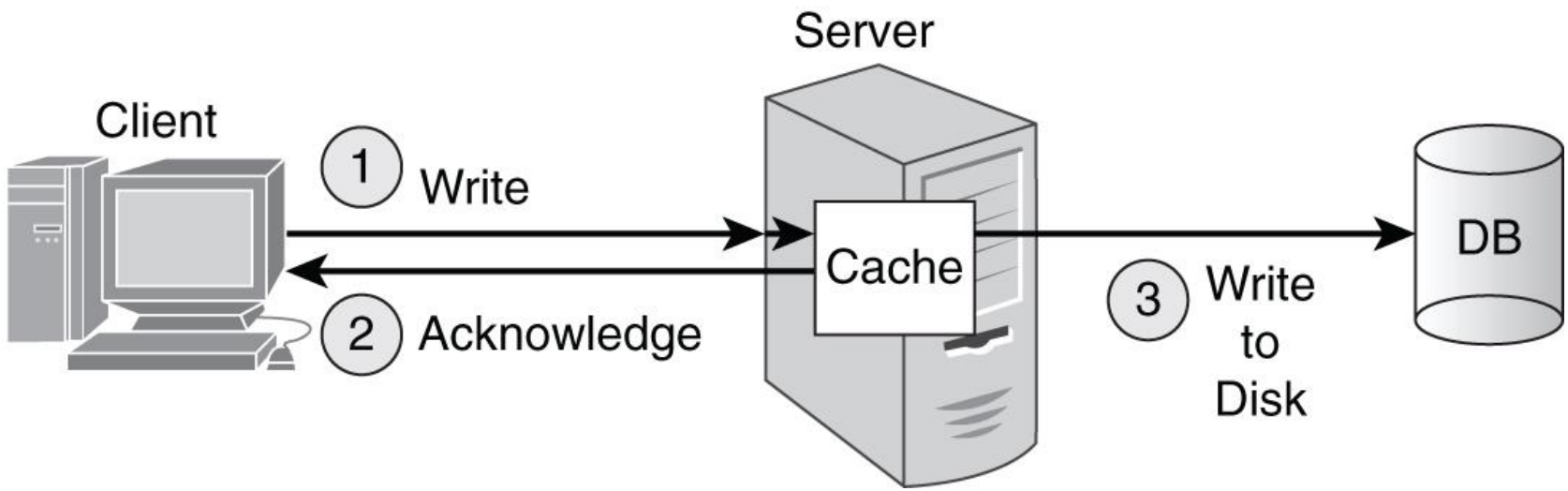

Figure 3.5 Write operations can return control to the calling application faster by first writing inserts and updates to RAM and then updating disk storage.

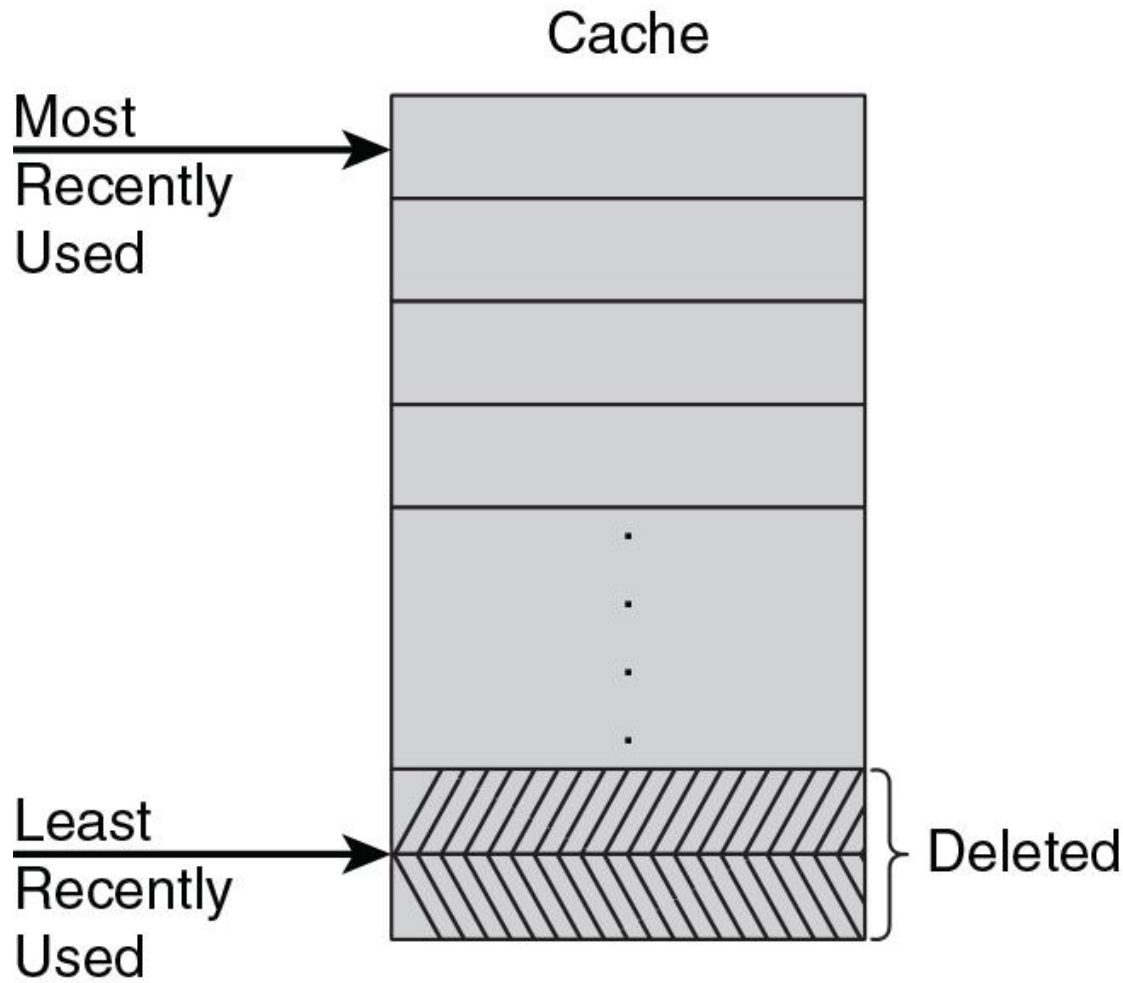

Figure 3.6 Least recently used algorithms delete data that has not been read or written as recently as other data.

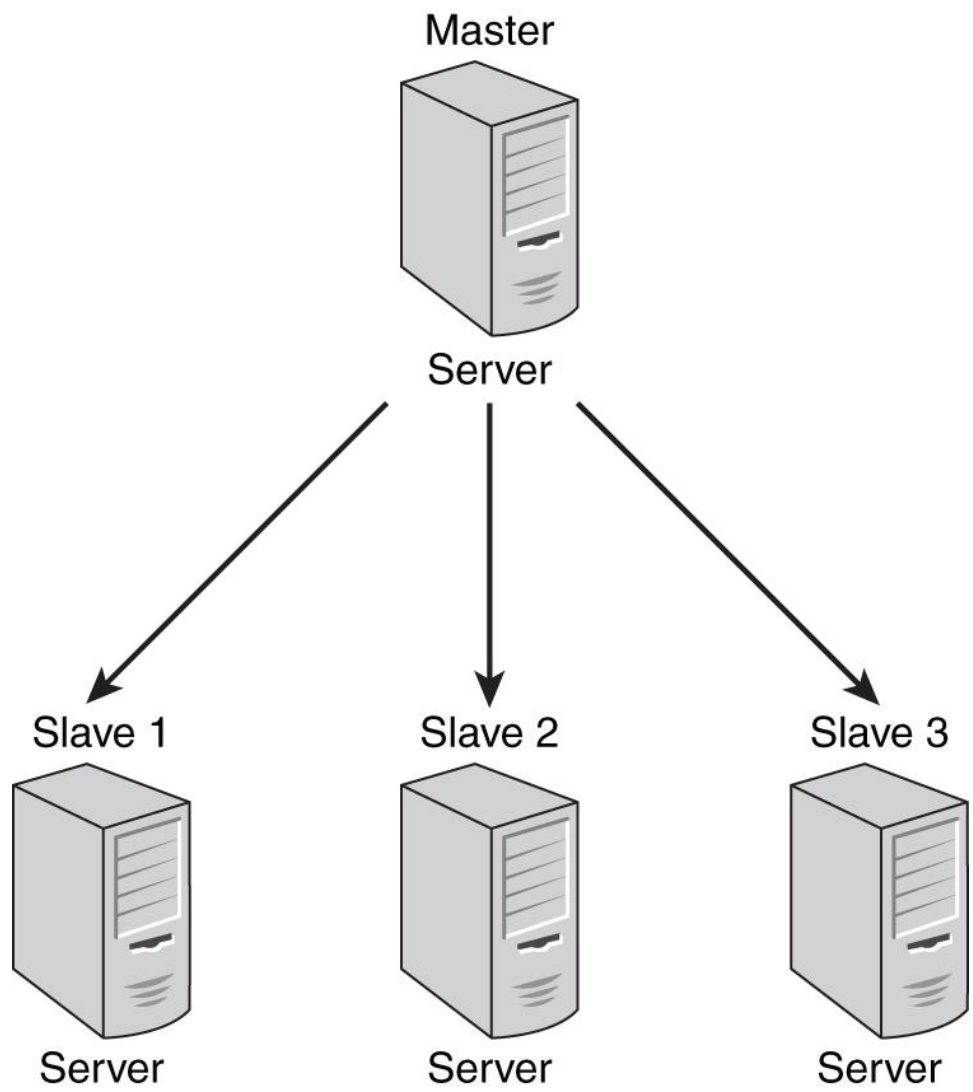

Figure 3.7 Master-slave architectures have a simple communication pattern during normal operations.

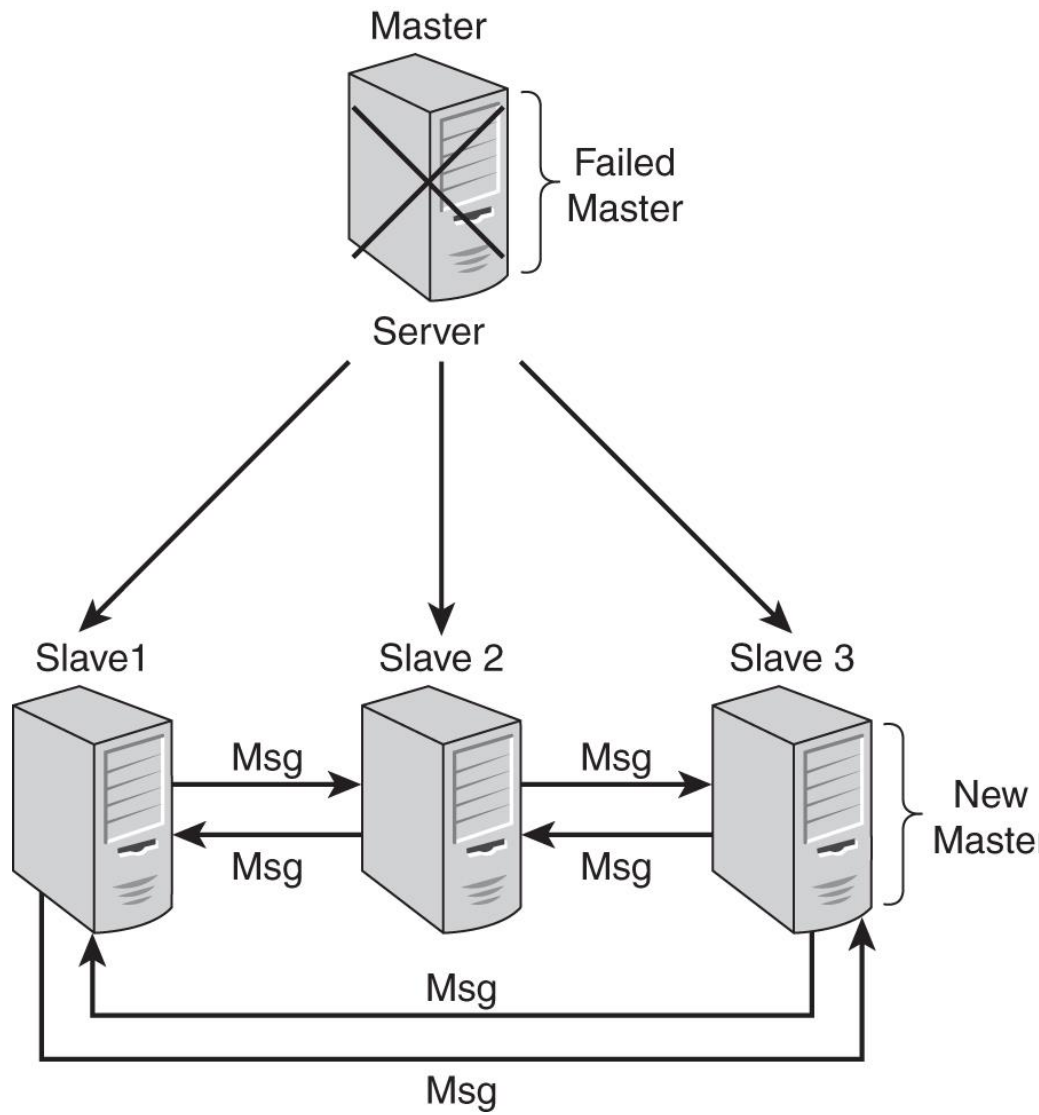

Figure 3.8 Once a failed master server is detected, the slaves initiate a protocol to elect a new master.

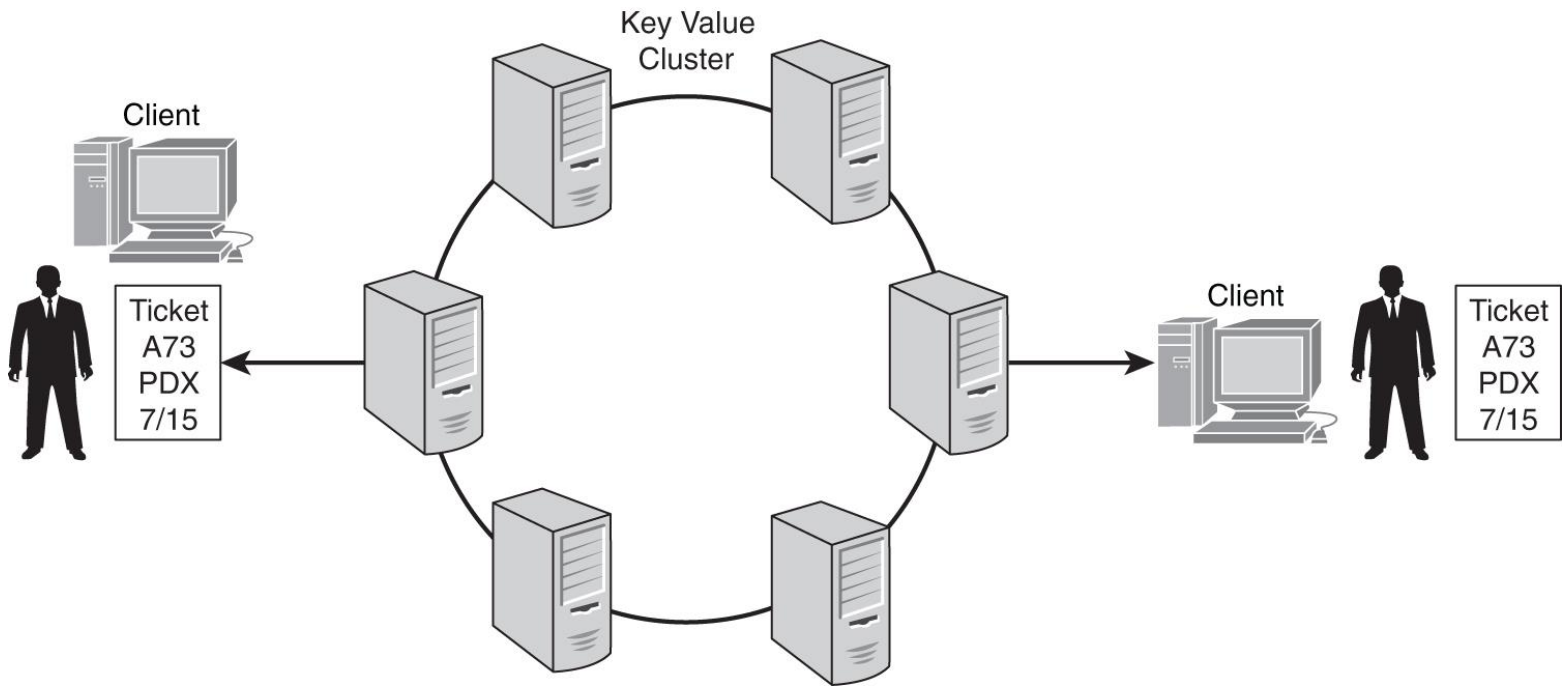

Figure 3.9 A fan's worst nightmare: Multiple fans are able to purchase tickets for the same seat.

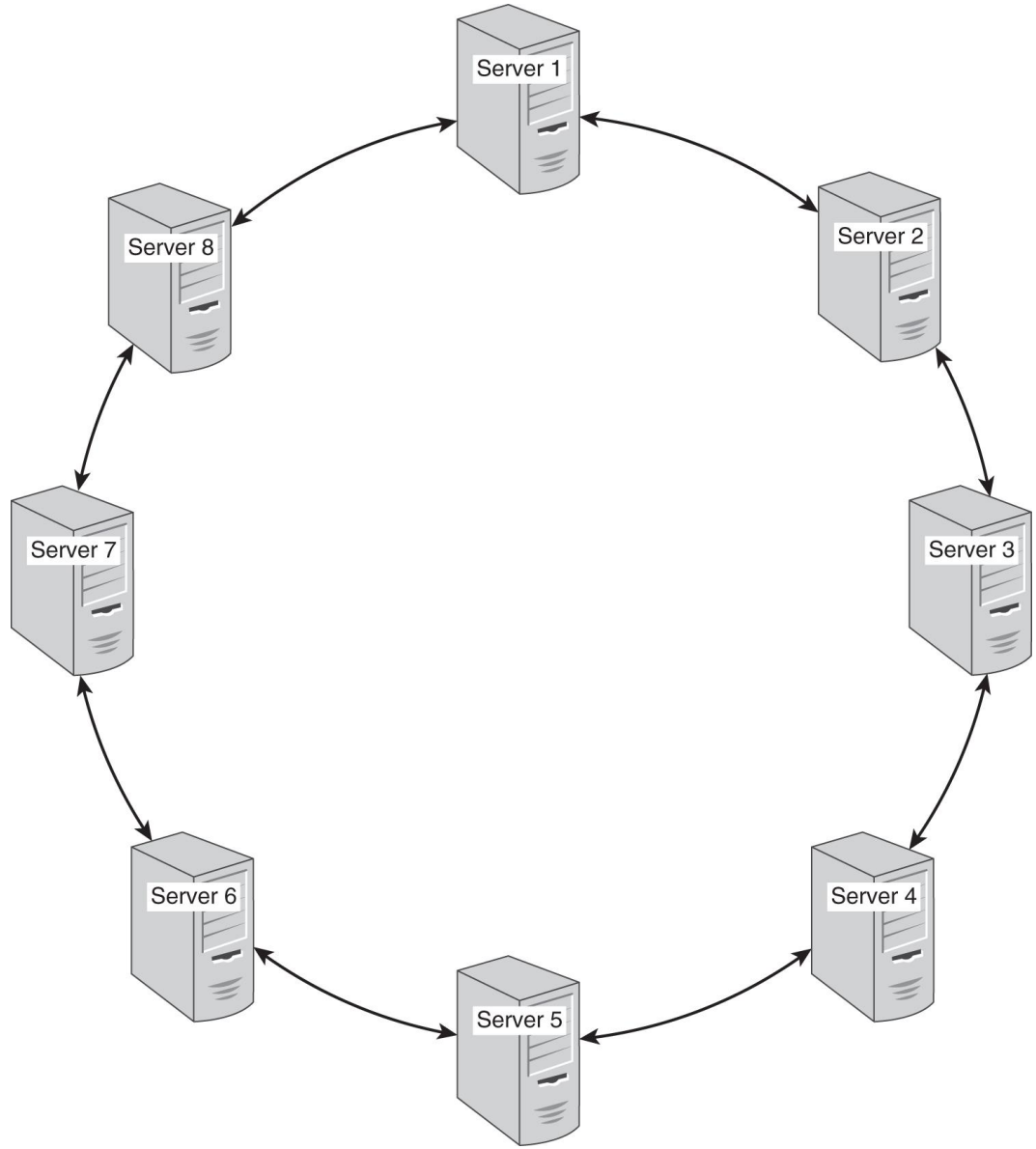

Figure 3.10 An eight-server cluster in a ring configuration.

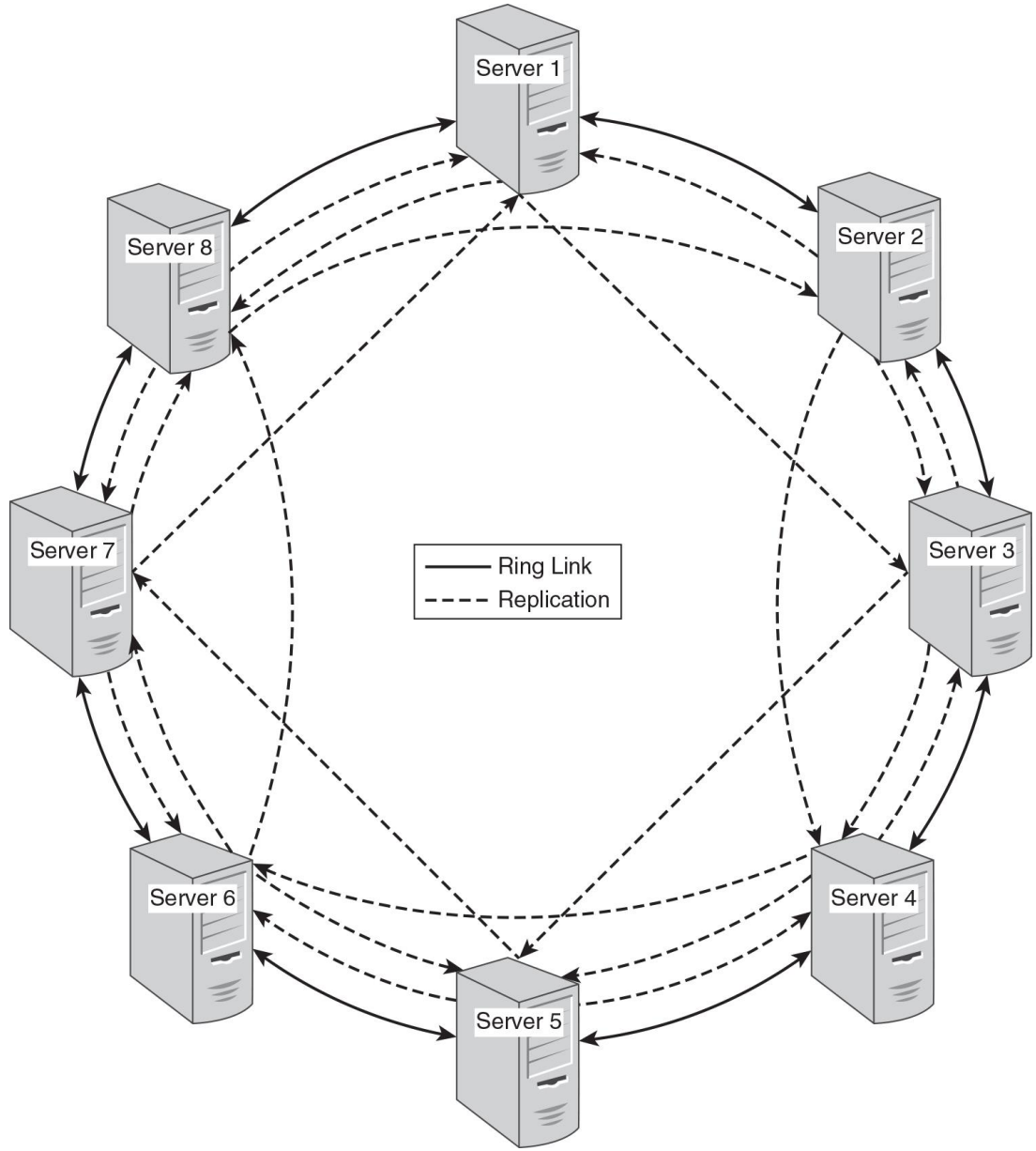

Figure 3.11 An eight-server cluster in a ring configuration with a replication factor of 4.

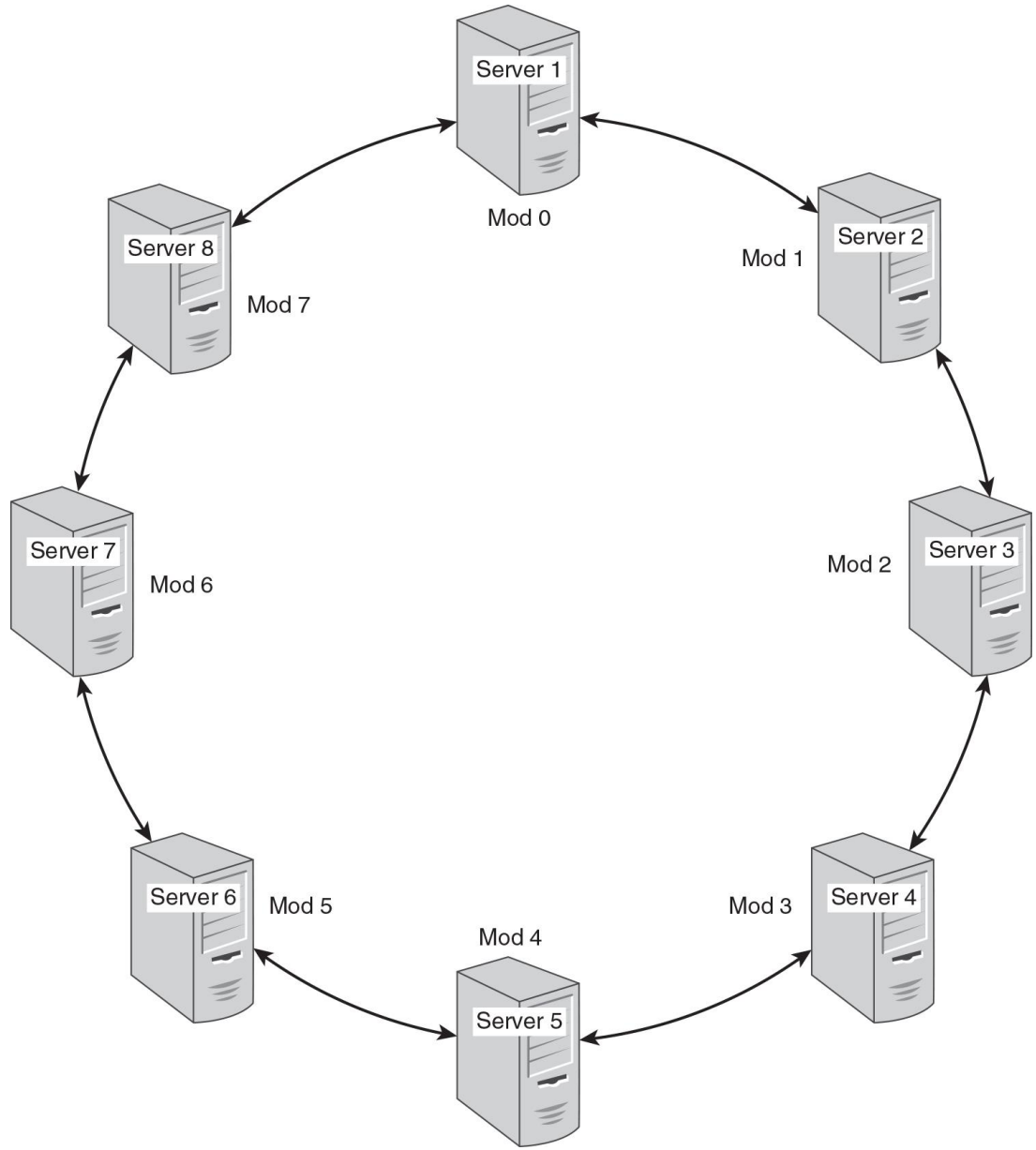

Figure 3.12 An eight-server cluster in a ring configuration with modulo number assigned.

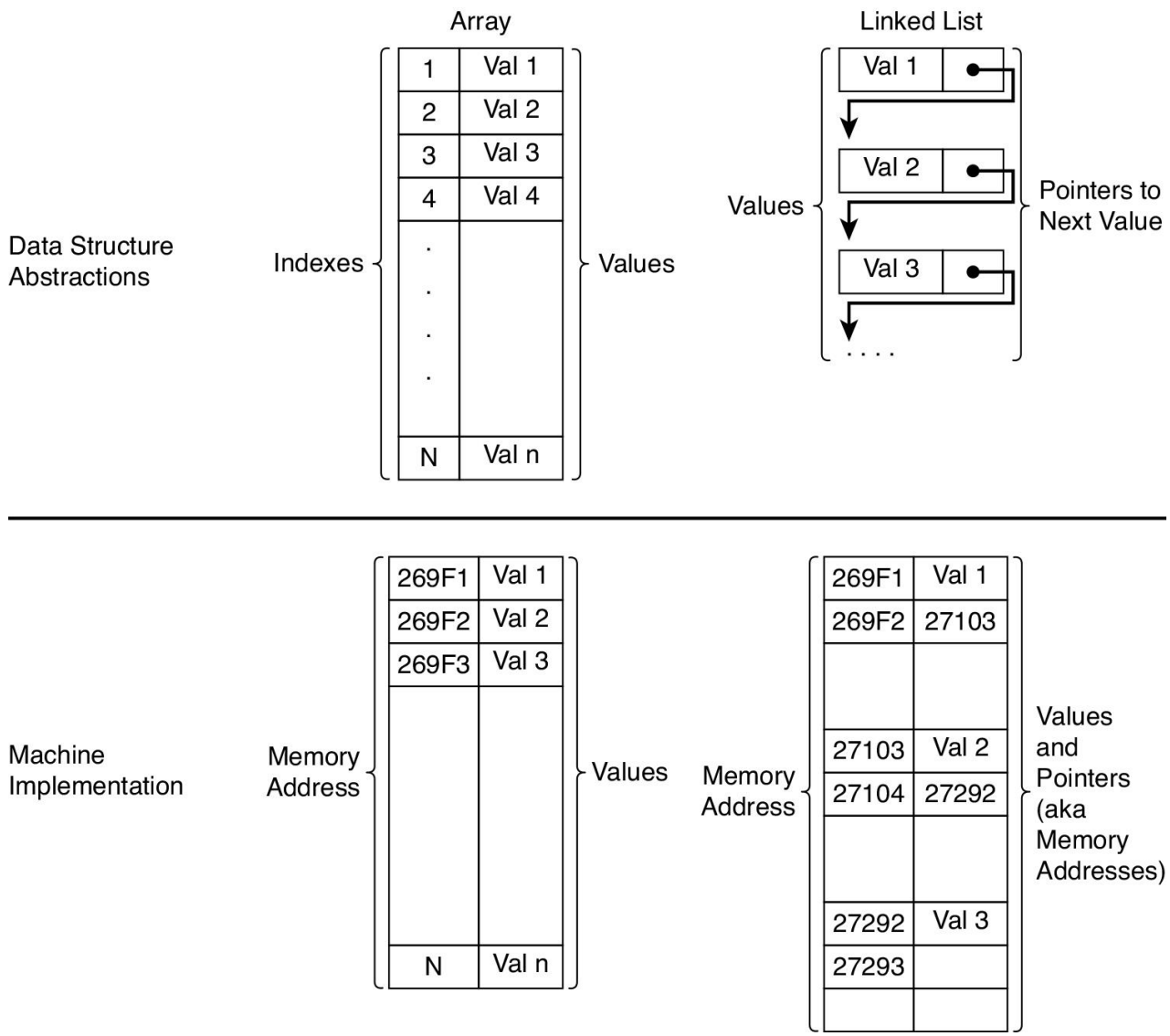

Figure 4.1 Data structures provide higher-level organizations than available at the machine level.

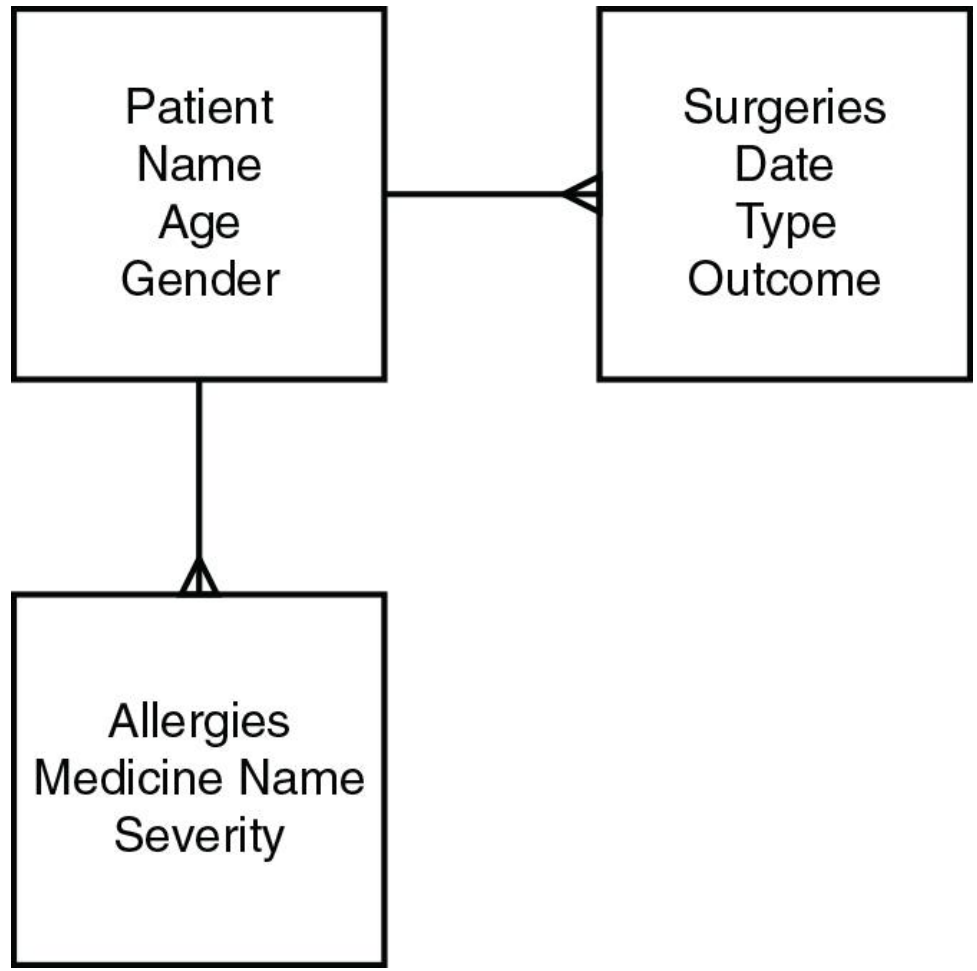

Figure 4.2 Data models provide a layer of abstraction above data structures that allows database application developers to focus more on the information that must be managed and less on implementation issues.

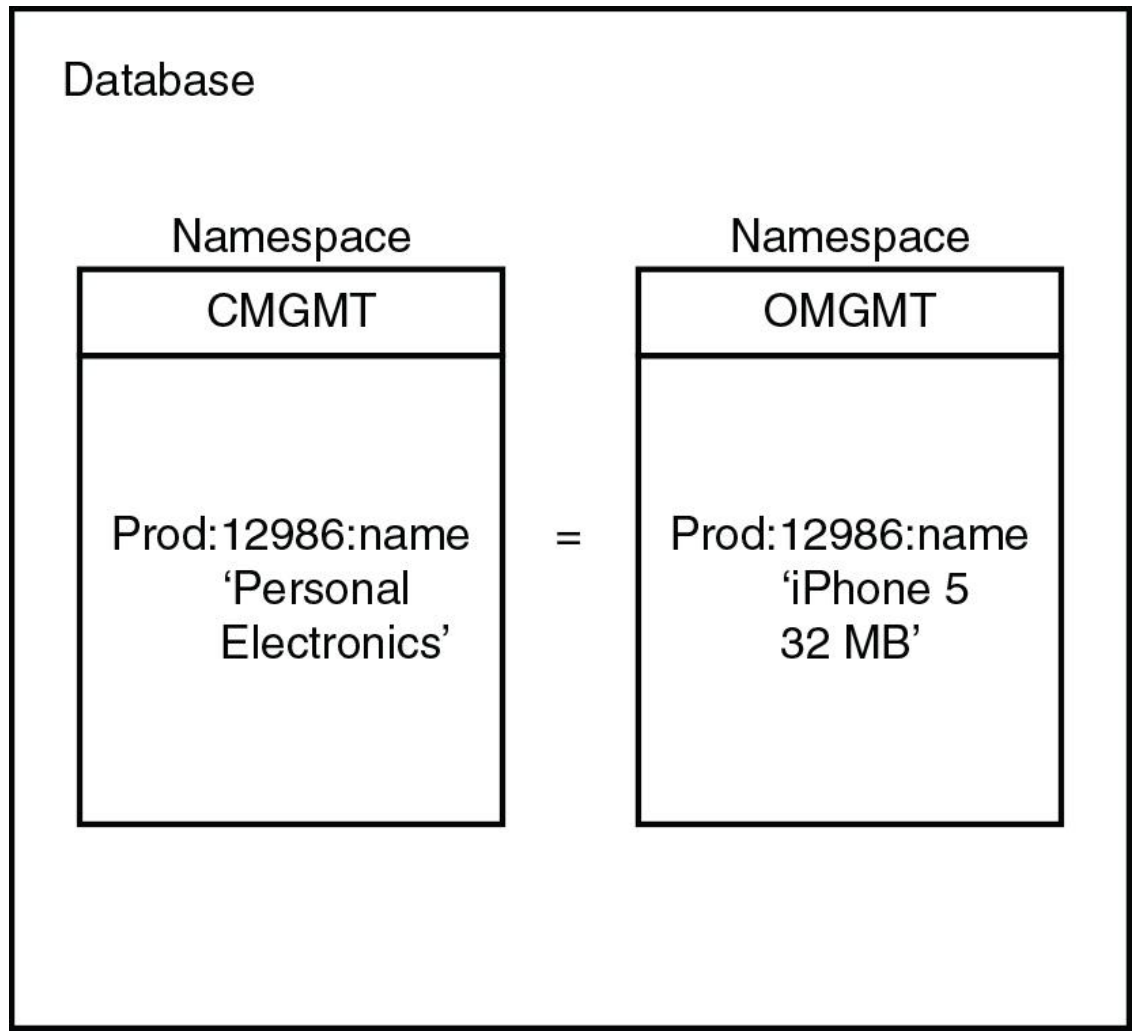

Figure 4.3 Namespaces enable duplicate keys to exist without causing conflicts by maintaining separate collections of keys.

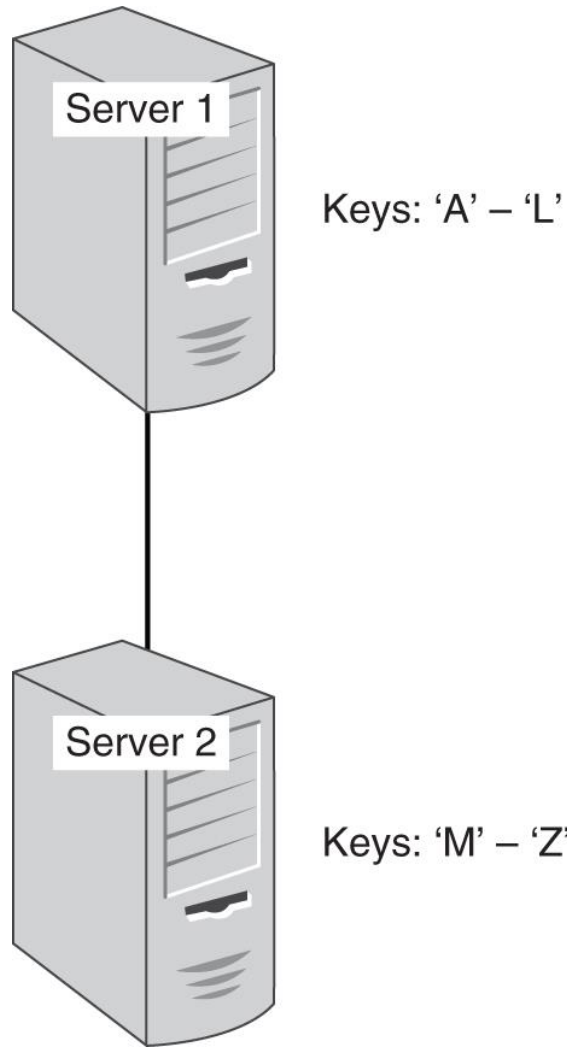

Figure 4.4 Servers in a cluster are assigned subsets of data to manage.

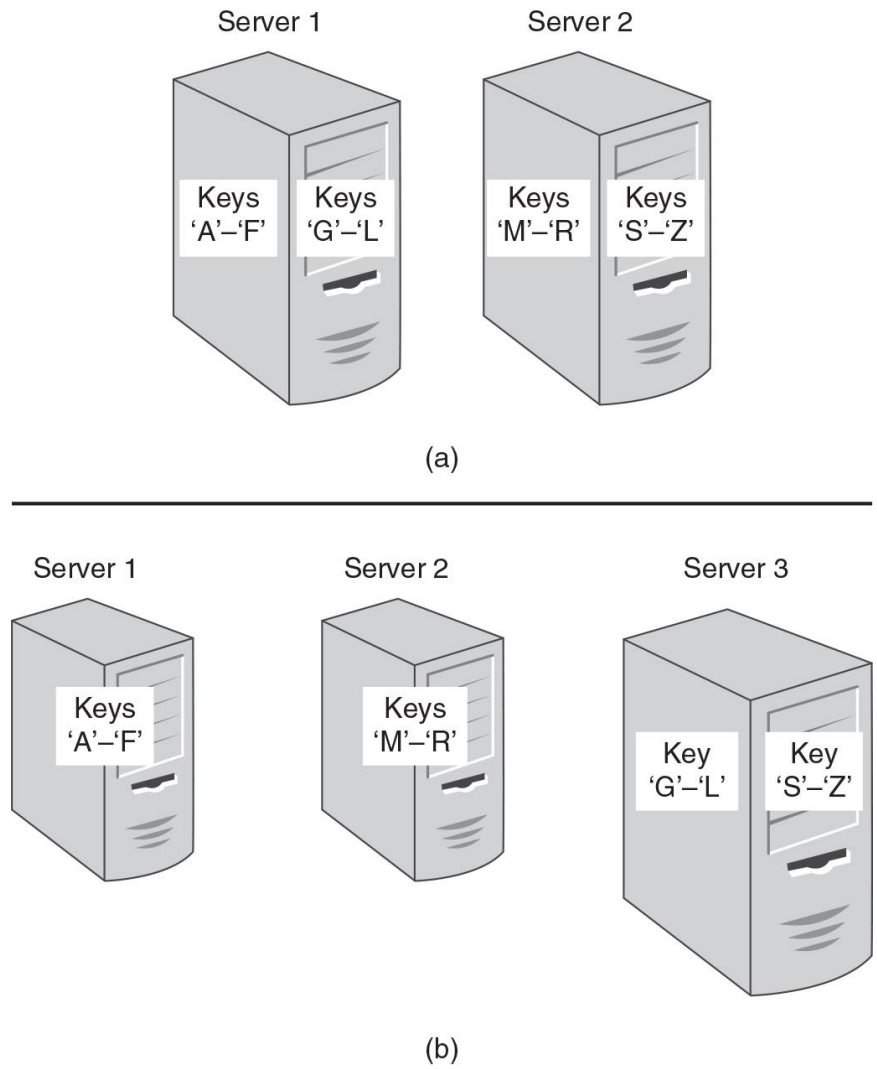

Figure 4.5 When multiple instances of key-value database software run on servers in a cluster, servers can be added to the cluster and instances reallocated to balance the workload.

Key-Value Database	
Keys	Values
cust:8983:firstName	‘Jane’
cust:8983:lastName	‘Anderson’
cust:8983:fullName	‘Jane Anderson’

Figure 4.6 Schemaless data models allow for multiple types of representations of the same data to exist simultaneously.

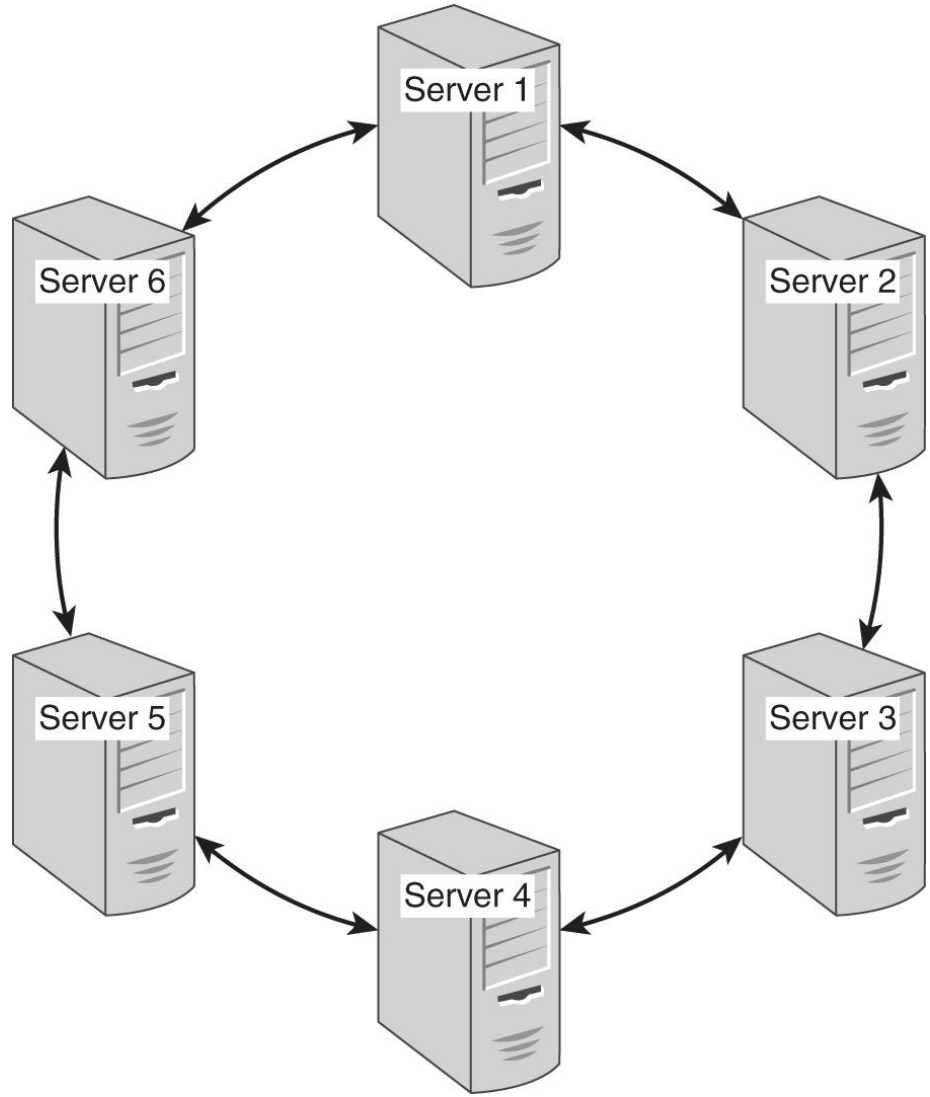

Figure 4.7 A ring architecture of key-value databases links adjacent nodes in the cluster.

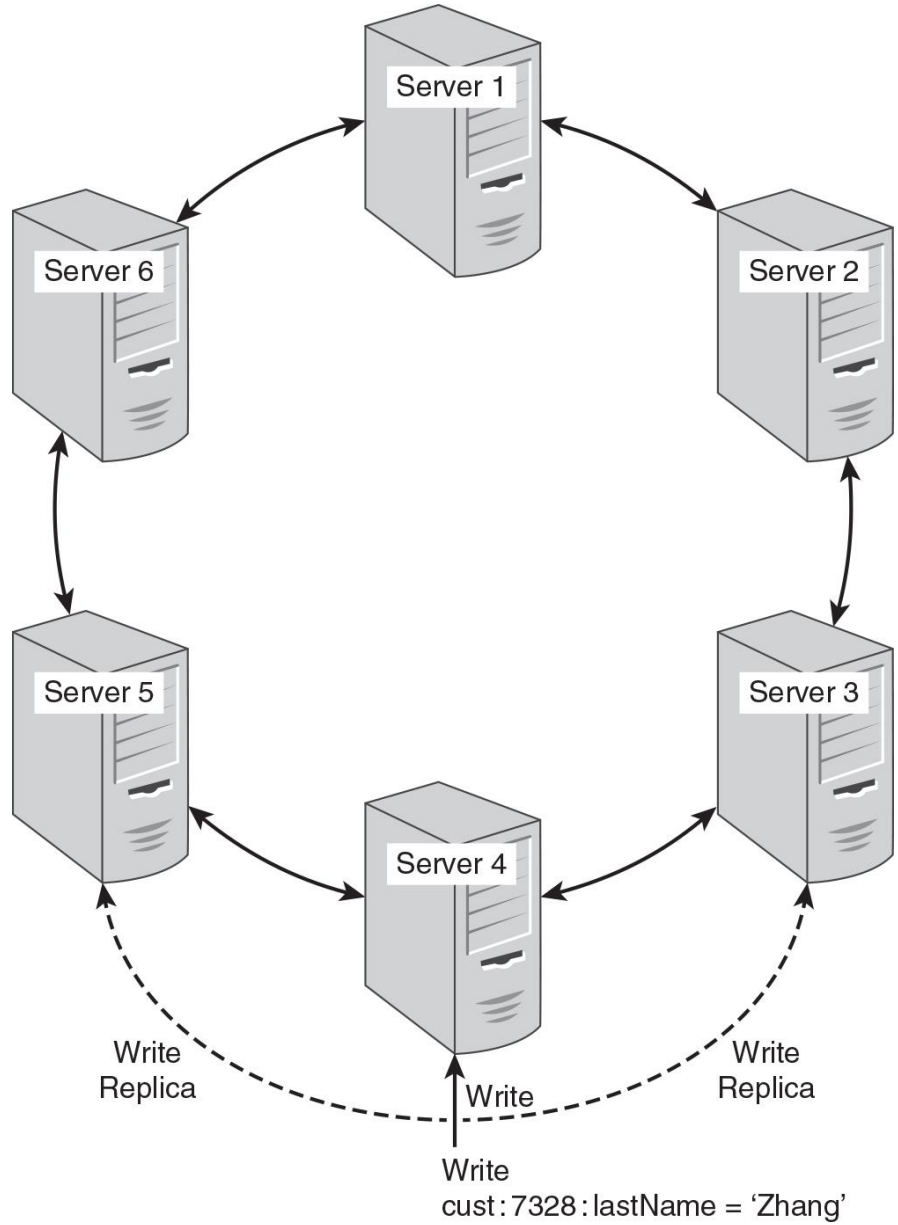

Figure 4.8 One way to replicate data is to write copies of data to adjacent nodes in the cluster ring.

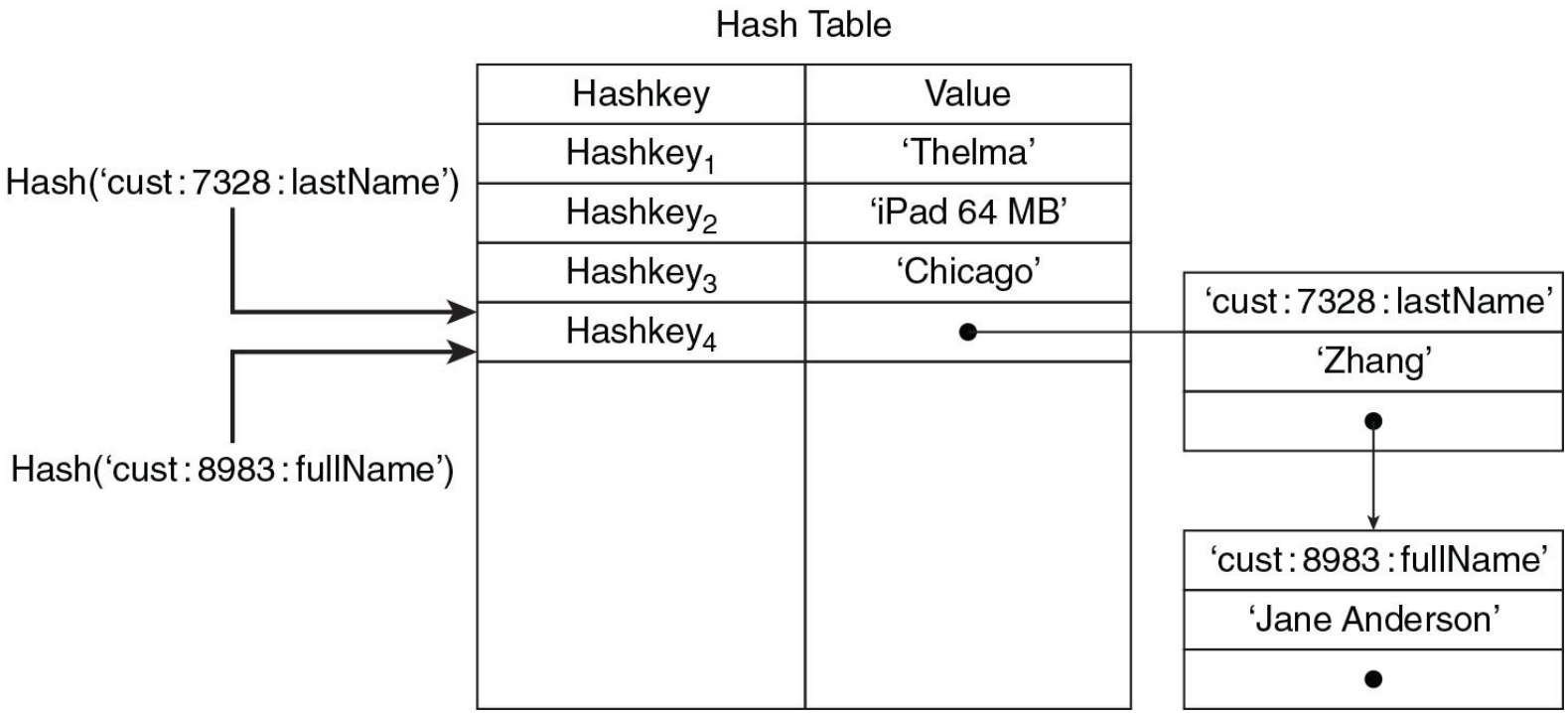

Figure 4.9 Collisions with hash functions are managed using collision resolution strategies, such as maintaining linked lists of values.

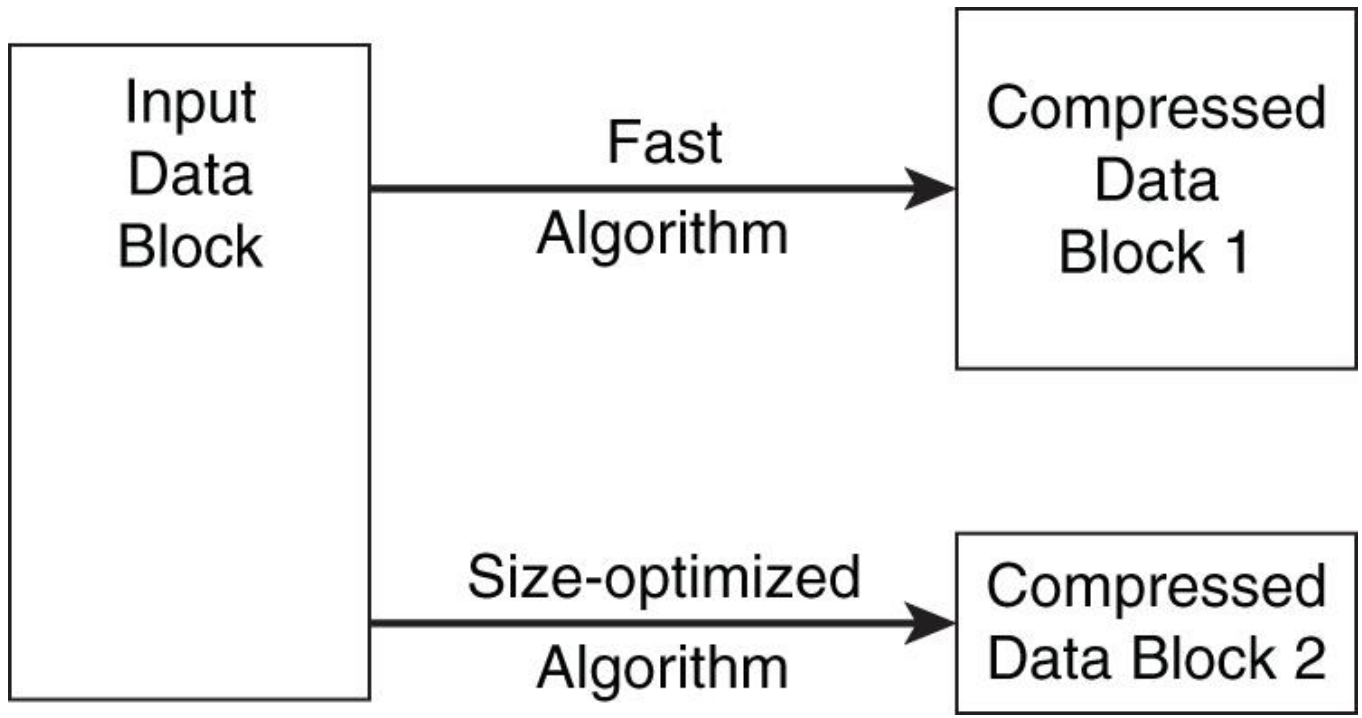

Figure 4.10 Compression algorithms may be designed to optimize for speed or data size.

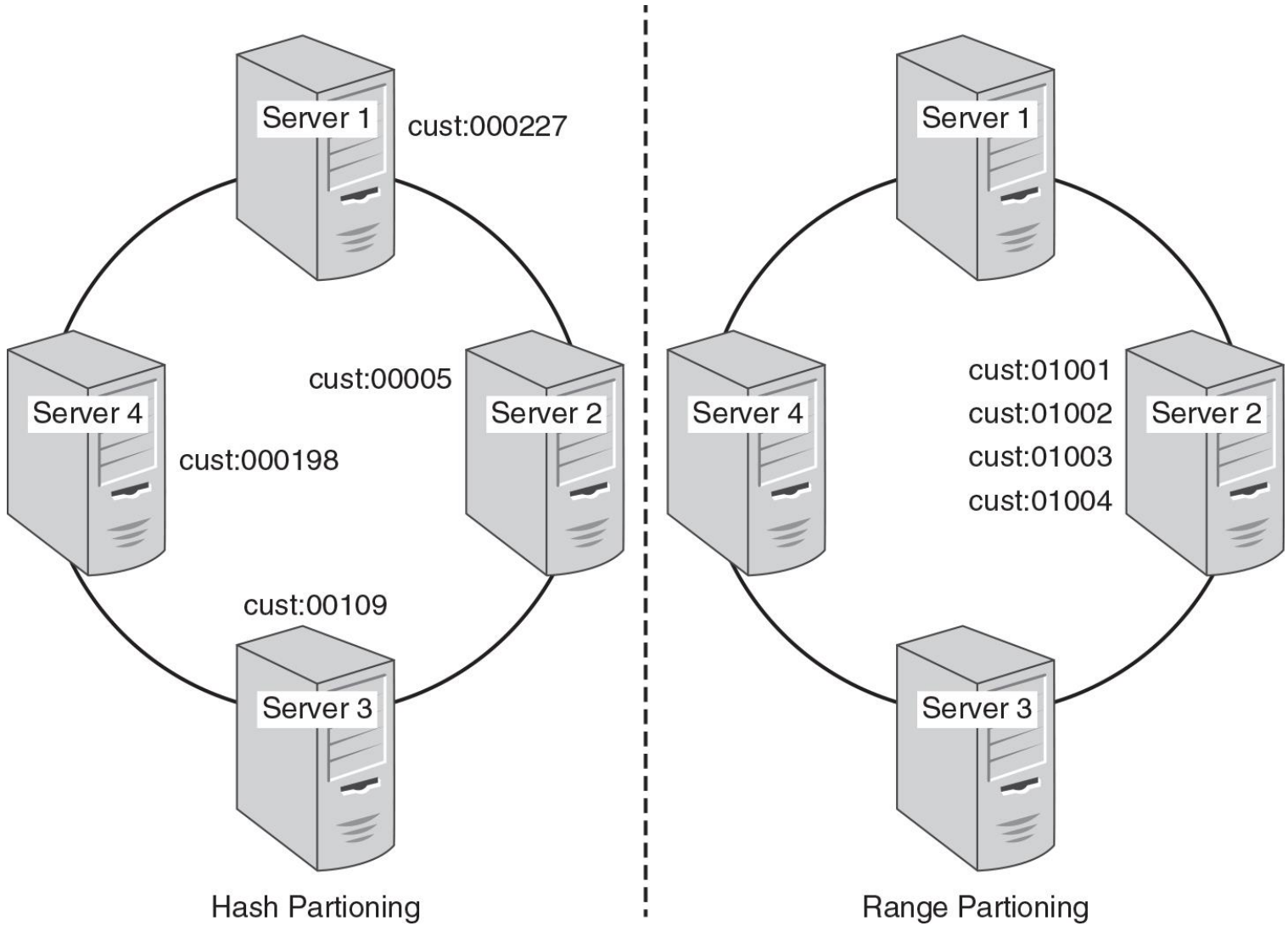

Figure 5.1 Different hashing schemes will lead to different key-to-node assignments.

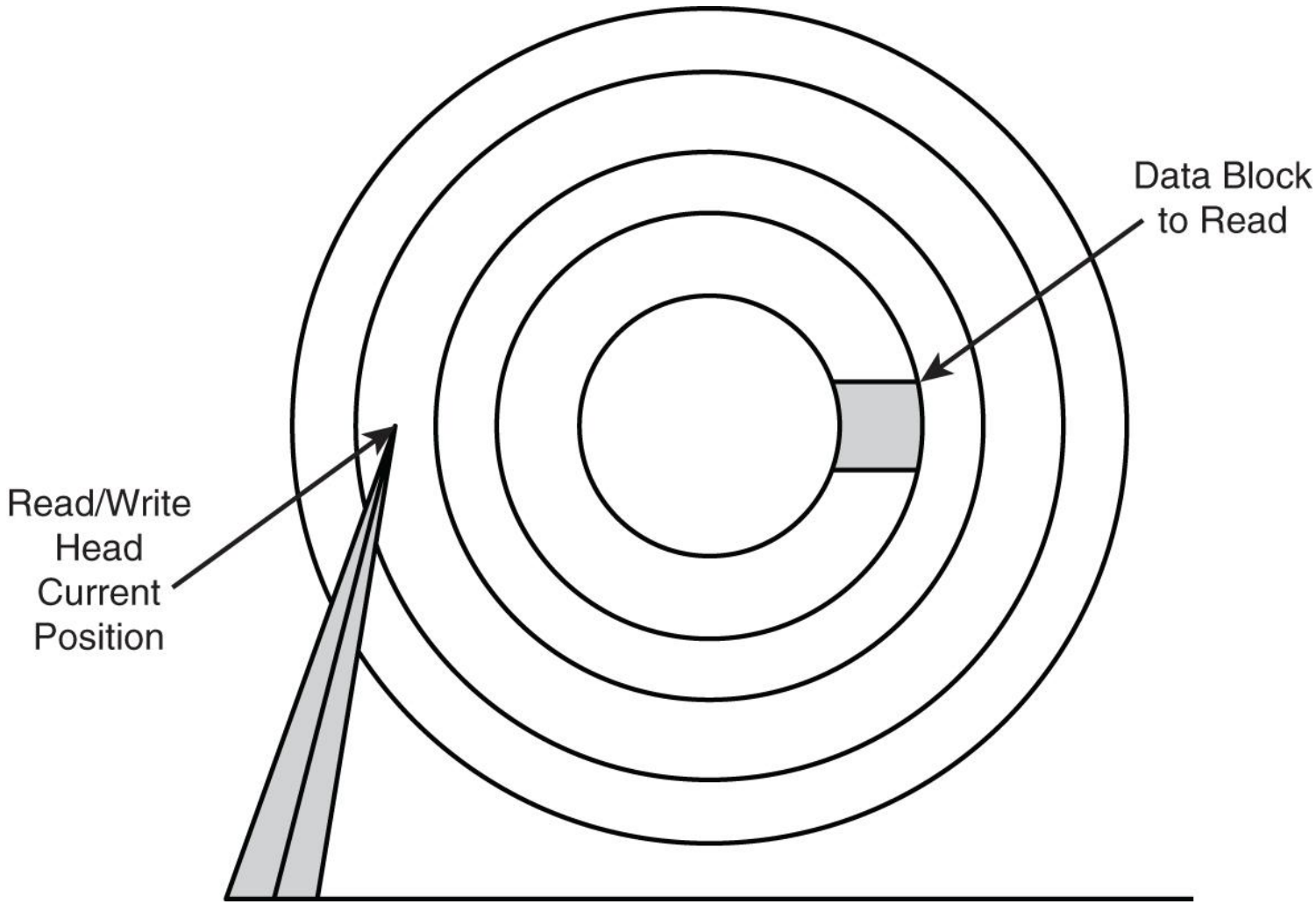

Figure 5.2 Reading a value from disk requires the read/write heads to move to the proper track and the platter to rotate to the proper block. This can lead to long latencies.

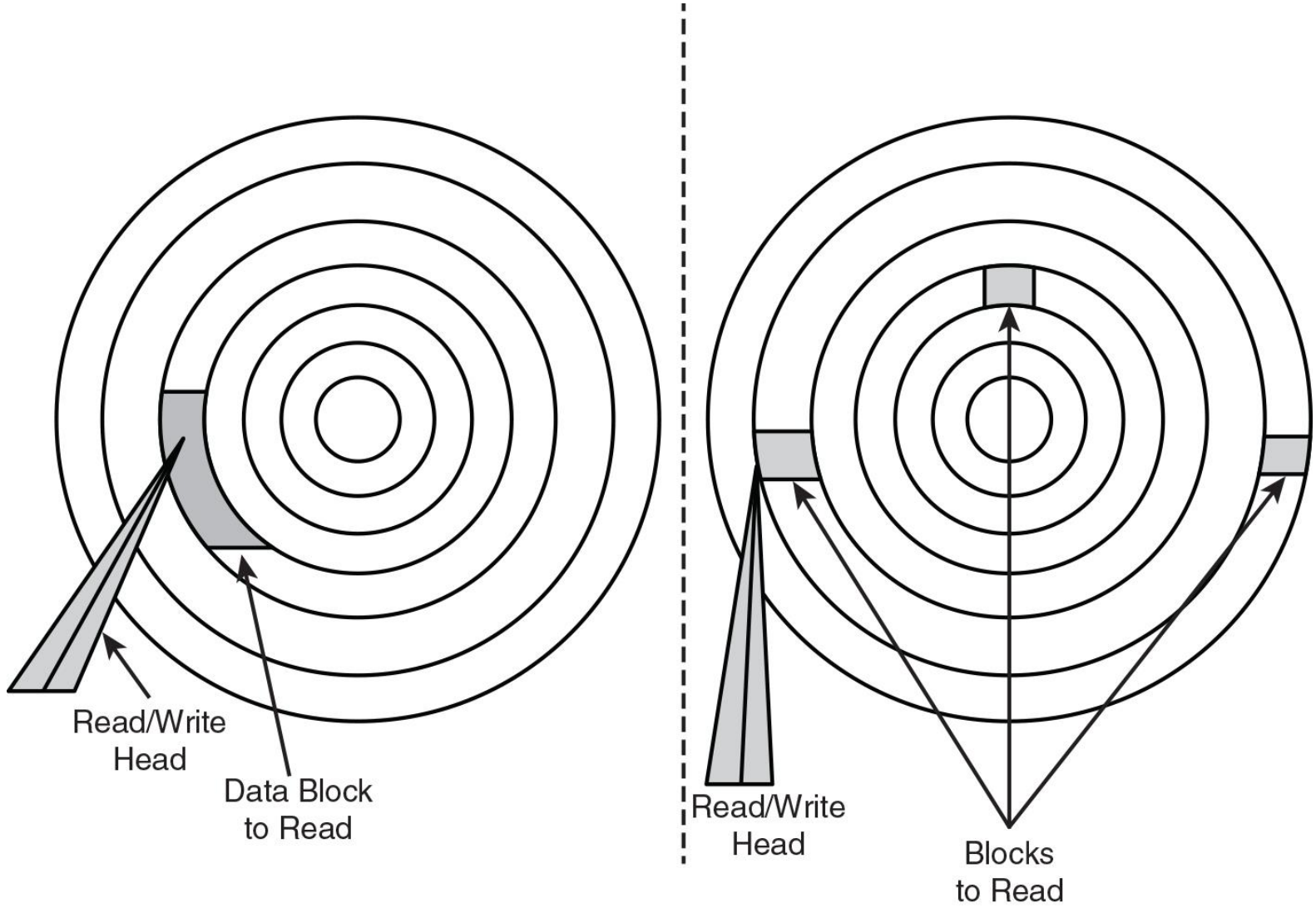

Figure 5.3 Reading a single block of data is faster than reading multiple blocks referenced by multiple keys.

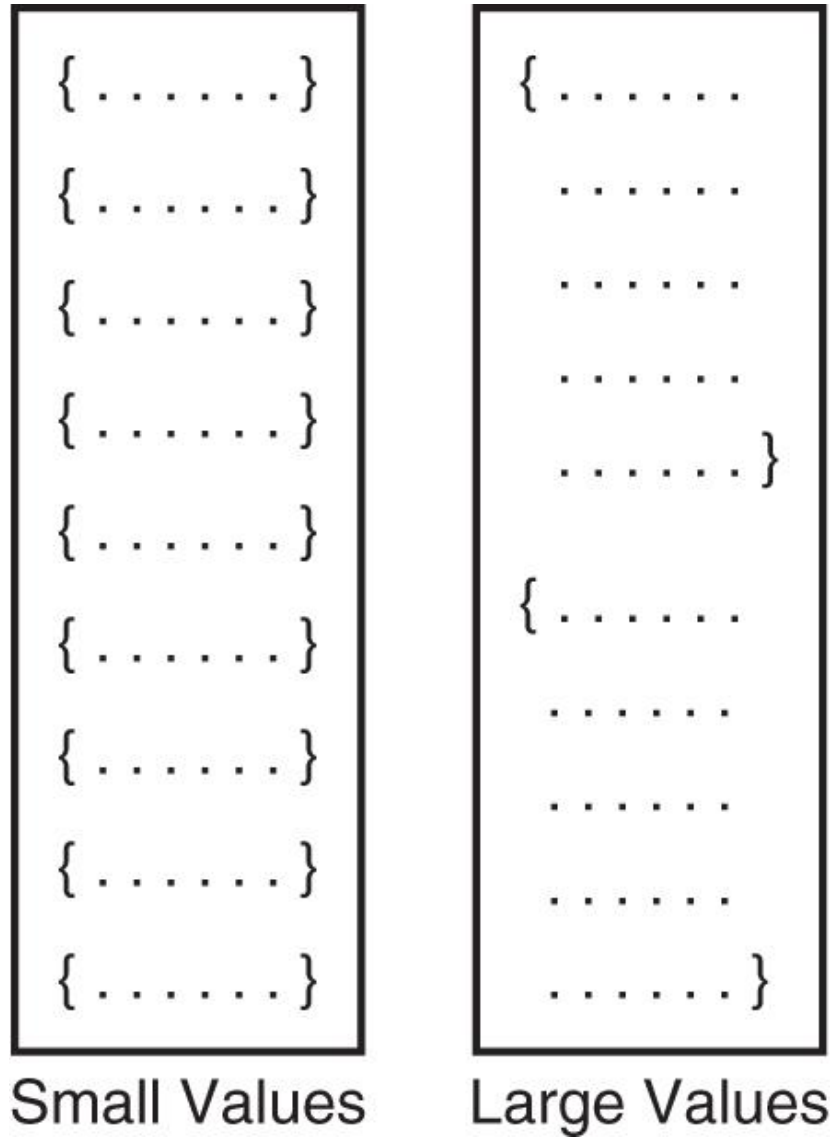

Figure 5.4 Data is read in blocks. Blocks may store a large number of smallsized values or few large-sized values. The former can lead to better performance if frequently used attributes are available in the cache.

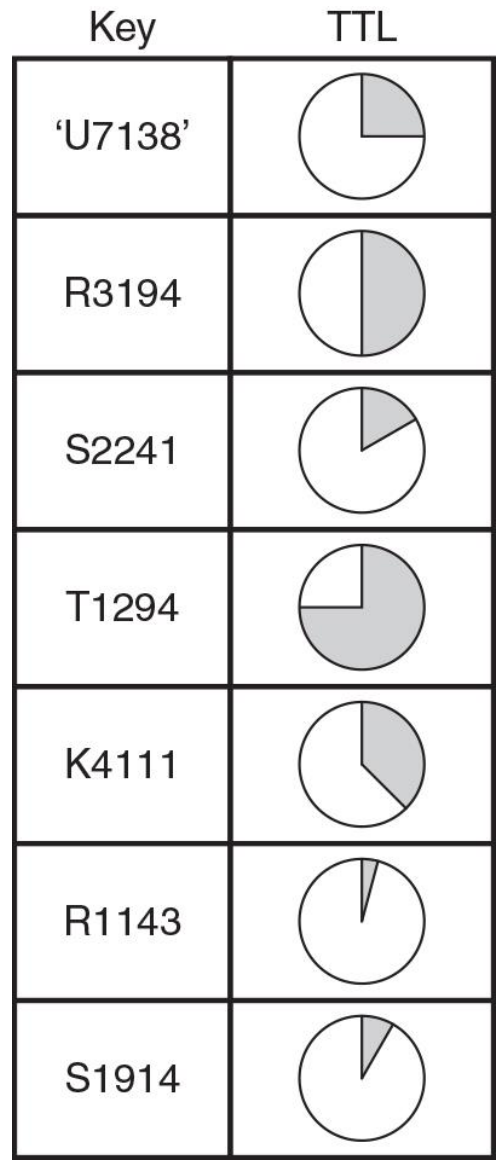

Figure 5.5 Time to Live keys are useful for allowing users to reserve a product or resource for a limited time while other operations, such as making a payment, complete.

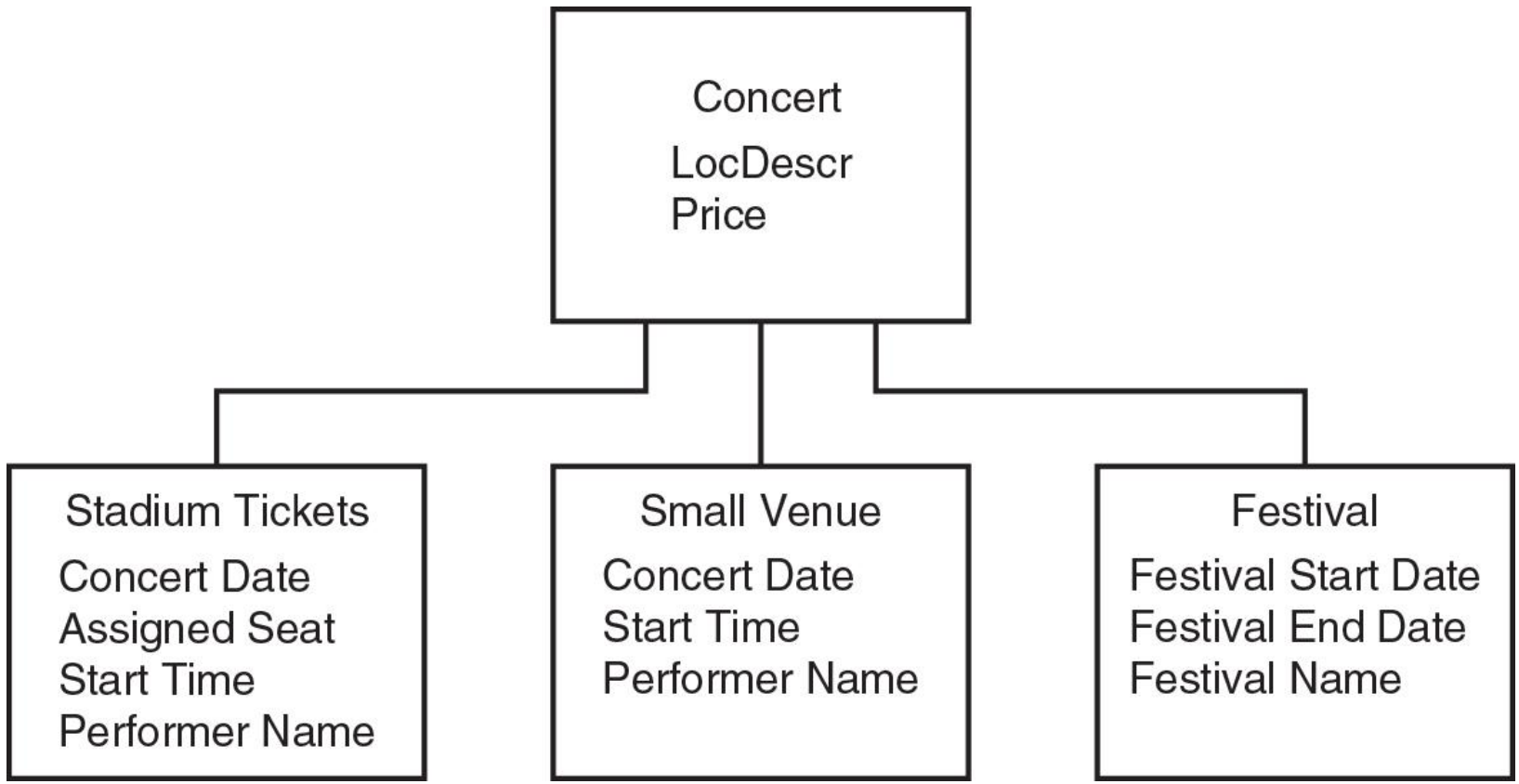

Figure 5.6 Entity subtypes can be modeled in relational databases as a common table with tables to store attributes of each subtype.

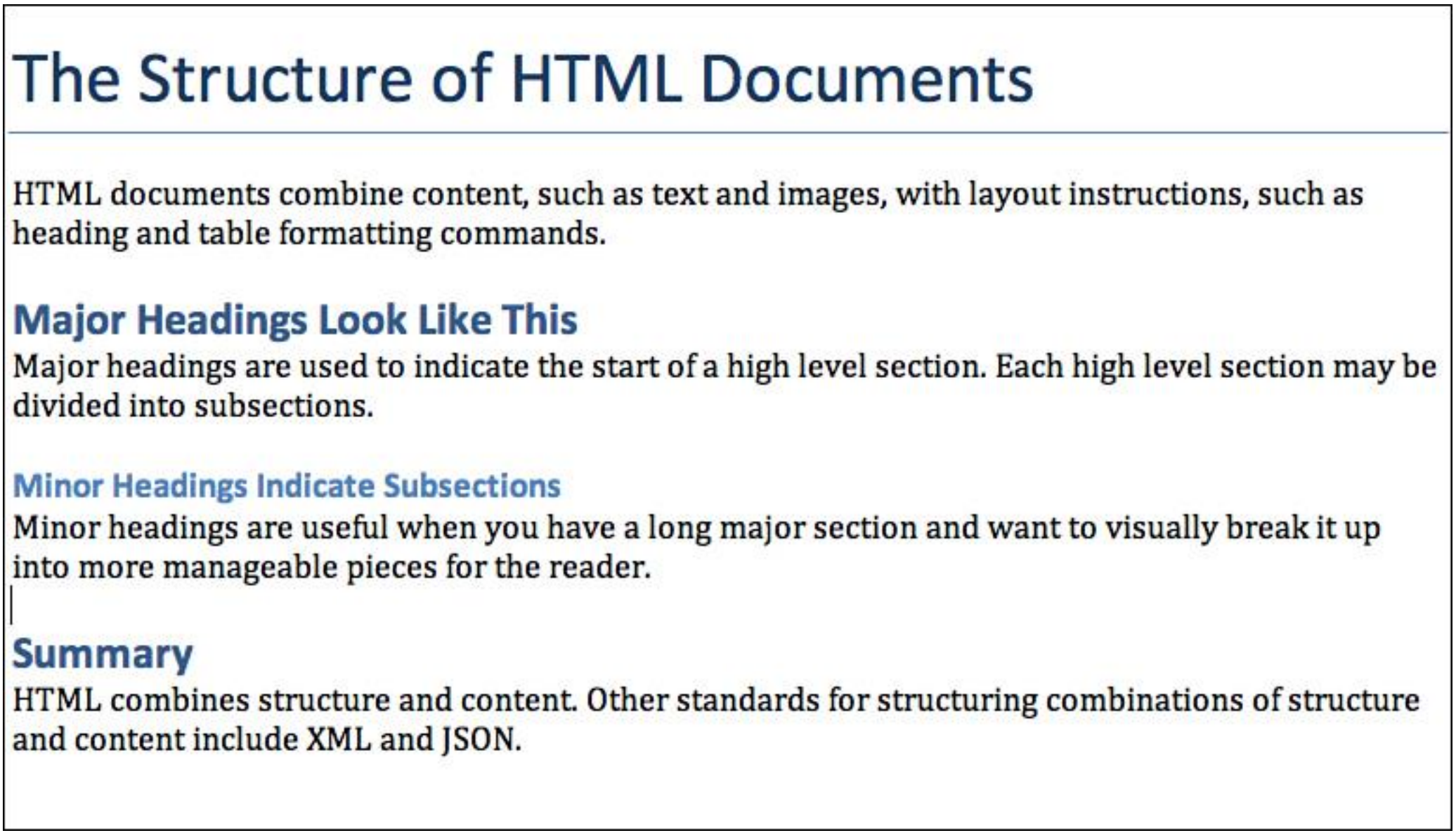

Figure 6.1 A simple example of an HTML document with basic formatting commands.

Document Database

```
Last Purchase Date > (Today () – 180)
```

Key Value

```
Customer_List =  Return Customer_ID
                    Where Last Purchase Date >
                       (Today () – 180);

Customer_Name =  Return Customer Name
                    Value Where
                       Customer_ID in
                          Customer_List;

Customer_Address =  Return Customer Address
                    Value Where
                       Customer_ID in
                          Customer_List;
```

Figure 6.2 Document databases require less code than key-value data stores to query multiple attributes.

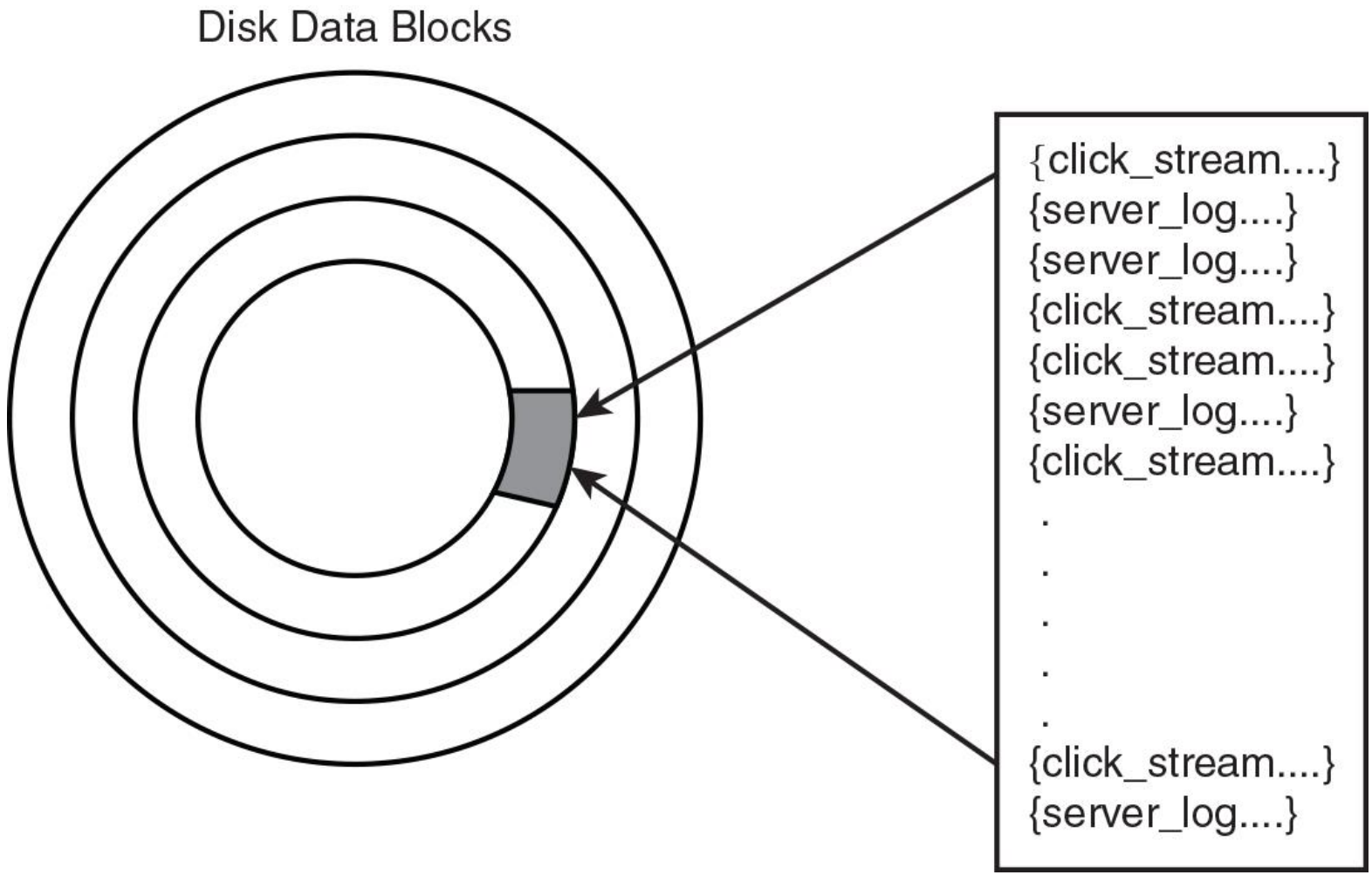

Figure 6.3 Mixing document types can lead to multiple document types in a disk data block. This can lead to inefficiencies because data is read from disk but not used by the application that filters documents based on type.

High-Level Branching

```
   doc.
If (doc_type = 'click_stream'):
     process_click_stream (doc)
Else
     process_server_log (doc)
```

Lower-Level Branching

```
book.title = doc.title
book.author = doc.author
book.year = doc.publication_year
book.publisher = doc.publisher
book.descr = book.title + book.author + book.year + book.publisher
if (doc.ebook = true);
  book.descr = book.descr + doc.ebook_size
```

Figure 6.4 High-level branching in functions manipulating documents can indicate a need to create separate collections. Branching at lower levels is common when some documents have optional attributes.

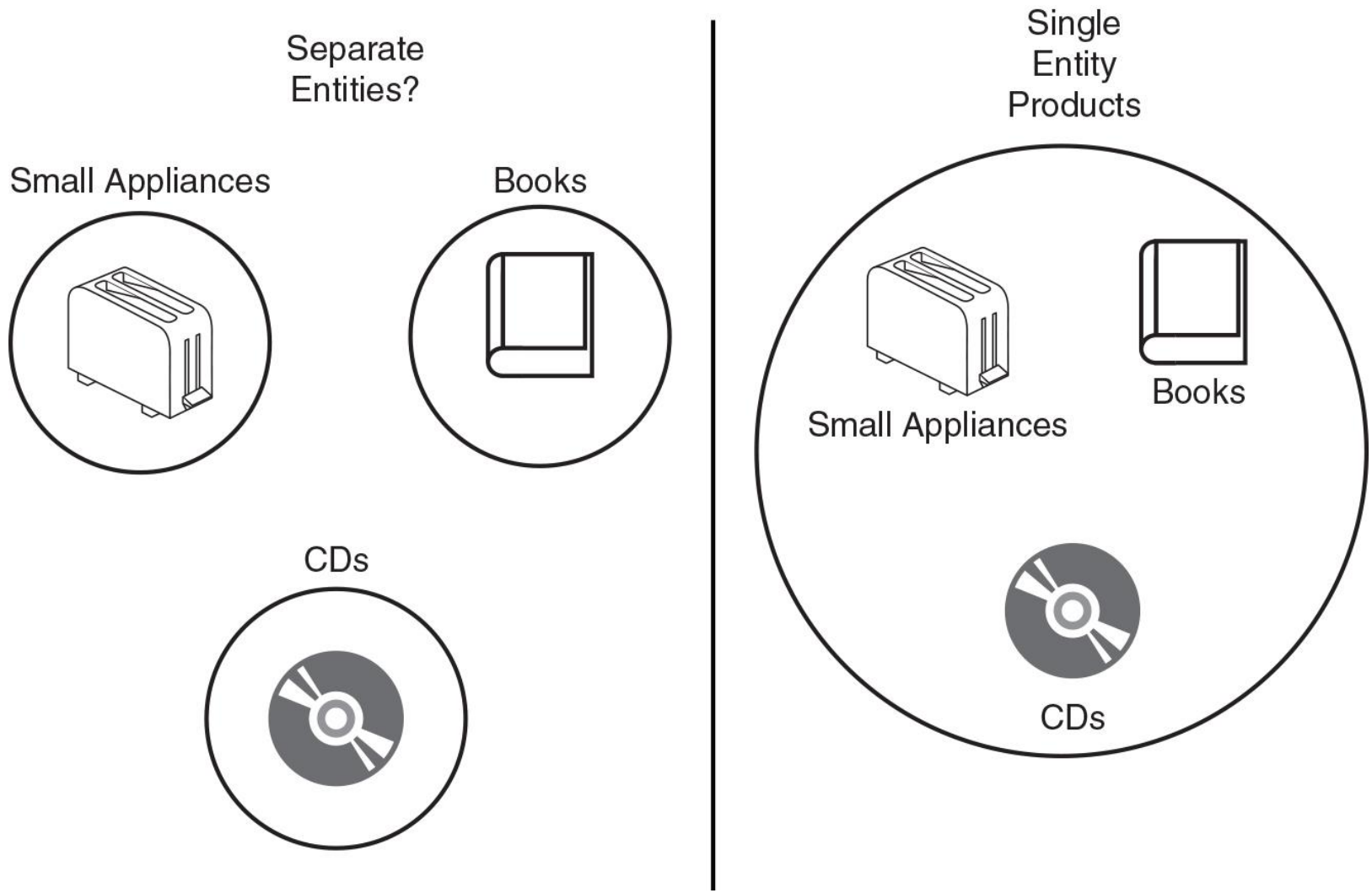

Figure 6.5 When is a toaster the same as a database design book? When they are treated the same by application queries. Queries can help guide the organization of documents and collections.

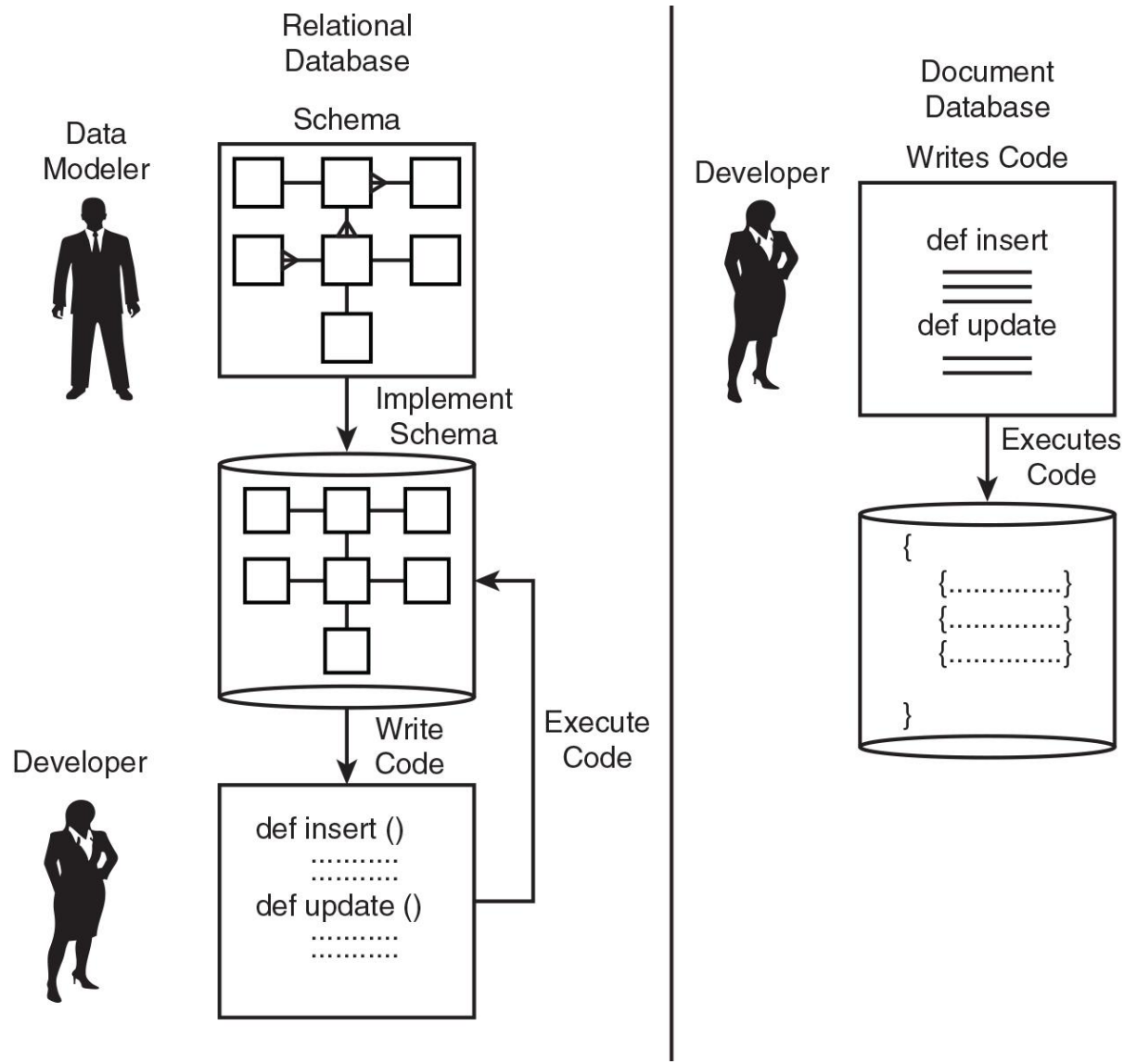

Figure 6.6 Relational databases require an explicitly defined schema, but document databases do not.

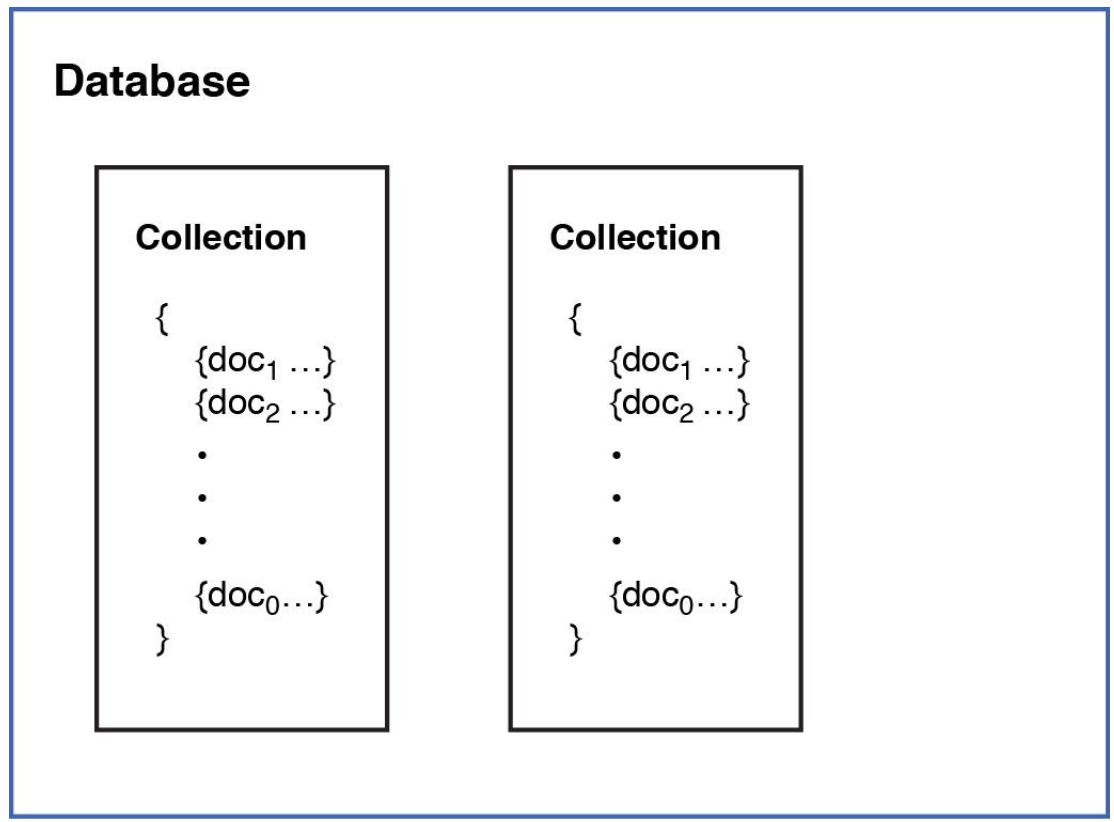

Figure 6.7 The database is the highest-level logical structure in a document database and contains collections of documents.

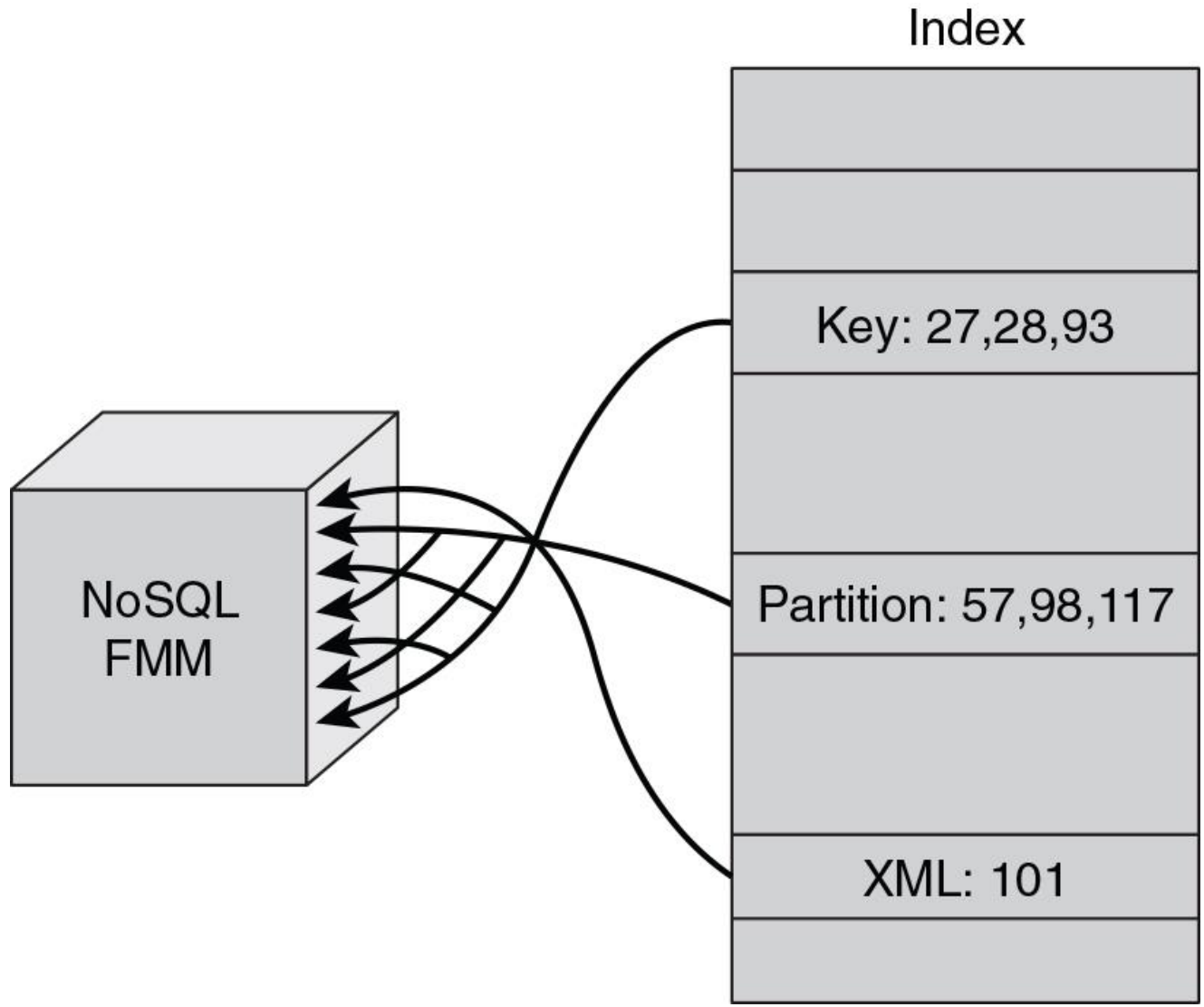

Figure 7.1 Indexes map attributes, such as key terms, to related information, such as page numbers. Using an index is faster than scanning an entire book for key terms.

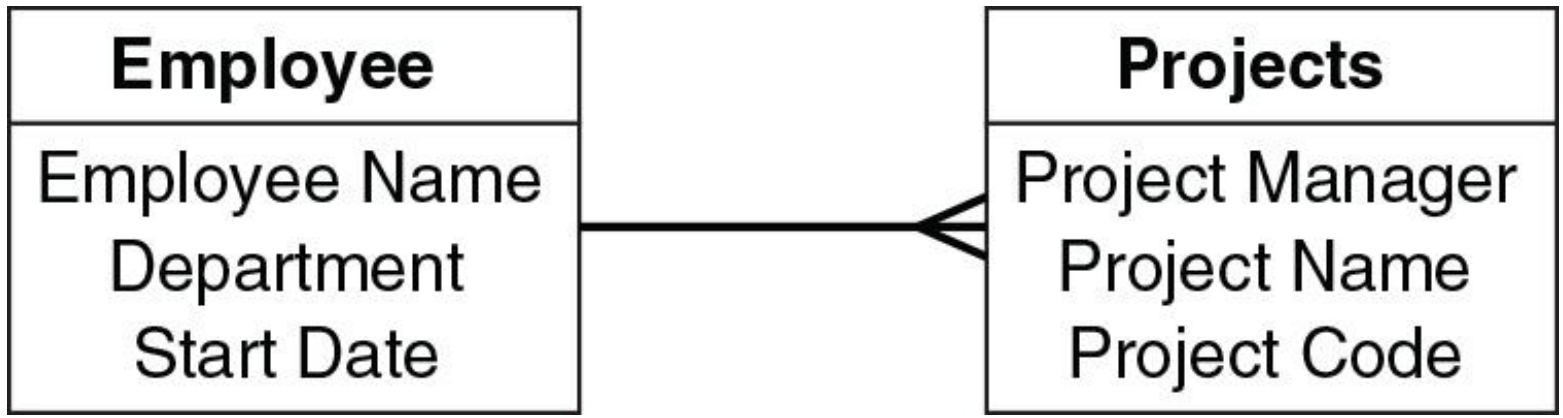

Figure 7.2 Relational data models separate data about different entities into separate tables. This requires looking up information in both tables using a process known as joining.

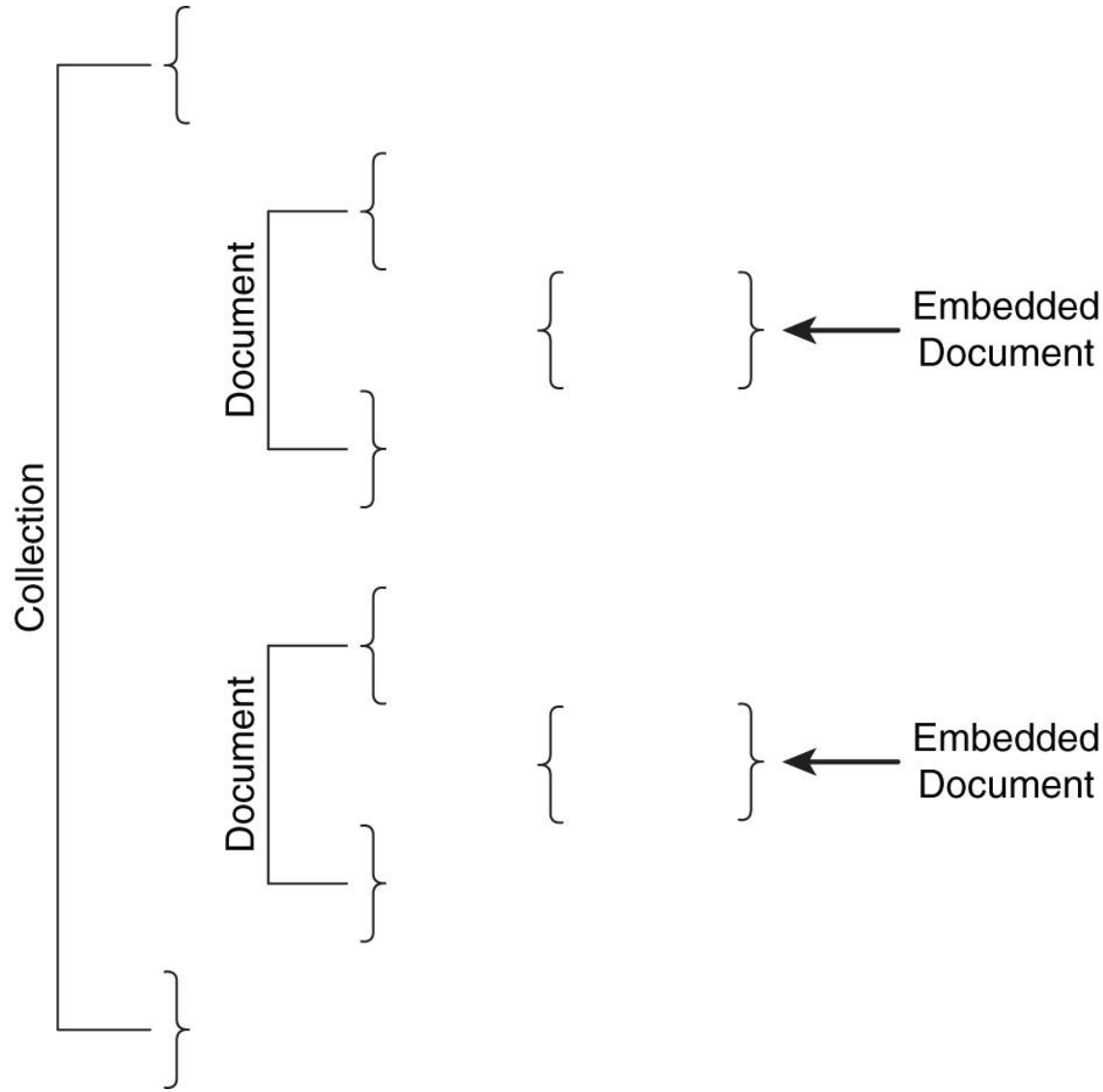

Figure 7.3 Embedded documents are documents within a document. Embedding is used to efficiently store and retrieve data that is frequently used together.

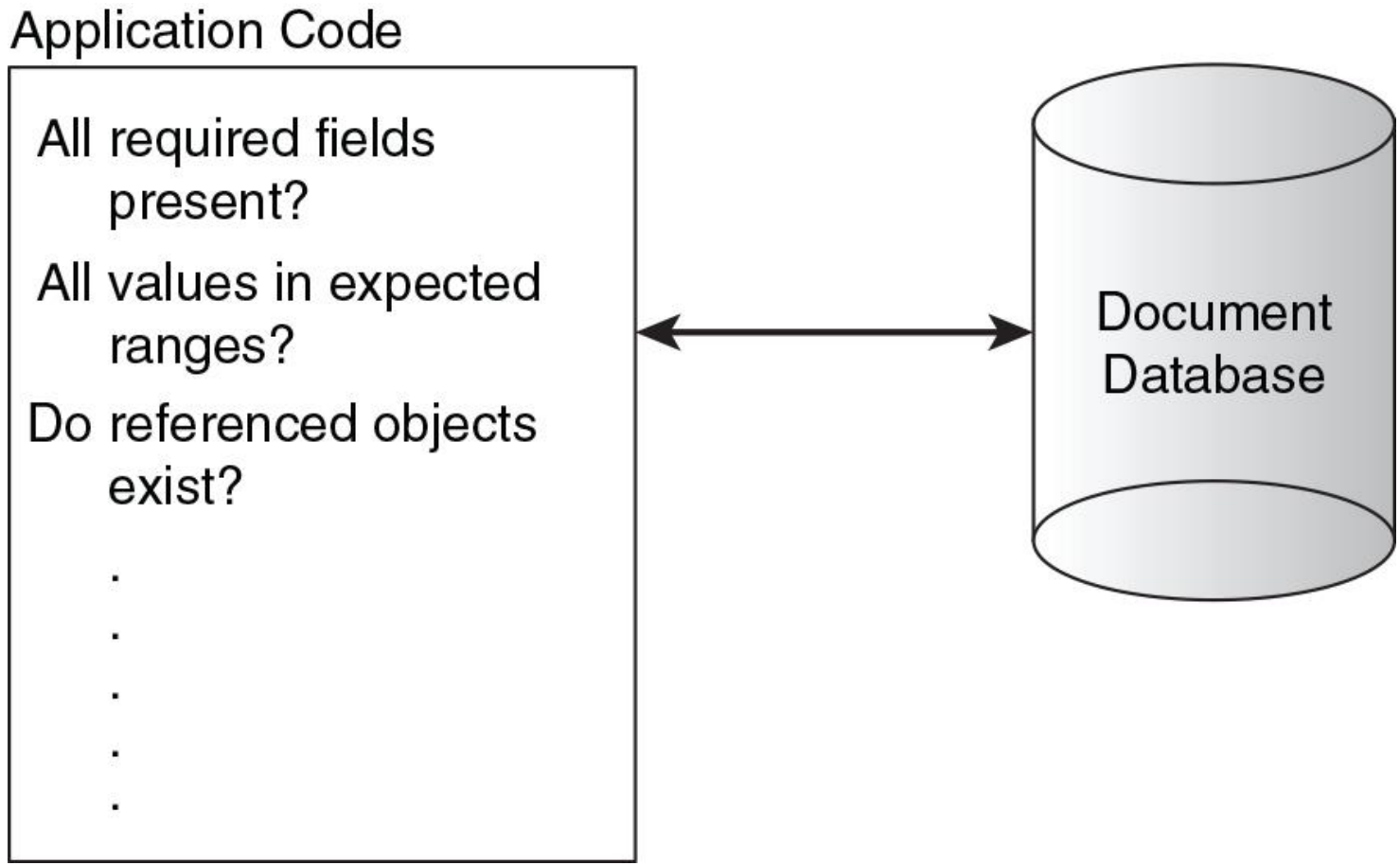

Figure 7.4 Data validation code and error-handling code is used throughout applications to compensate for the lack of automatic, schema-based validation checks.

Polymorphic

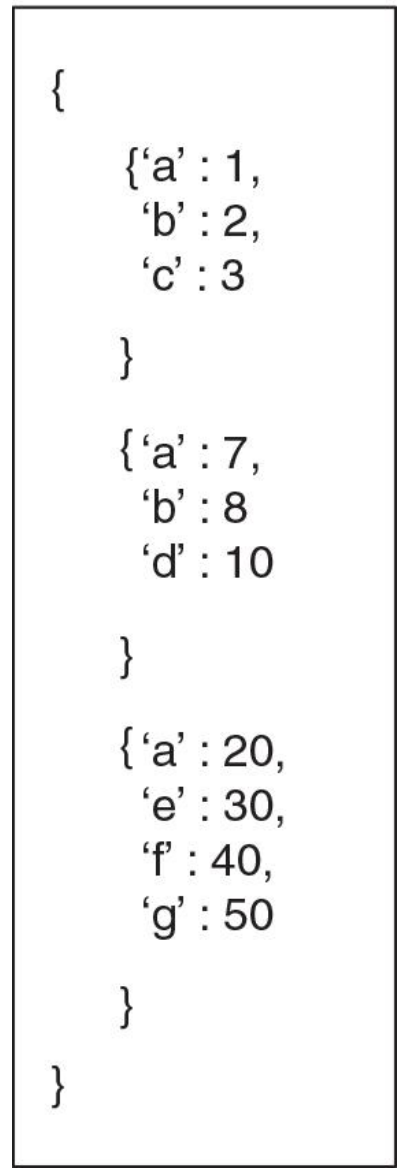

Figure 7.5 Schemaless means there is no formal definition of structure. Polymorphic schema means there are many document structures implied in the set of documents that occur in a collection.

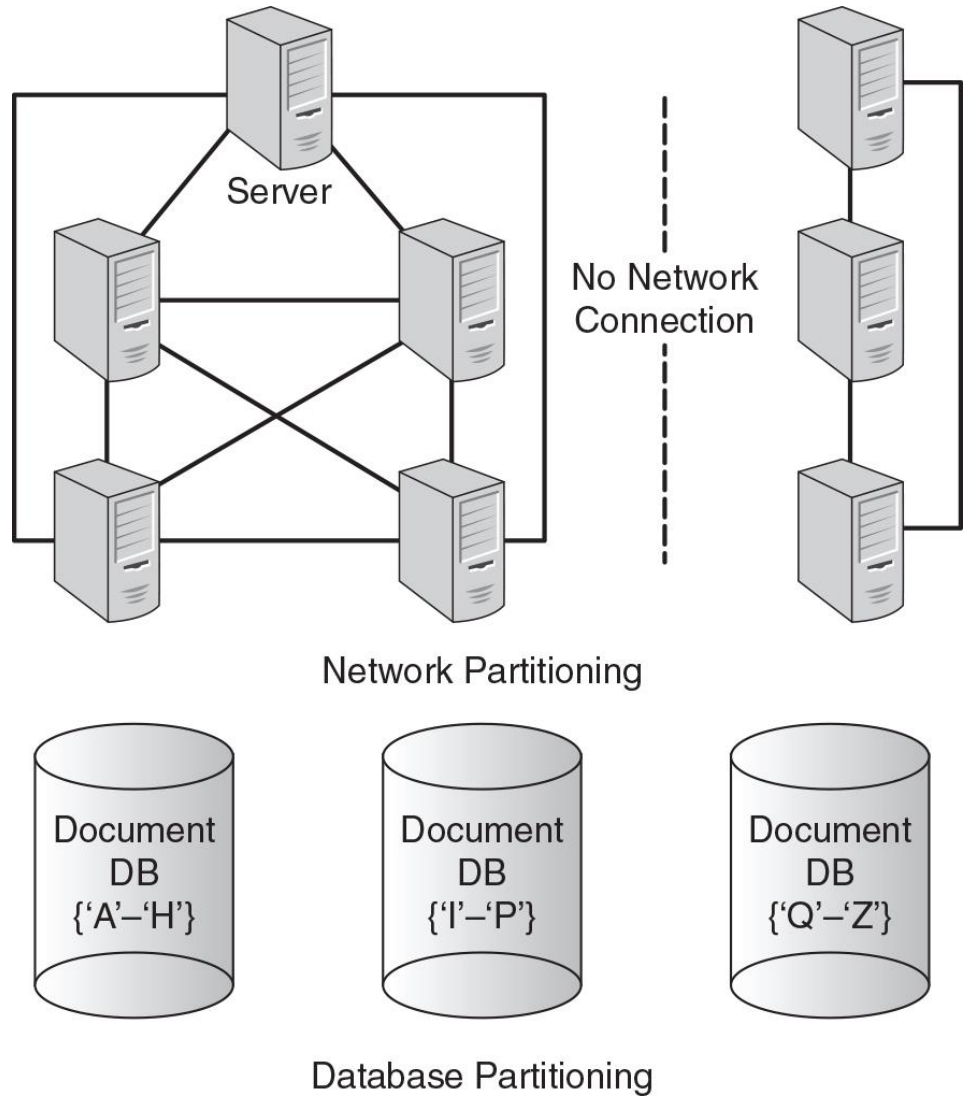

Figure 7.6 The term partitioning has multiple meanings that are distinguished by the context, such as in the context of networks versus in the context of databases.

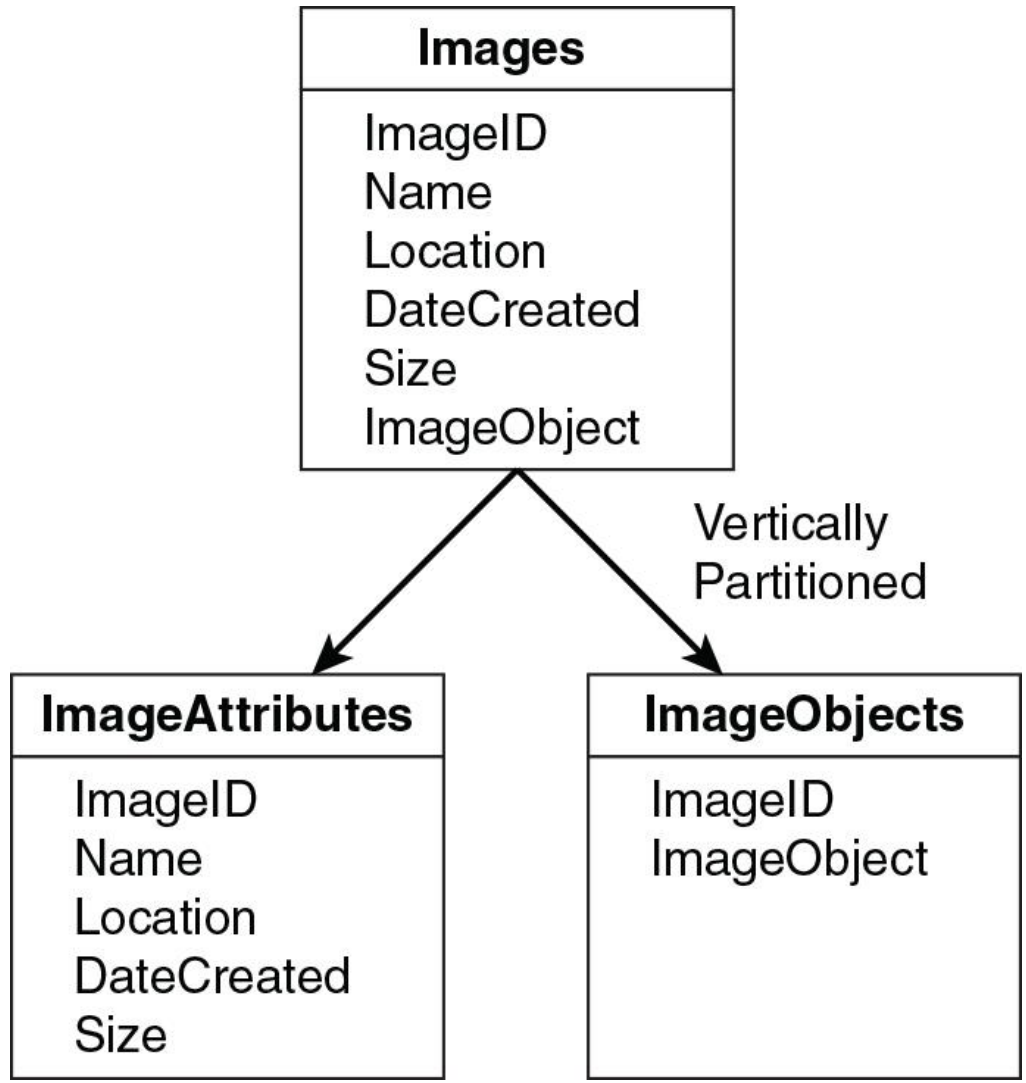

Figure 7.7 Vertical partitioning is typically used with relational tables because they have a fixed structure.

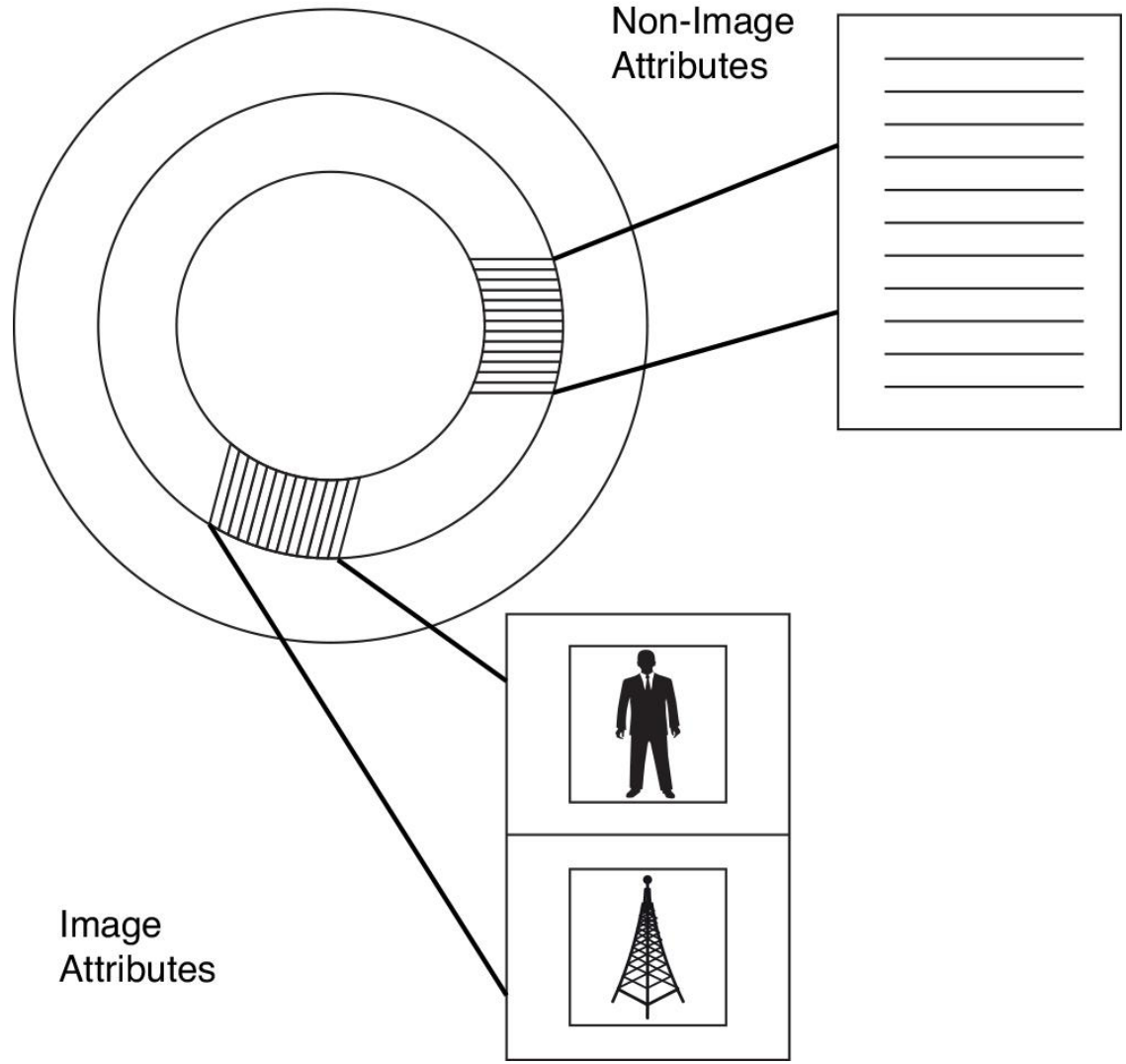

Figure 7.8 Separating columns into separate tables can improve the efficiency of reads because data that is not needed (for example, an image object) is not retrieved along with data that is likely needed (for example, image attributes).

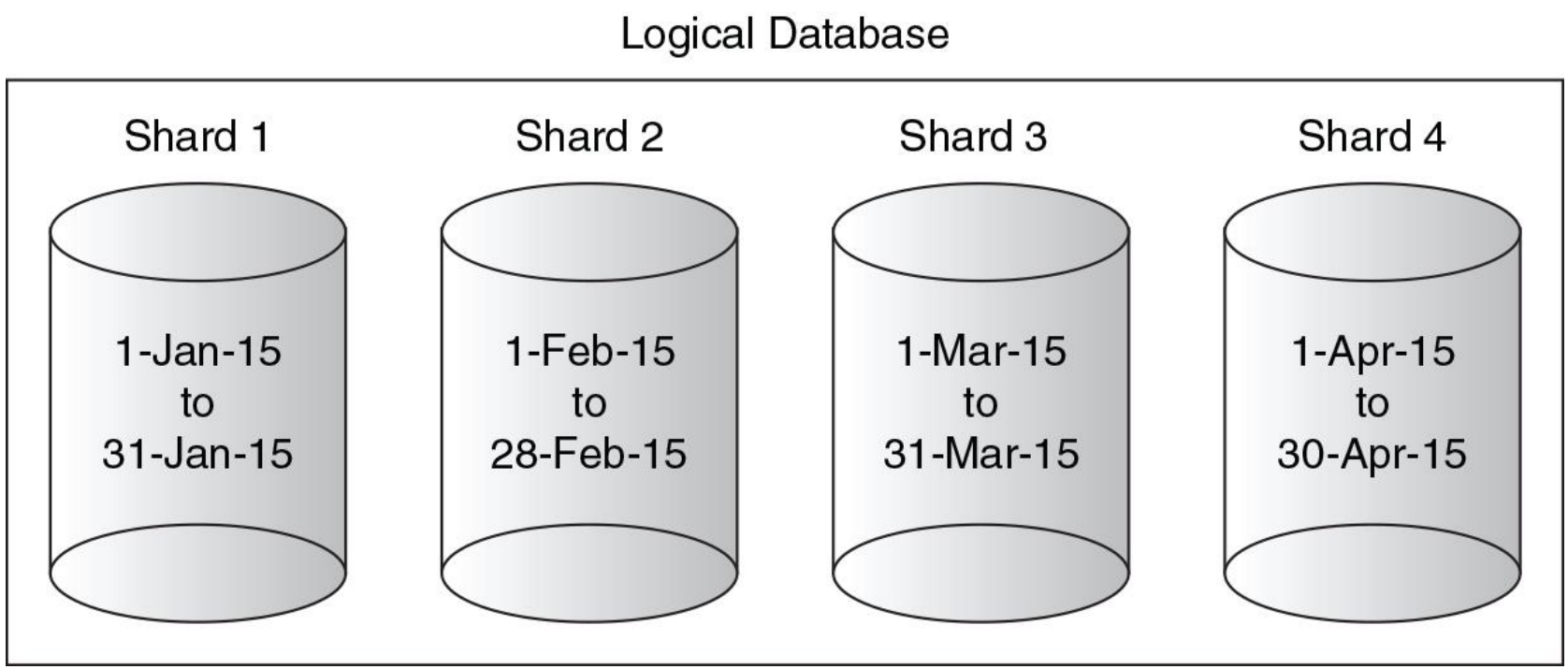

Figure 7.9 Horizontal sharding splits a database by document or row and distributes sections of the database, known as shards, to different servers. When a cluster implements replication, a single shard will be available on multiple servers.

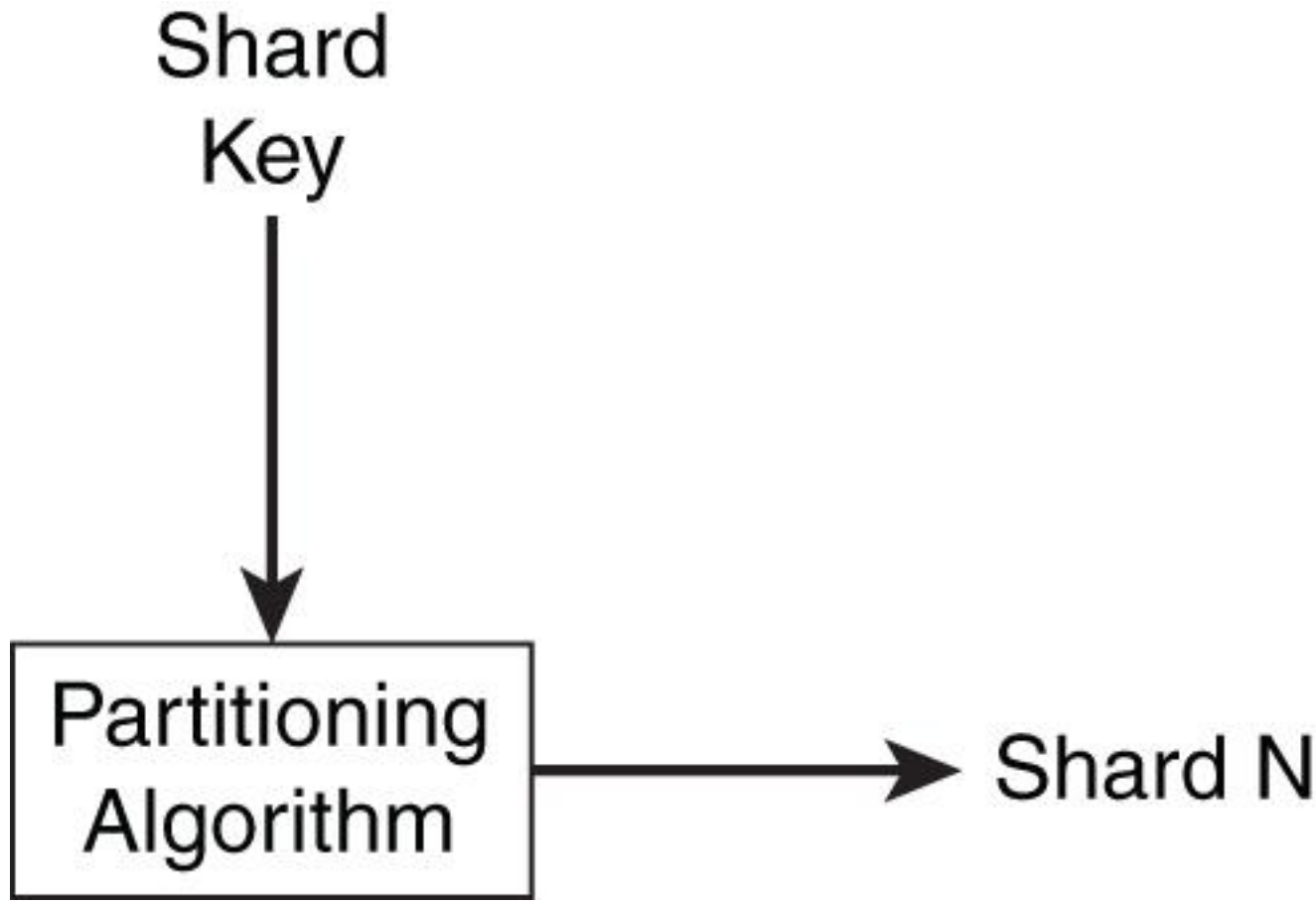

Figure 7.10 Shard keys are input to the partitioning algorithm that outputs a shard.

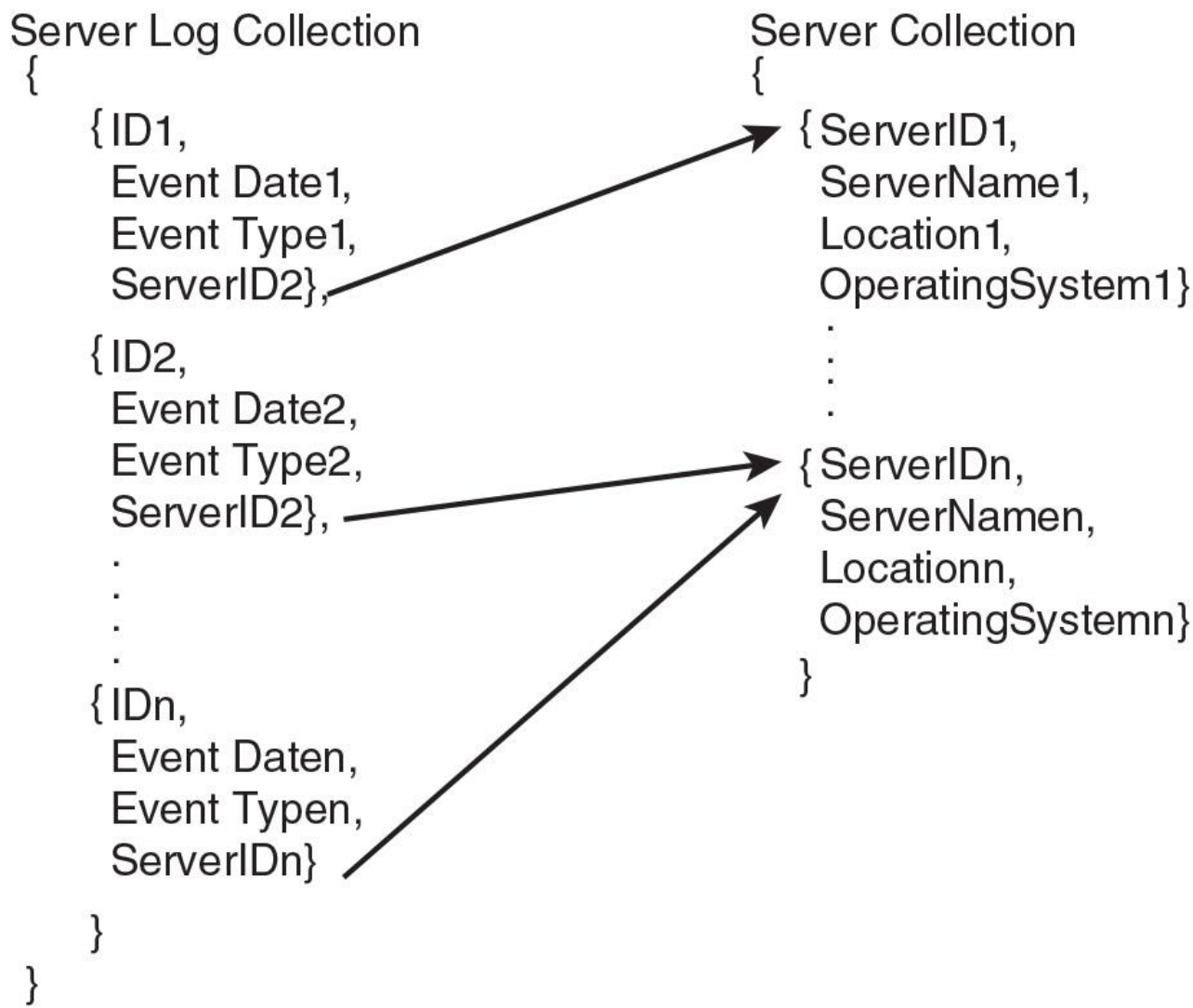

Figure 7.11 Normalized documents reduce redundant data by referencing a single copy of data rather than repeating it in each document.

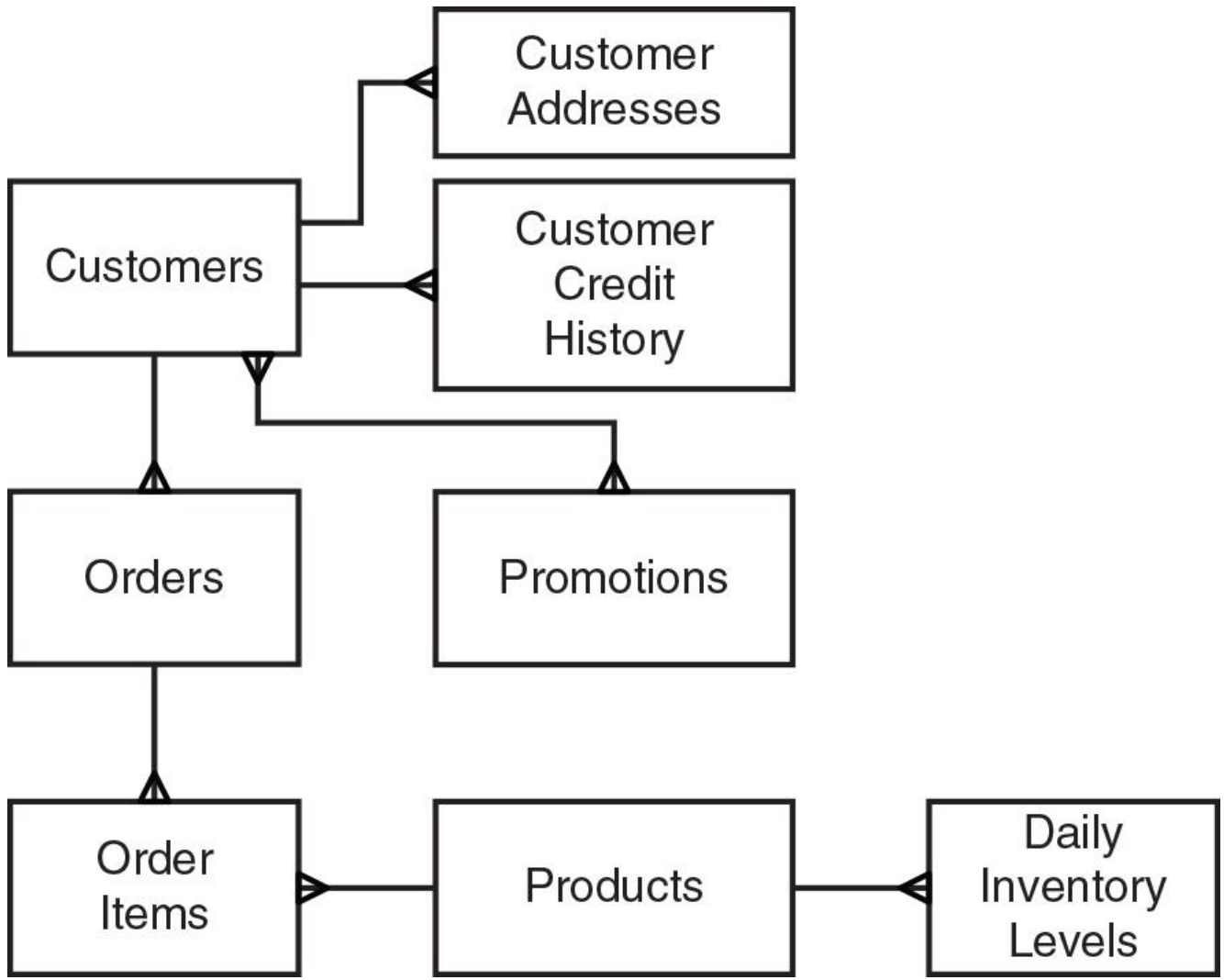

Figure 8.1 Normalized databases have separate tables for entities. Data about entities is isolated and redundant data is avoided.

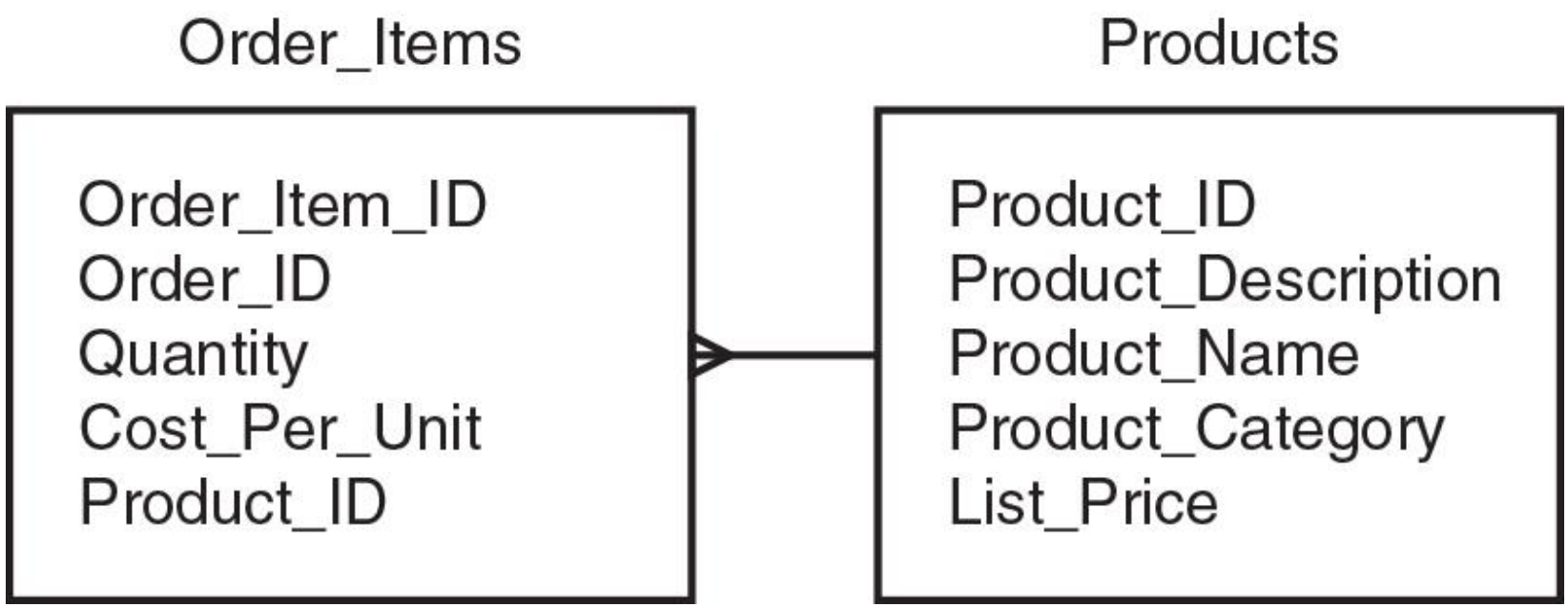

Figure 8.2 Products and Order Items are in a one-to-many relationship. To retrieve Product data about an Order item, they need to share an attribute that serves as a common reference. In this case, Product_ID is the shared attribute.

Order Items				
Order_Item_ID	**Order_ID**	**Quantity**	**Cost_Per_Unit**	**Product_ID**
1298	789	1	$25.99	345
1299	789	2	$20.00	372
1300	790	1	$12.50	591
1301	790	1	$20.00	372
1302	790	3	$12.99	413

Products				
Product_ID	**Product_Description**	**Product_Name**	**Product_Category**	**List_Price**
345	Easy clean tablet cover that fits most 10" Android tablets.	Easy Clean Cover	Electronic Accessories	25.99
372	Lightweight blue ear buds with comfort fit.	Acme Ear Buds	Electronic Accessories	20
413	Set of 10 dry erase markers.	10-Pack Markers	Office Supplies	15
420	60"×48" whiteboard with marker and eraser holder.	Large Whiteboard	Office Supplies	56.99
591	Pack of 100 individually wrapped screen wipes.	Screen Clean Wipes	Office Supplies	12.99

Figure 8.3 To be joined, tables must share a common value known as a foreign key.

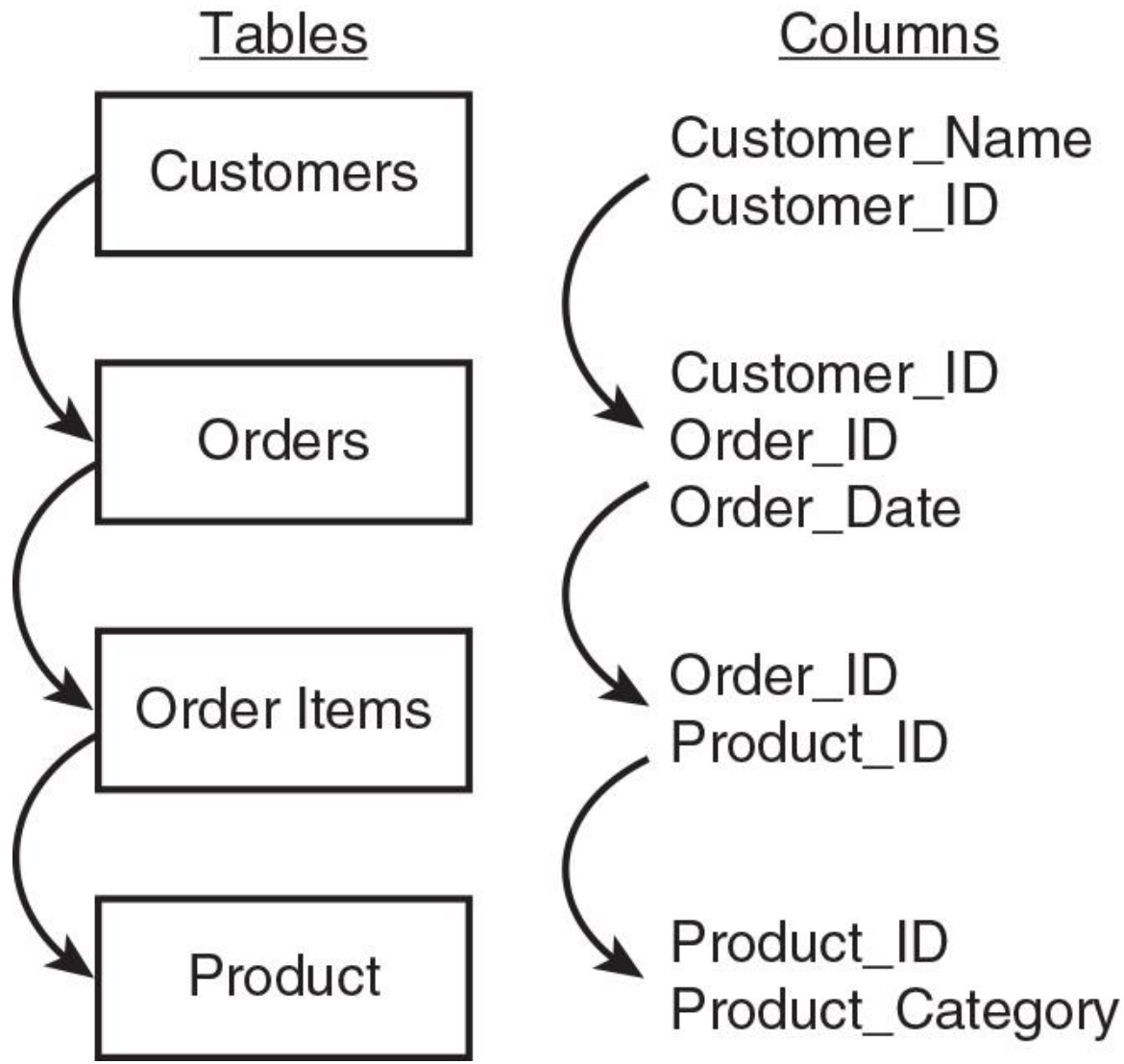

Figure 8.4 Analyzing customers who bought a particular type of product requires three joins between four tables.

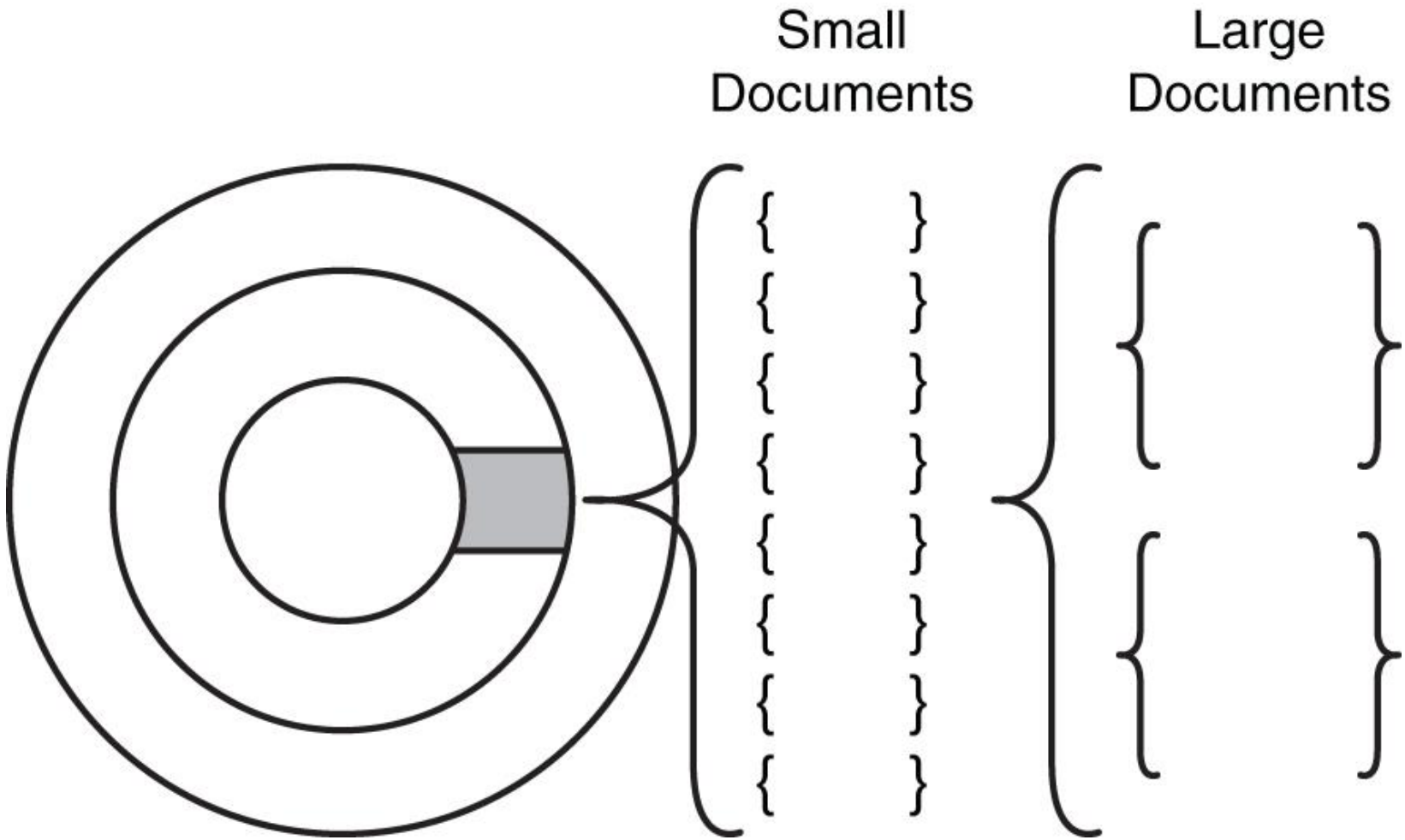

Figure 8.5 Large documents can lead to fewer documents retrieved when a block of data is read from persistent storage. This can increase the total number of data block reads to retrieve a collection or subset of collections.

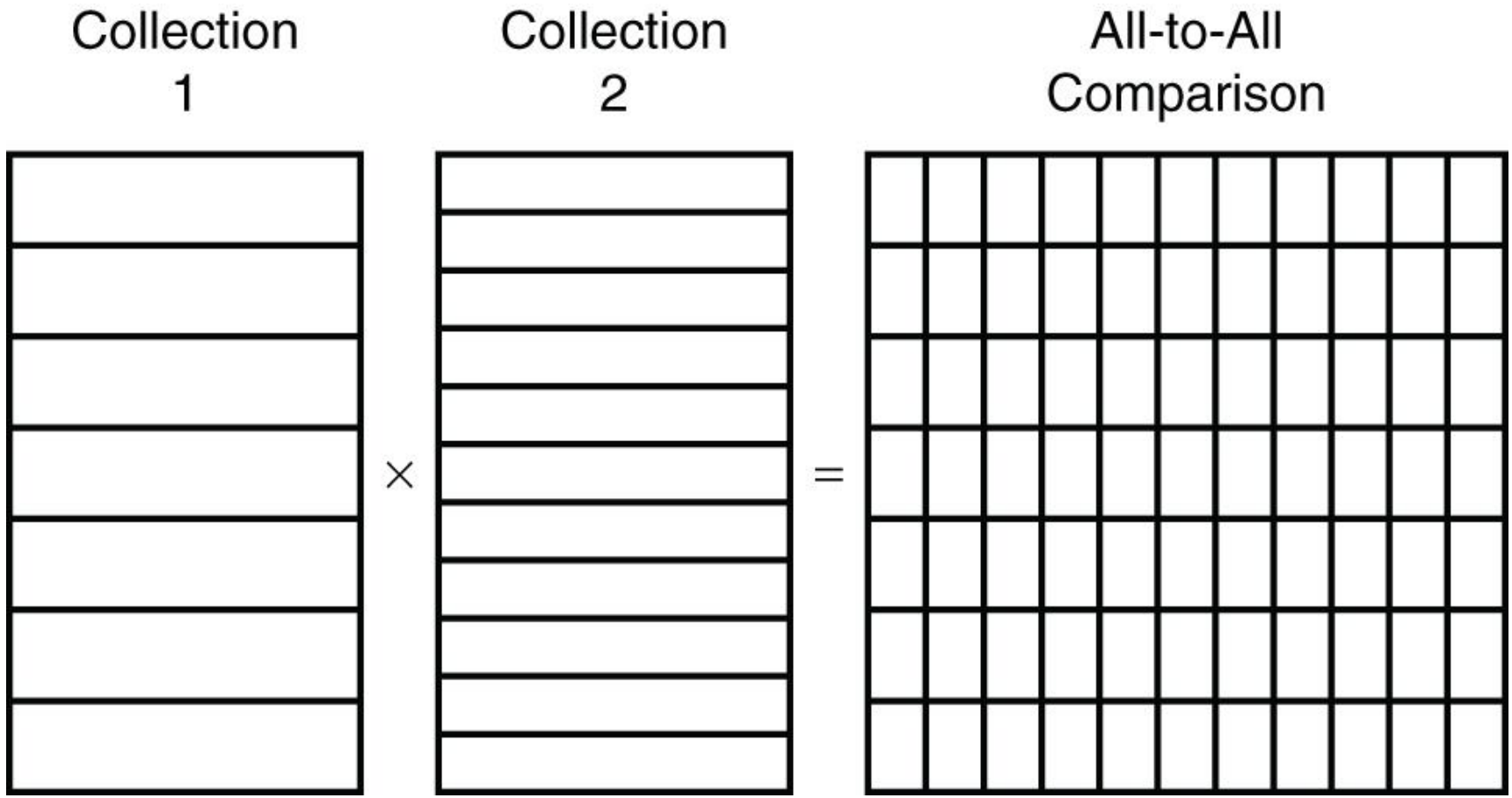

Figure 8.6 Simple join operations that compare all documents in one collection to all documents in another collection can lead to poor performance on large collections. Joins such as this can be improved by using indexes, filtering, and, in some cases, sorting.

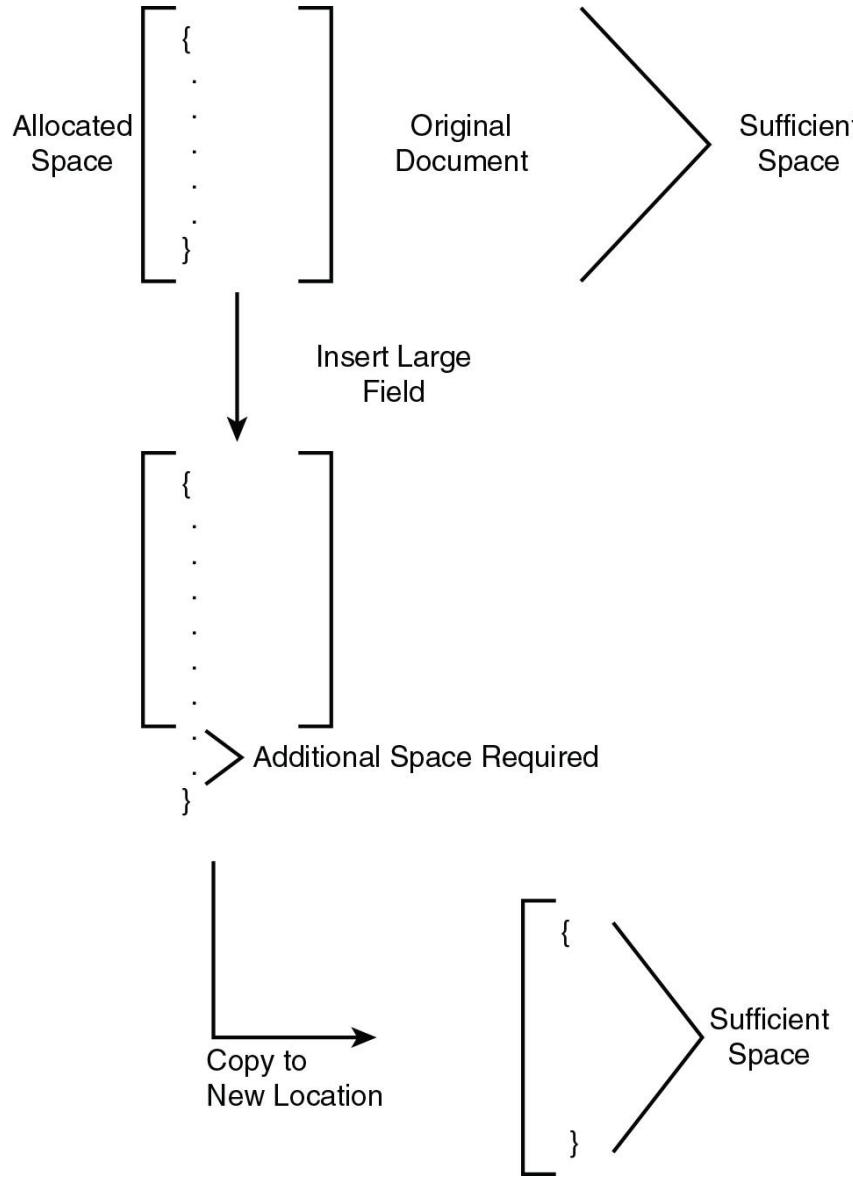

Figure 8.7 When documents grow larger than the amount of space allocated for them, they may be moved to another location. This puts additional load on the storage systems and can adversely affect performance.

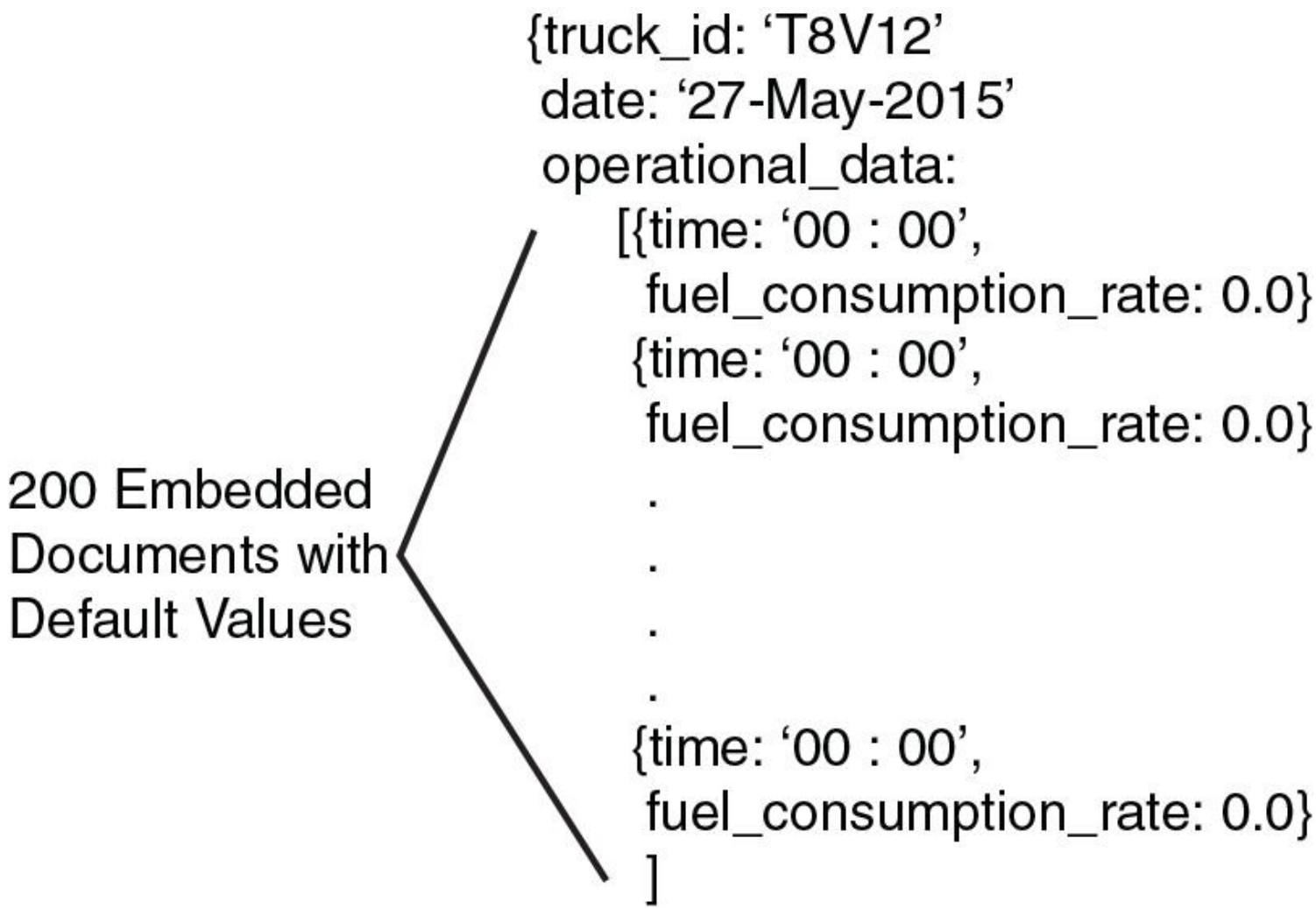

Figure 8.8 Creating documents with sufficient space for anticipated growth reduces the need to relocate documents.

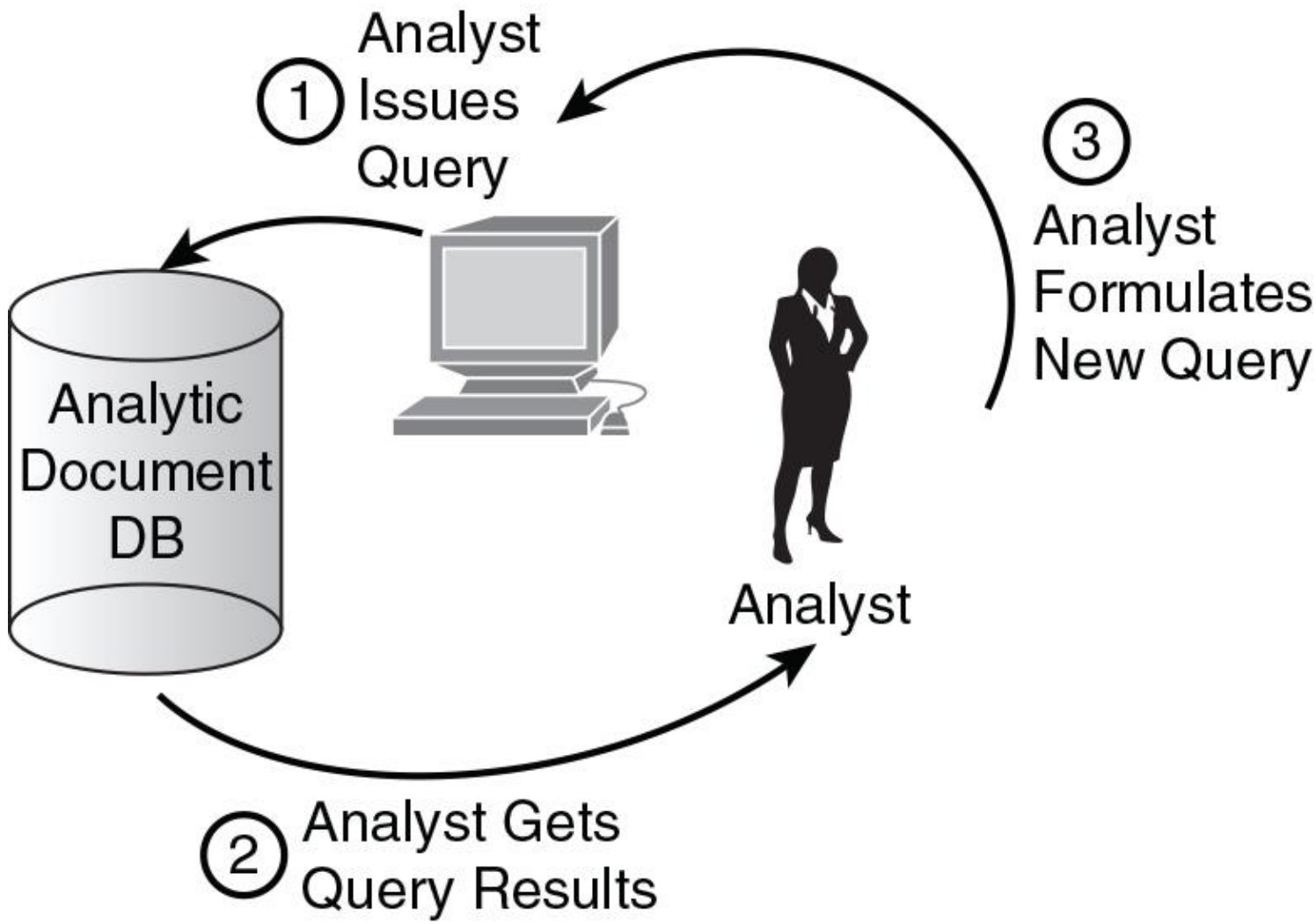

Figure 8.9 Querying analytic databases is an iterative process. Virtually any field could potentially be used to filter results. In such cases, indexes may be created on most fields.

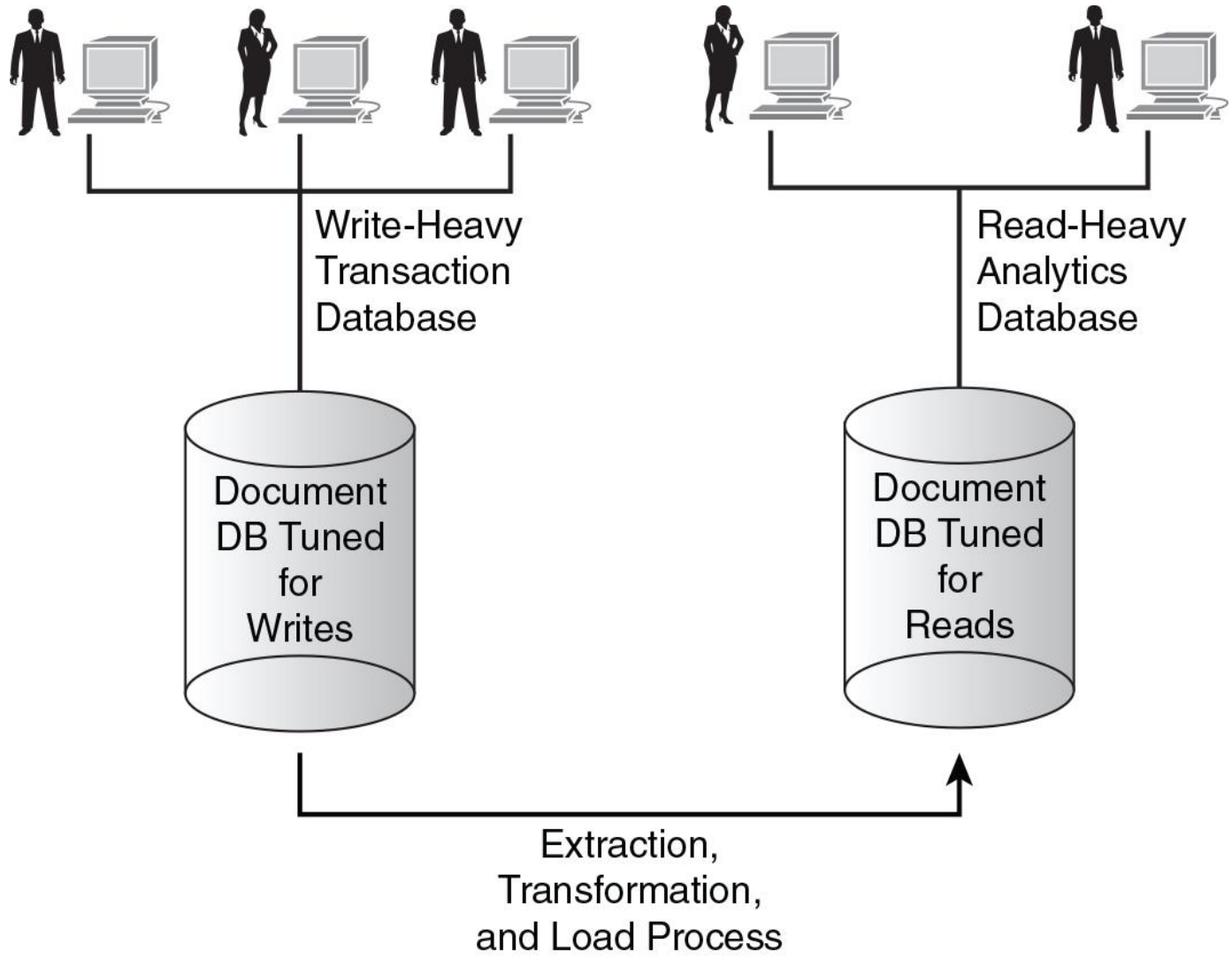

Figure 8.10 When both write-heavy and read-heavy applications must be supported, a two-database solution may be the best option.

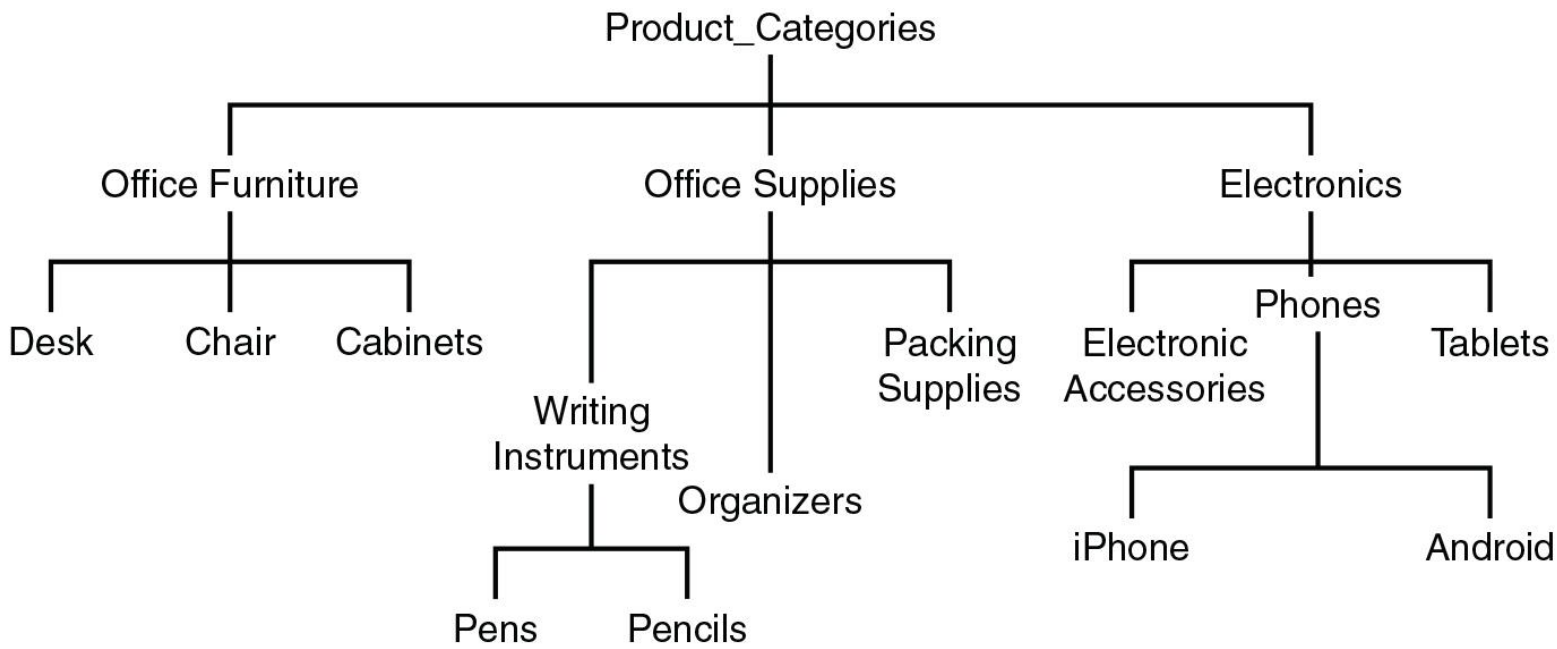

Figure 8.11 Hierarchies describe parent-child or part-subpart relations.

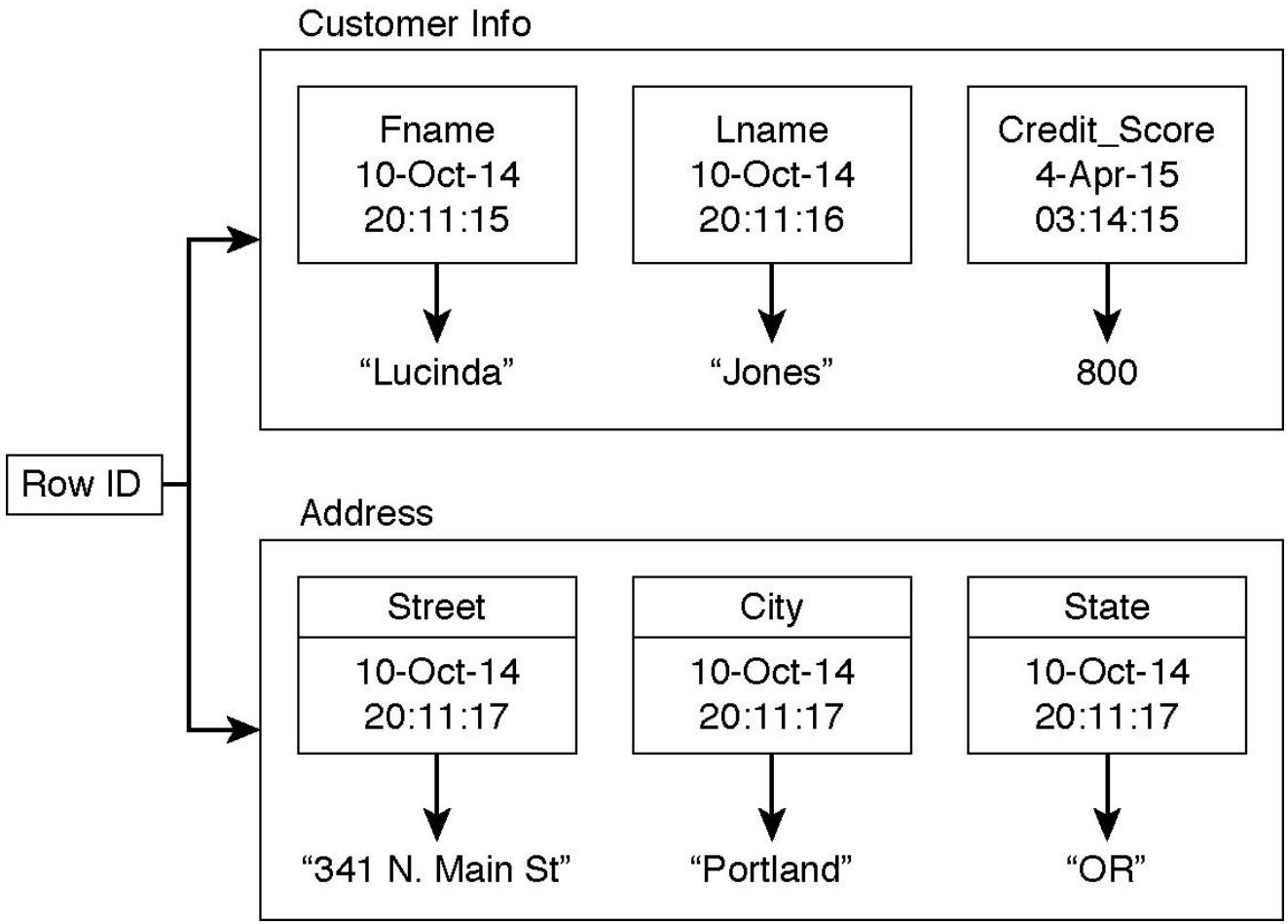

Figure 9.1 A row in a column family database is organized as a set of column families. Column families consist of related columns; a data value is indexed by a row, a column name, and a time stamp.

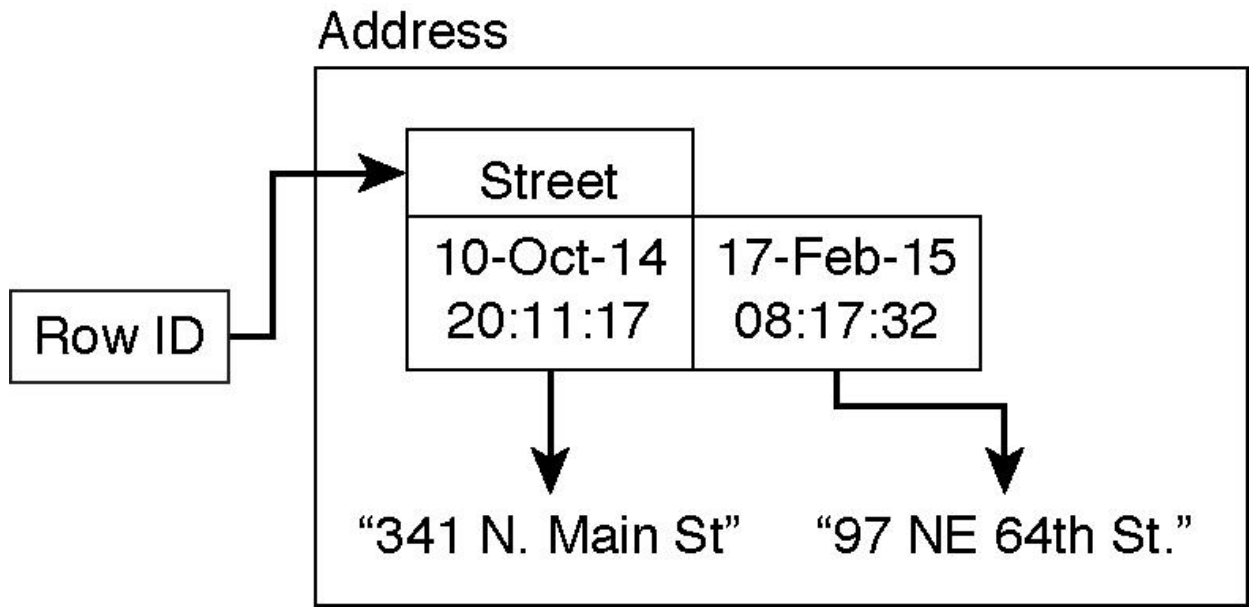

Figure 9.2 Data values are indexed by row identifier, column name, and time stamp. Multiple versions of a column value can exist. The latest version is returned by default when the column value is queried.

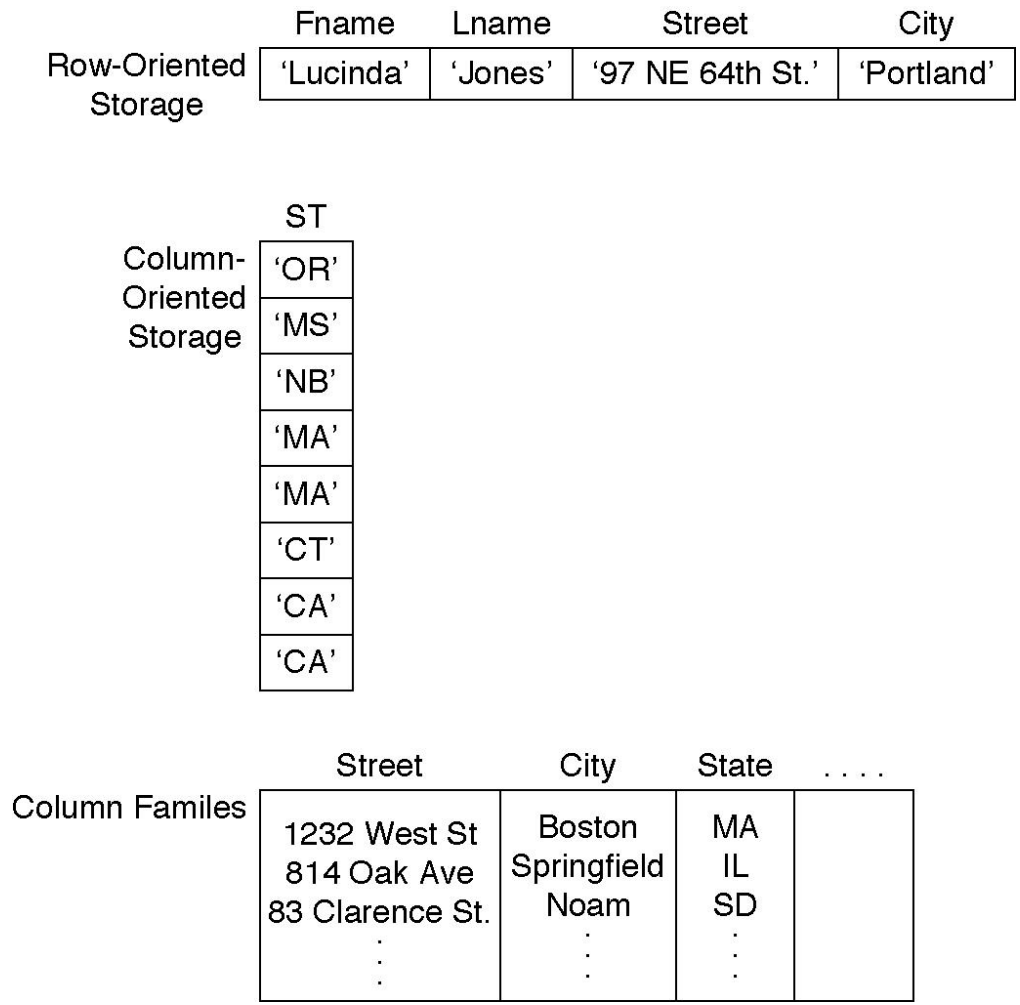

Figure 9.3 Different storage models offer different benefits. Choose a storage model that meets your query needs.

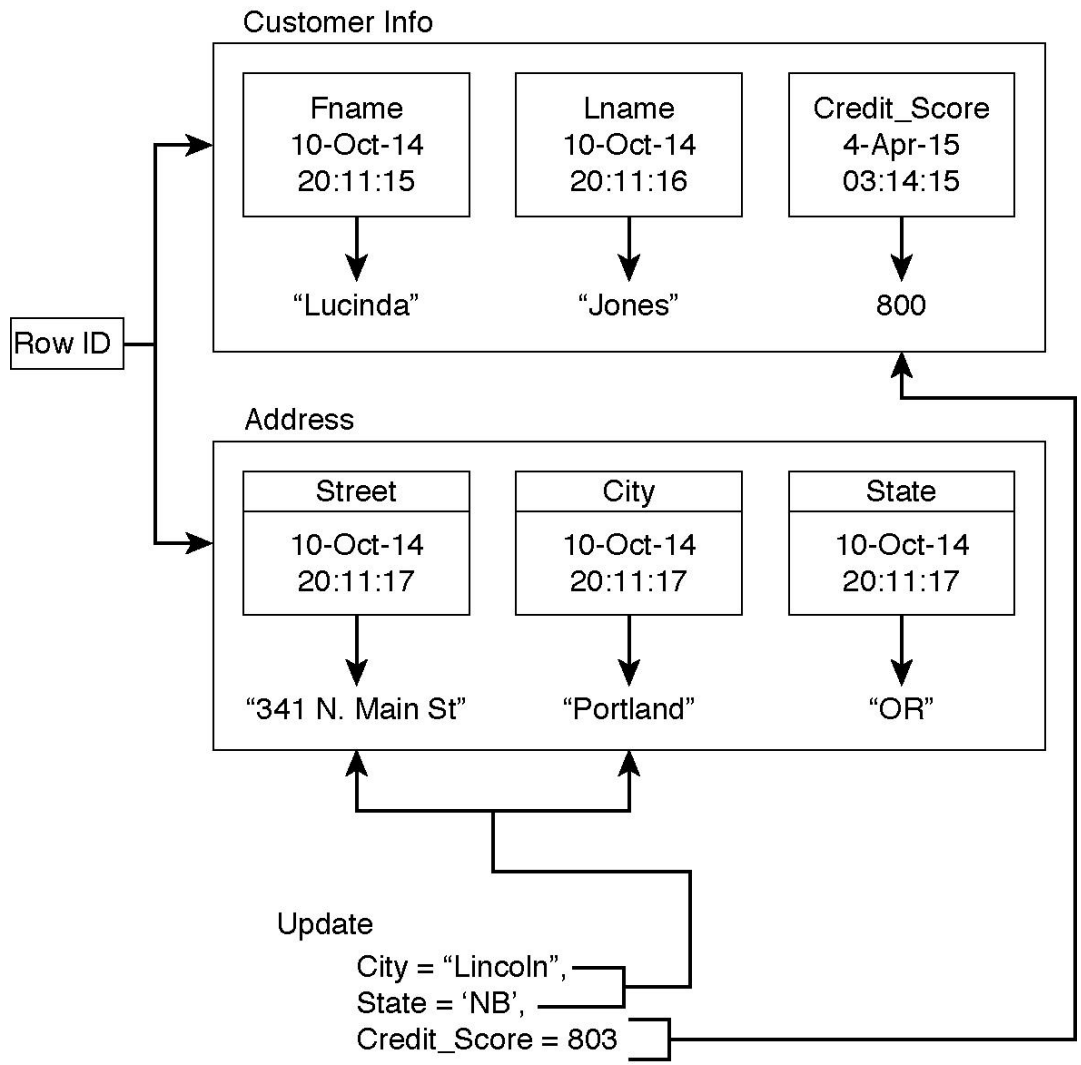

Figure 9.4 Read and write operations are atomic. All columns are read or written or none are.

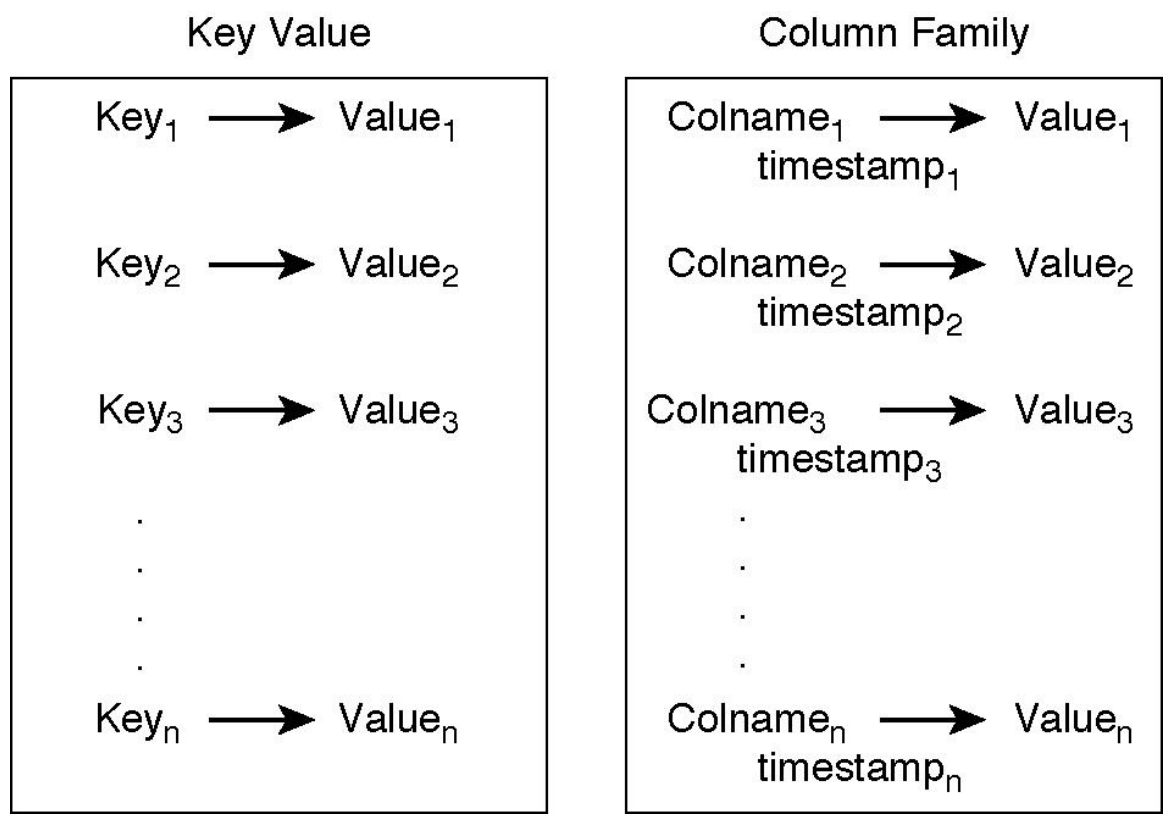

Figure 9.5 Keyspaces in key-value databases are analogous to column families in the way they maintain collections of attributes. Indexing, however, is different between the two database types.

Document Database:

```
{fname: 'Lucinda',
lname: 'Jones',
Credit_Rating: 800,
Address:
        {Street : '341 N. Main St.',
        City: 'Portland',
        State: 'OR'
        }
{fname: 'Frank',
lname: 'Antonio',
Credit_Rating: 768
}
```

Column Family Database:

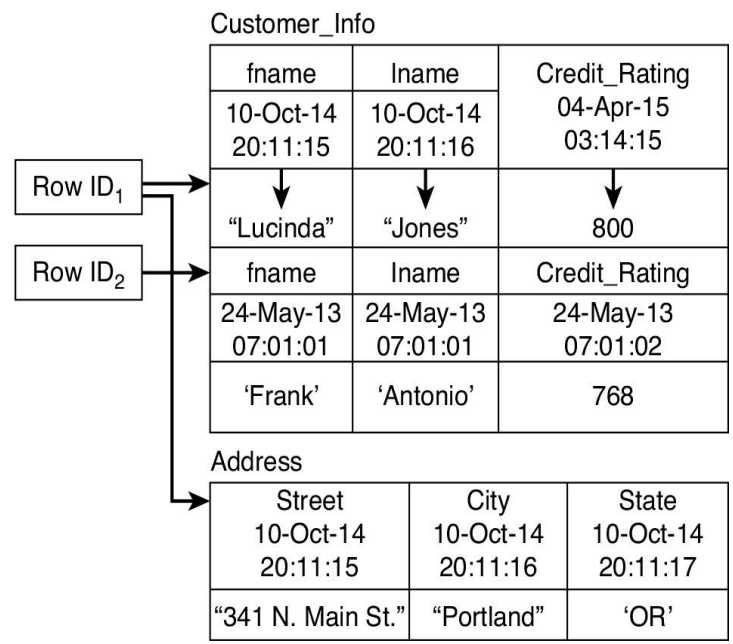

Figure 9.6 Column family databases, like document databases, may have values for some or all columns. Columns can be added programmatically as needed in both document and column family databases.

Row Key A	Column 1 Key	Column 2 Key	Column 3 Key	...	Column N Key
	Column 1 Value	Column 2 Value	Column 3 Value	...	Column N Value

Figure 9.7 Column family databases store data using maps of maps to column values.

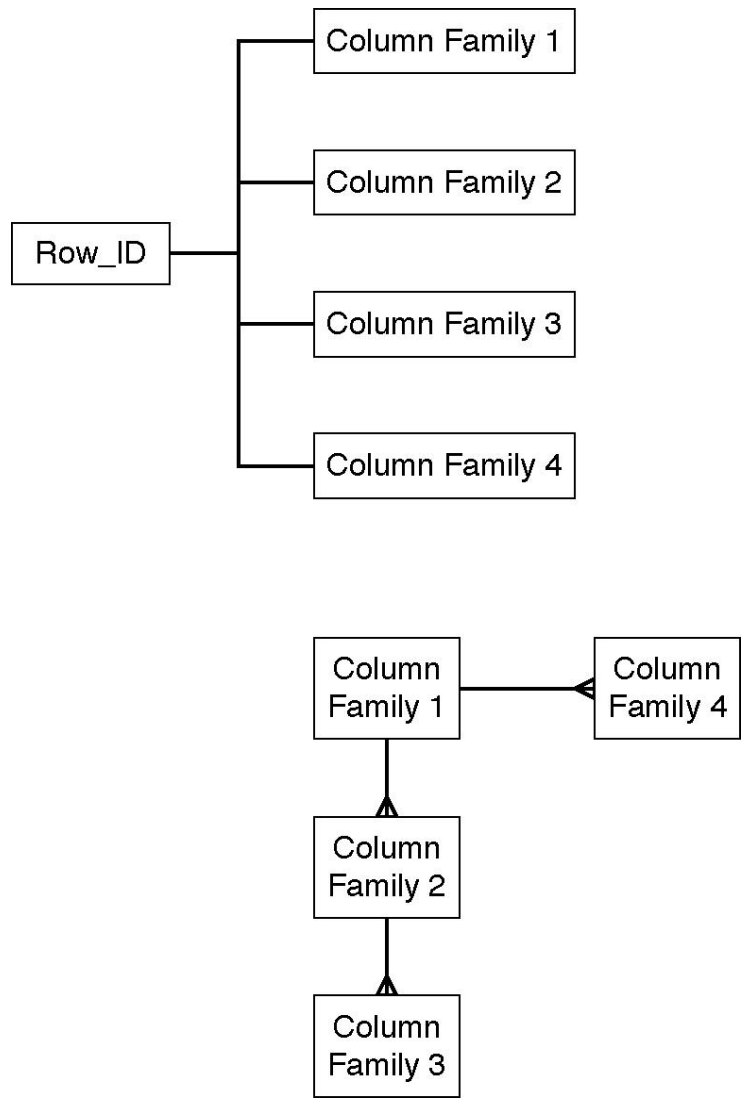

Figure 9.8 Instead of using joins and subqueries, as in a relational databases, column family databases use denormalization to maintain related information using a common row identifier.

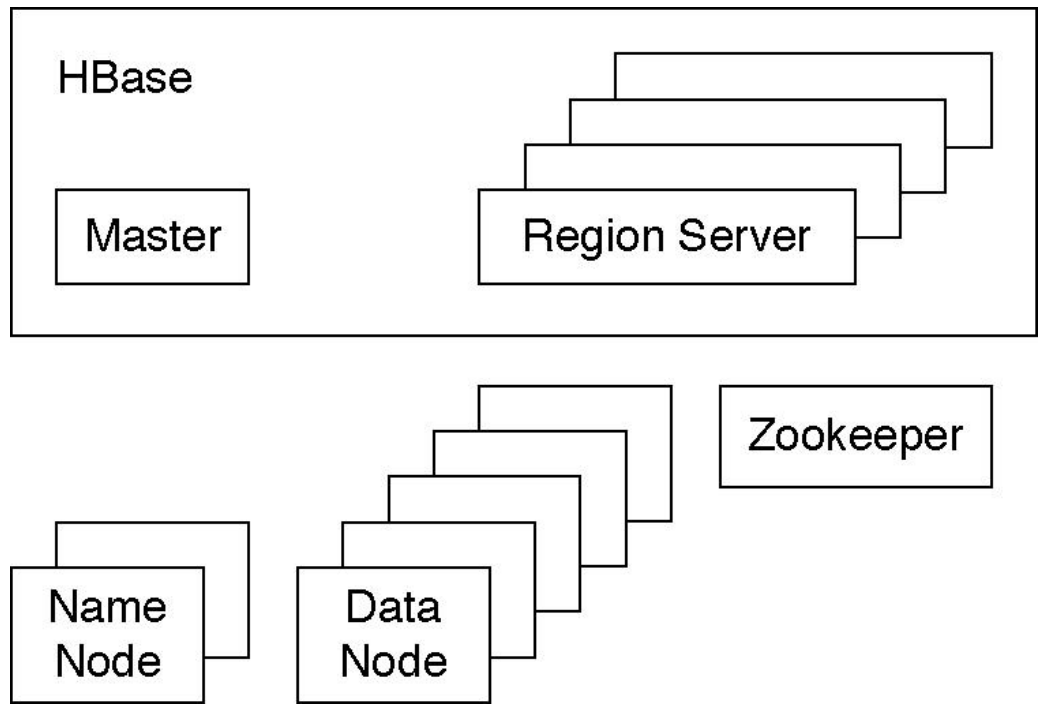

Figure 9.9 Apache HBase depends on multiple types of nodes that make up the Hadoop environment.

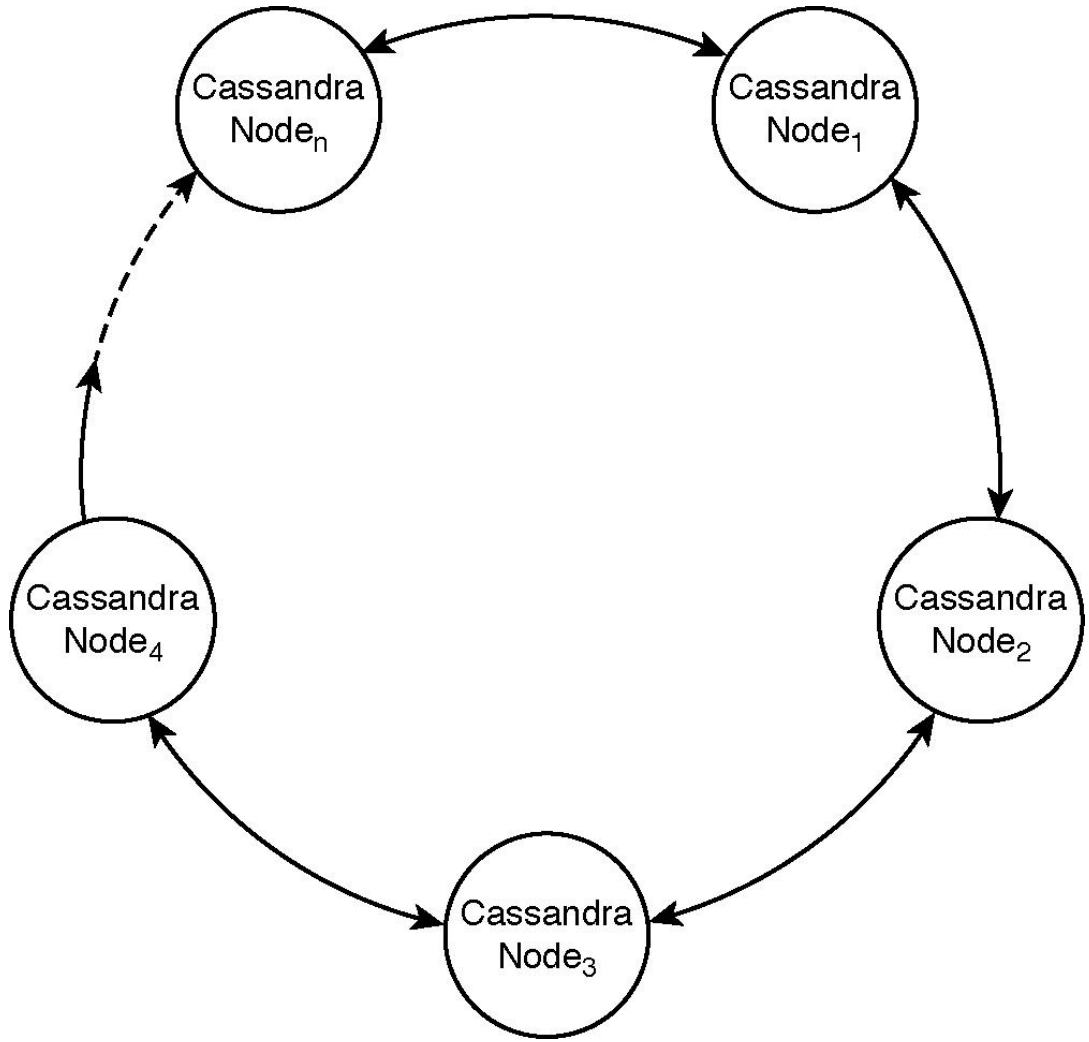

Figure 9.10 Cassandra uses a peer-to-peer architecture in which all nodes are the same.

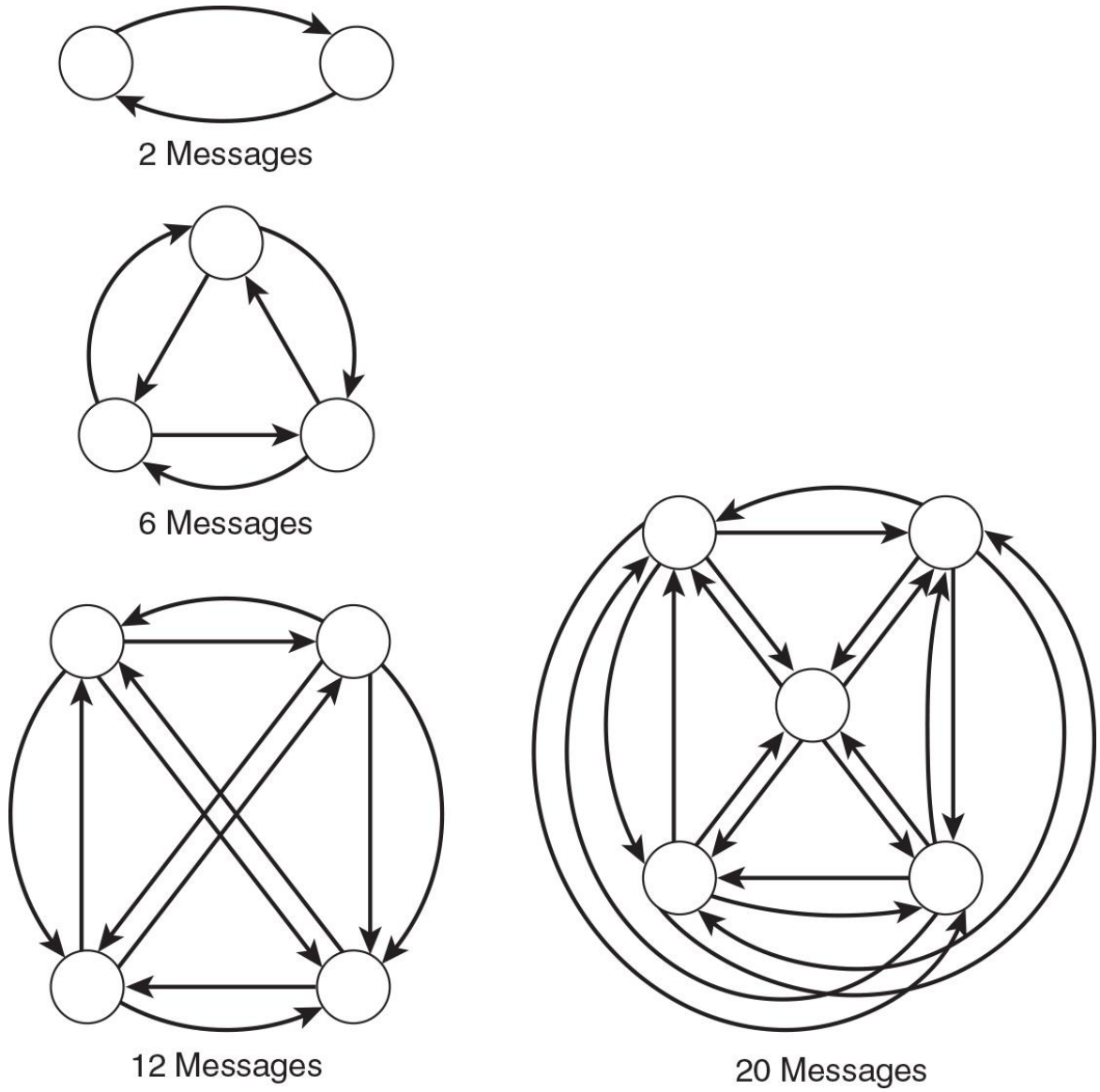

Figure 9.11 The number of messages sent in a complete server-to-server communication protocol grows more rapidly each time a server is added to the cluster.

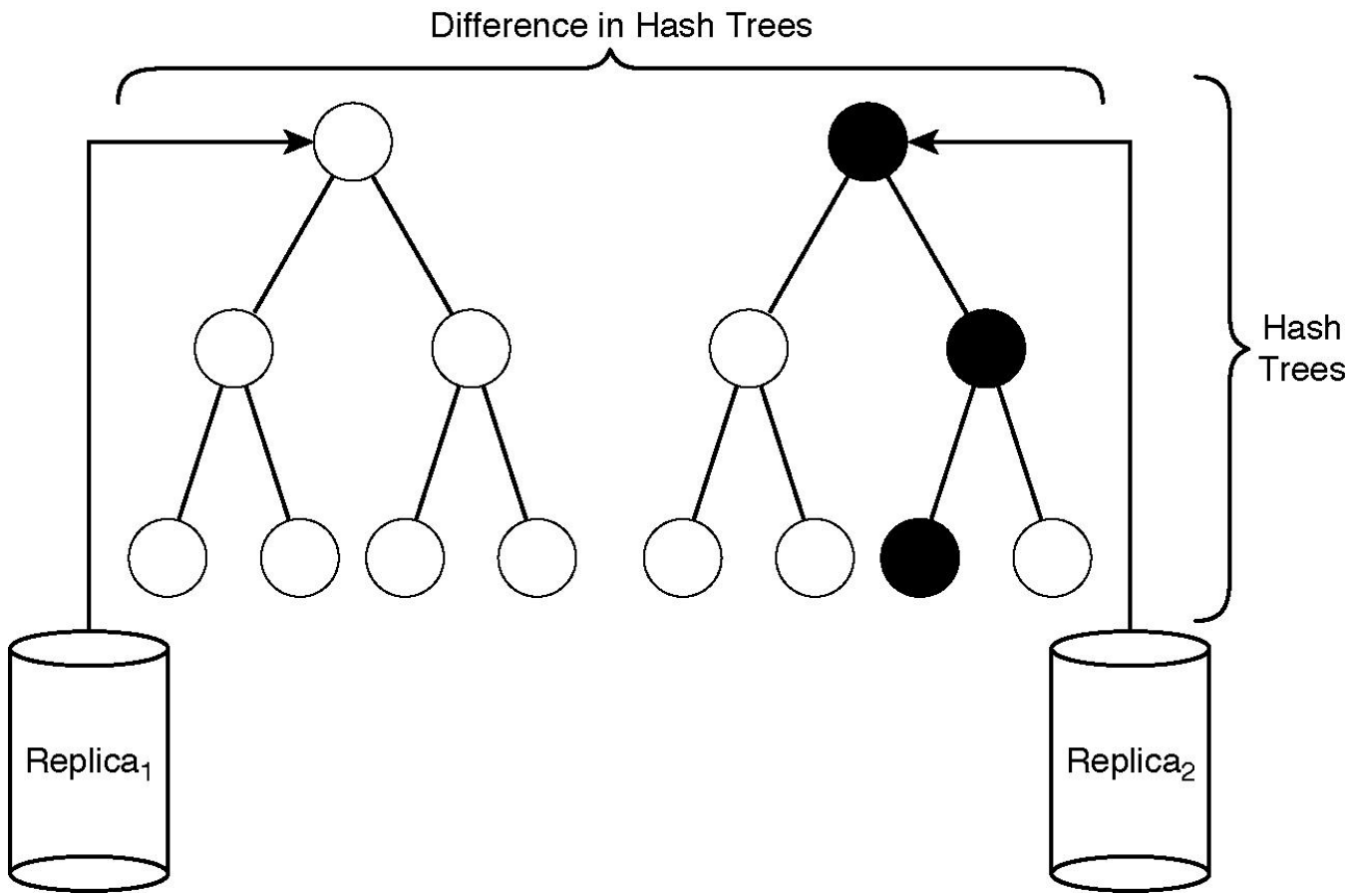

Figure 9.12 Cassandra regularly compares replicas of data to ensure they are up to date. Hashes are used to make this a relatively fast operation.

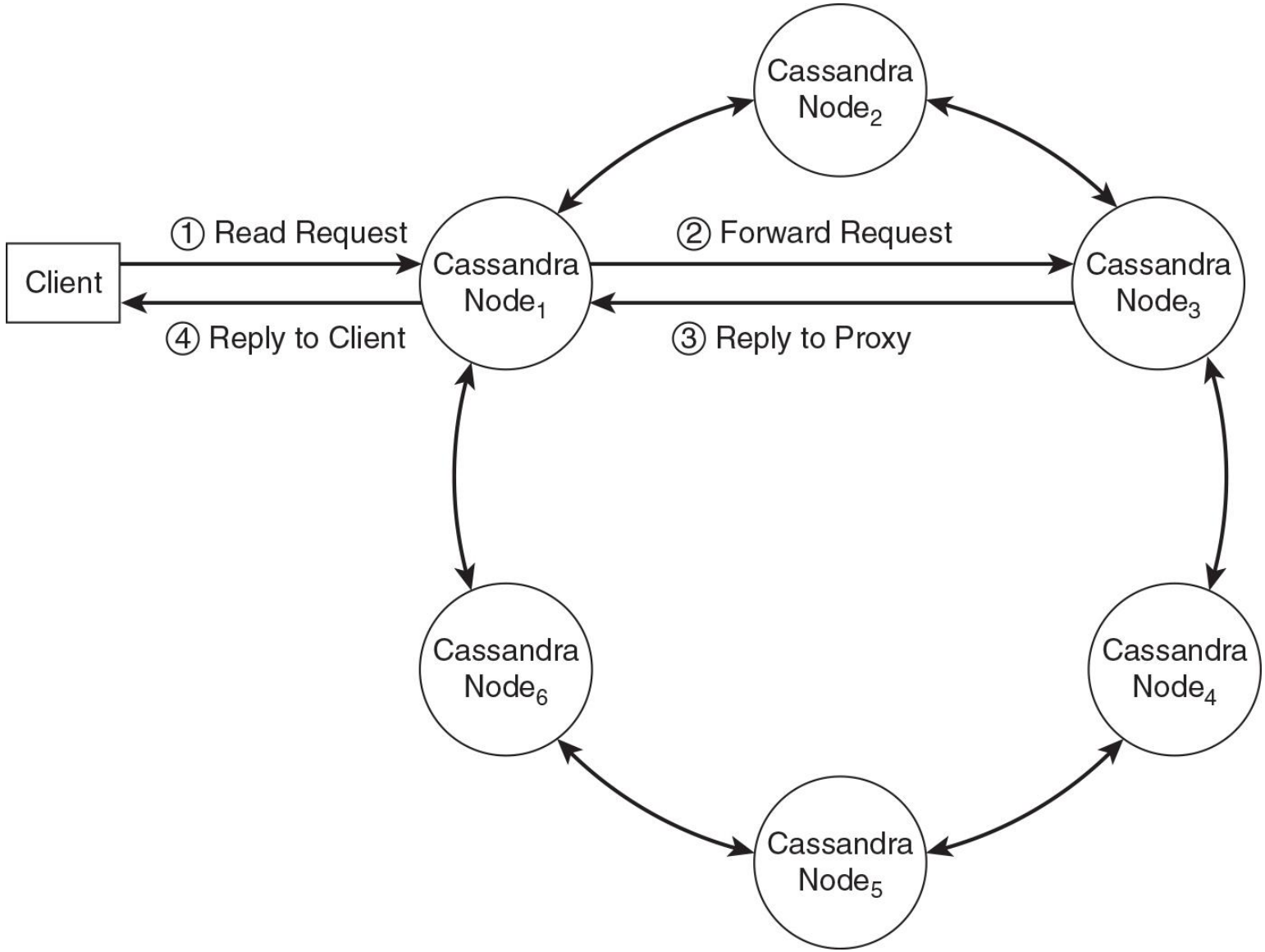

Figure 9.13 Any node in a Cassandra cluster can handle a client request. All nodes have information about the state of the cluster and can act as a proxy for a client, forwarding the request to the appropriate node in the cluster.

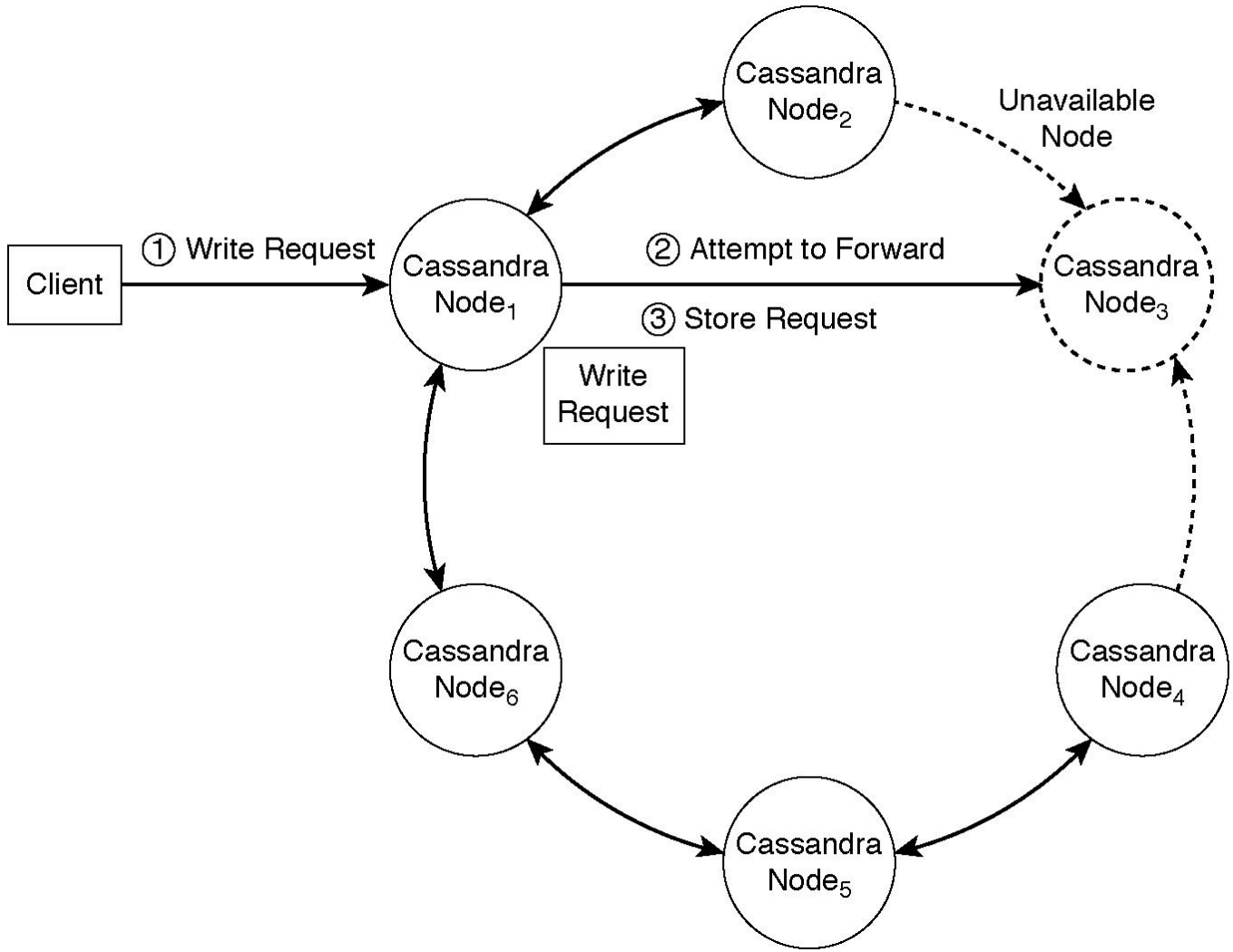

Figure 9.14 If a node is unavailable, then other nodes can receive write requests on its behalf and forward them to the intended node when it becomes available.

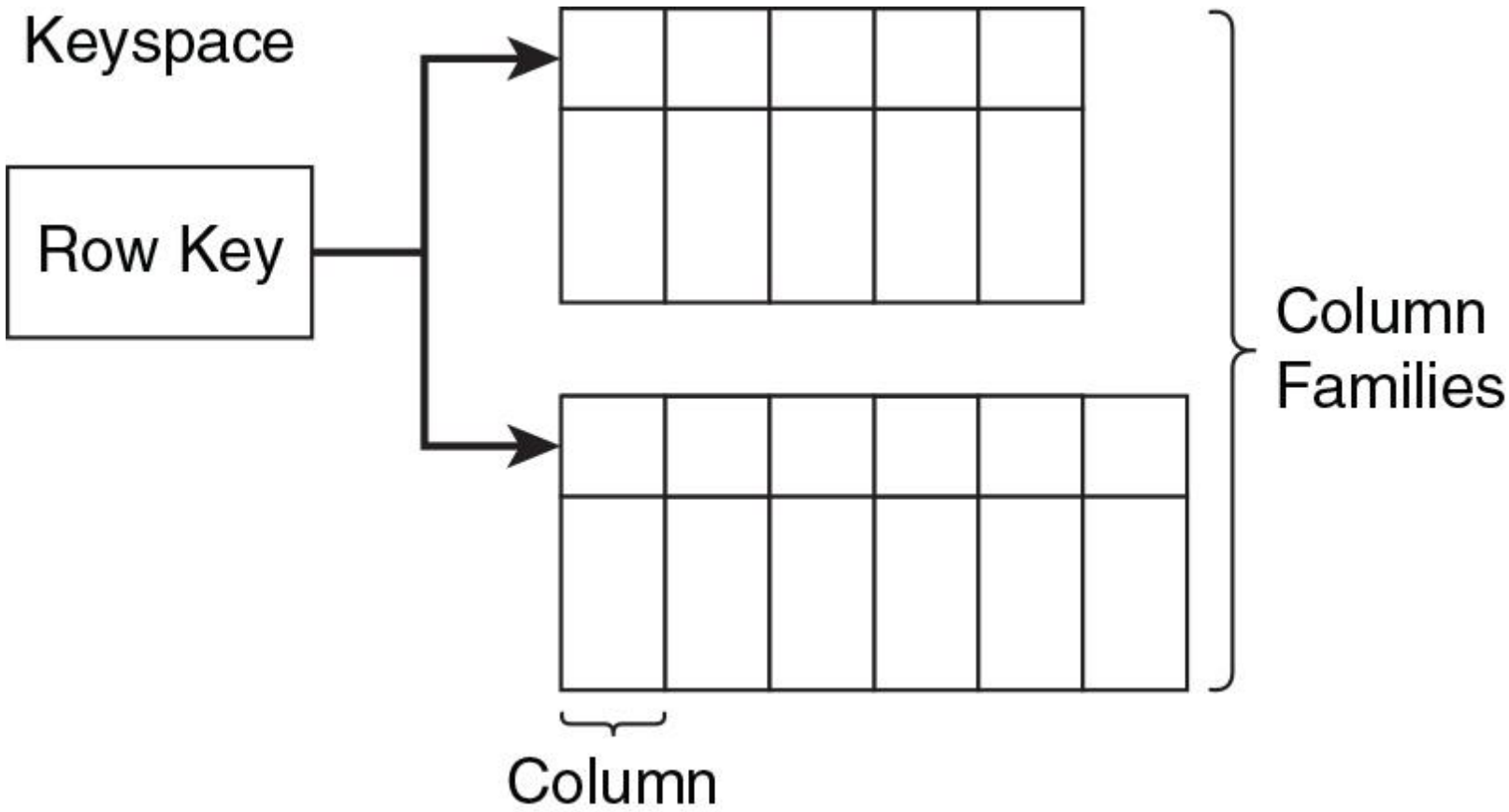

Figure 10.1 A keyspace is a top-level container that logically holds column families, row keys, and related data structures. Typically, there is one keyspace per application.

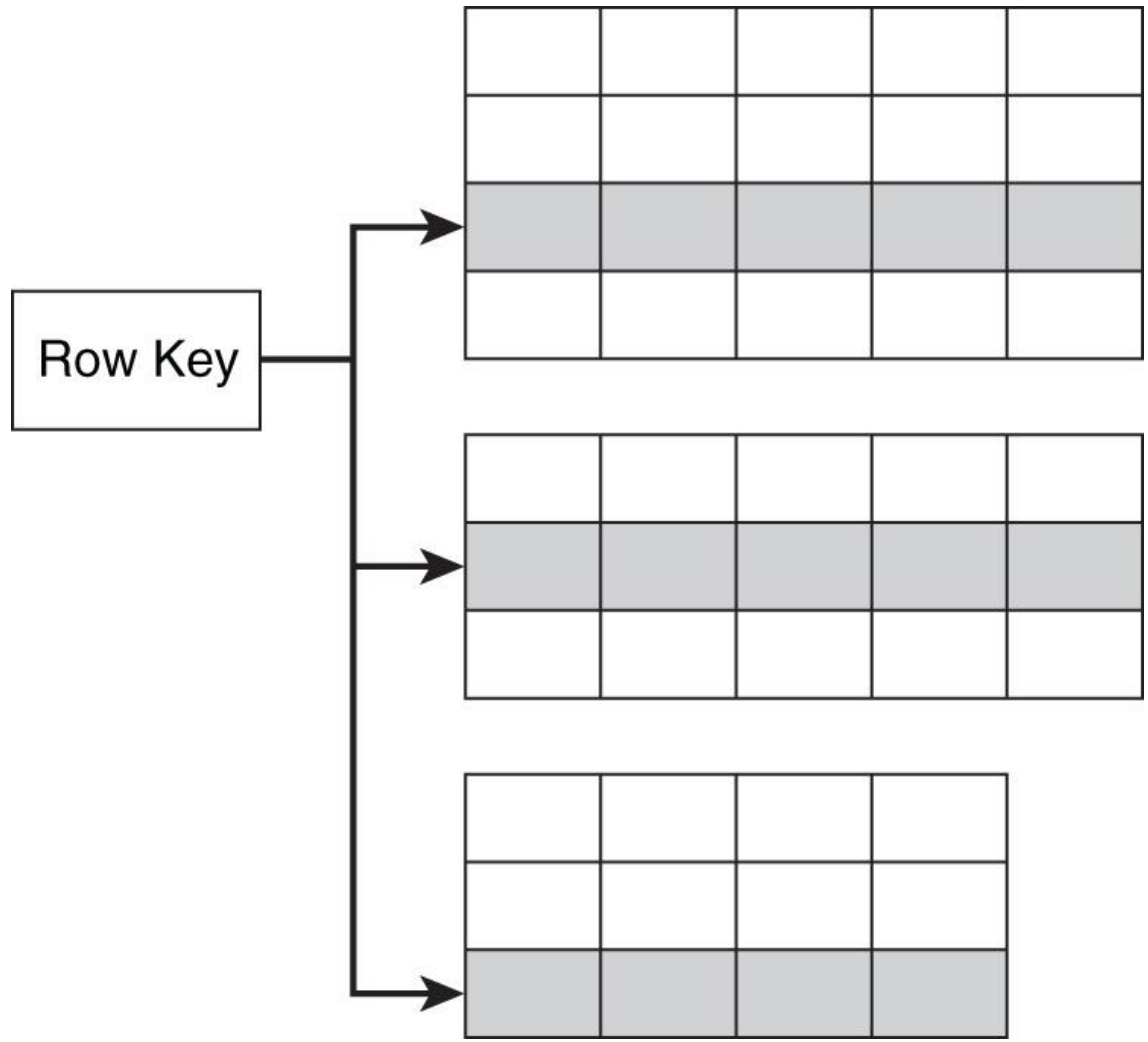

Figure 10.2 A row key uniquely identifies a row and has a role in determining the order in which data is stored.

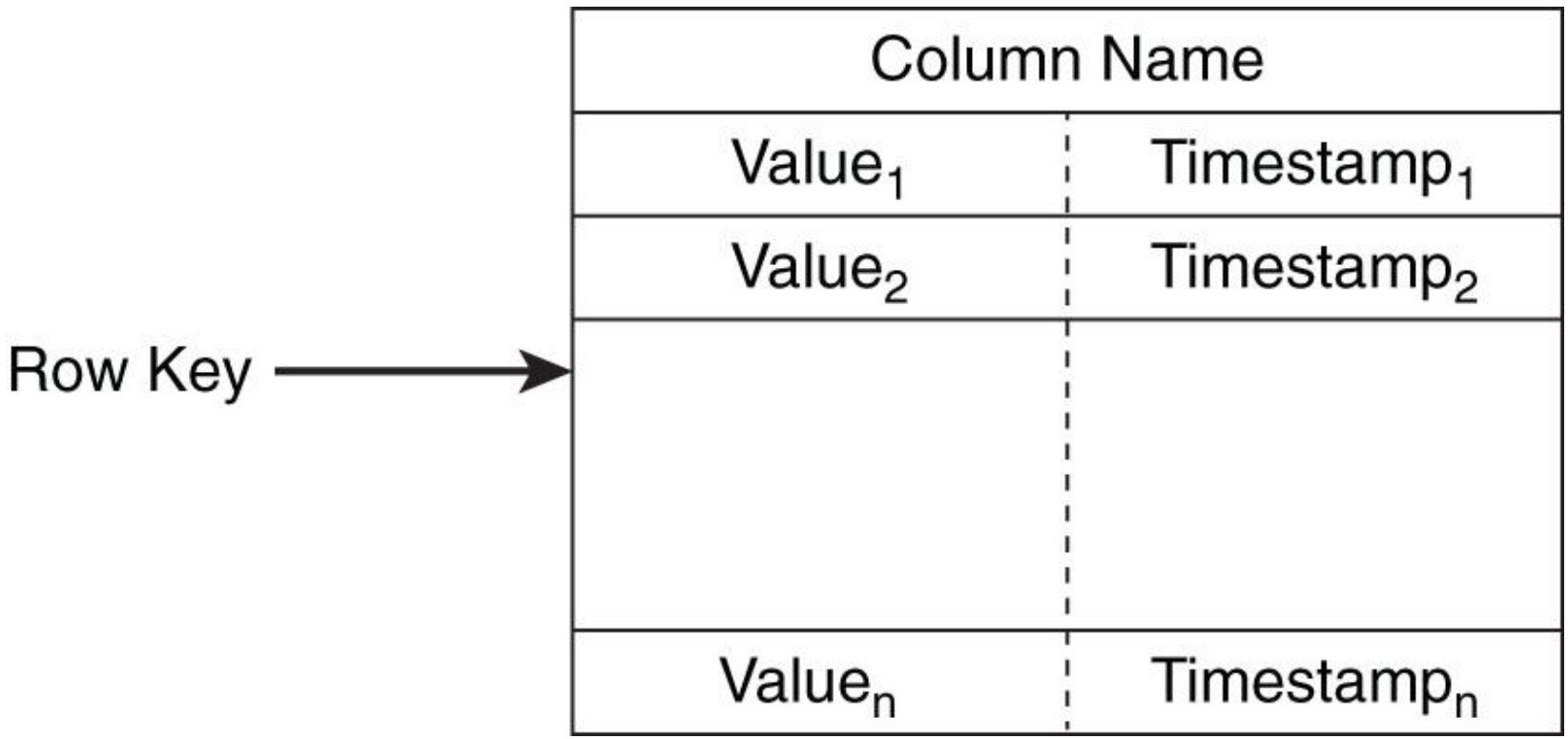

Figure 10.3 A column, along with a row key and version stamp, uniquely identifies values.

Street	City	State	Province	Zip	Postal Code	Country
178 Main St.	Boise	ID		83701		U.S.
89 Woodridge	Baltimore	MD		21218		U.S.
293 Archer St.	Ottawa		ON		K1A 2C5	Canada
8713 Alberta DR	Vancouver		BC		VSK 0AI	Canada

Figure 10.4 Column families are analogous to relational database tables: They store multiple rows and multiple columns. There are, however, significant differences between the two, including varying columns by row.

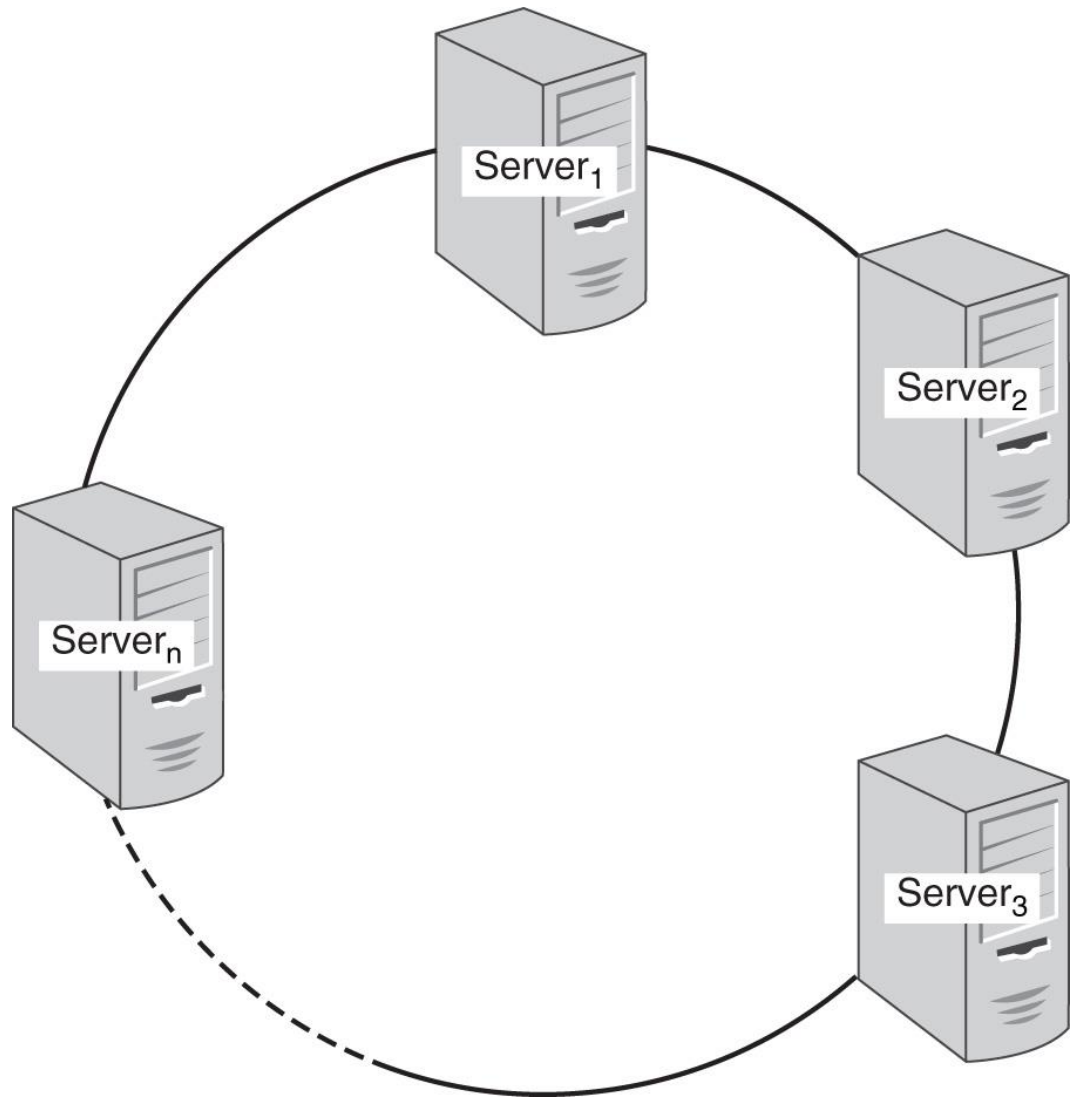

Figure 10.5 Clusters are collections of servers functioning together to implement a distributed service, such as a column family database.

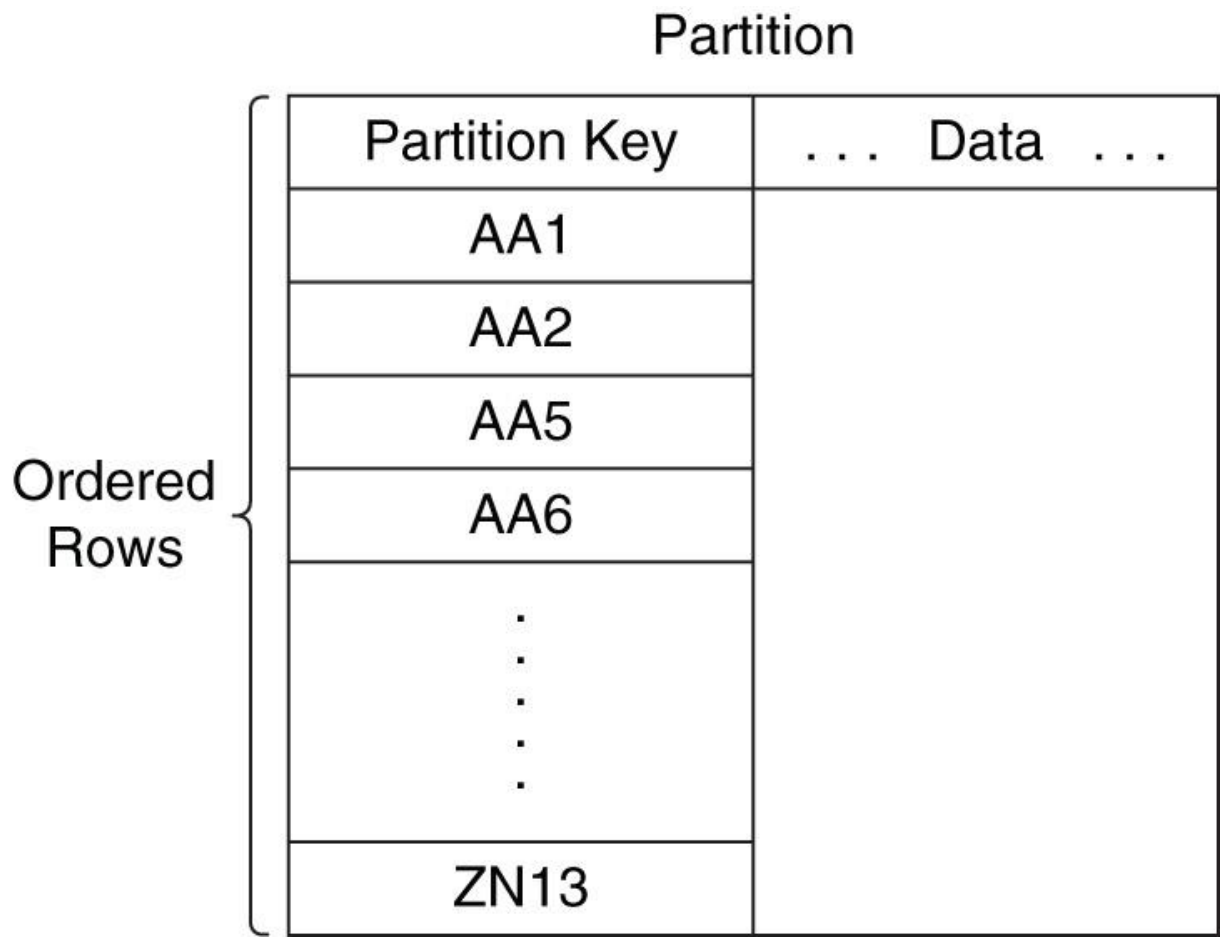

Figure 10.6 Partitions store data ordered by partition key.

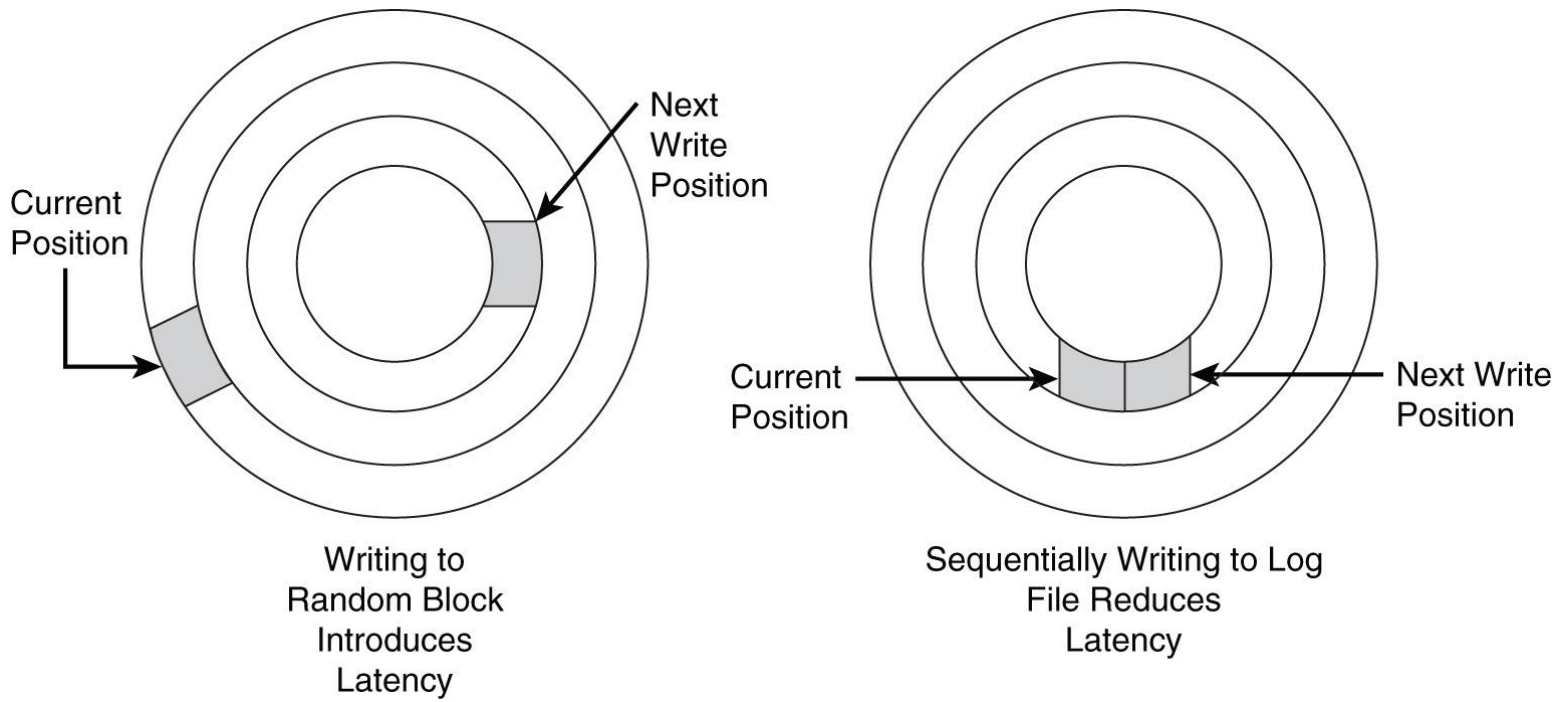

Figure 10.7 A commit log saves data written to the database prior to writing it to partitions. This reduces the latency introduced by random writes on disks.

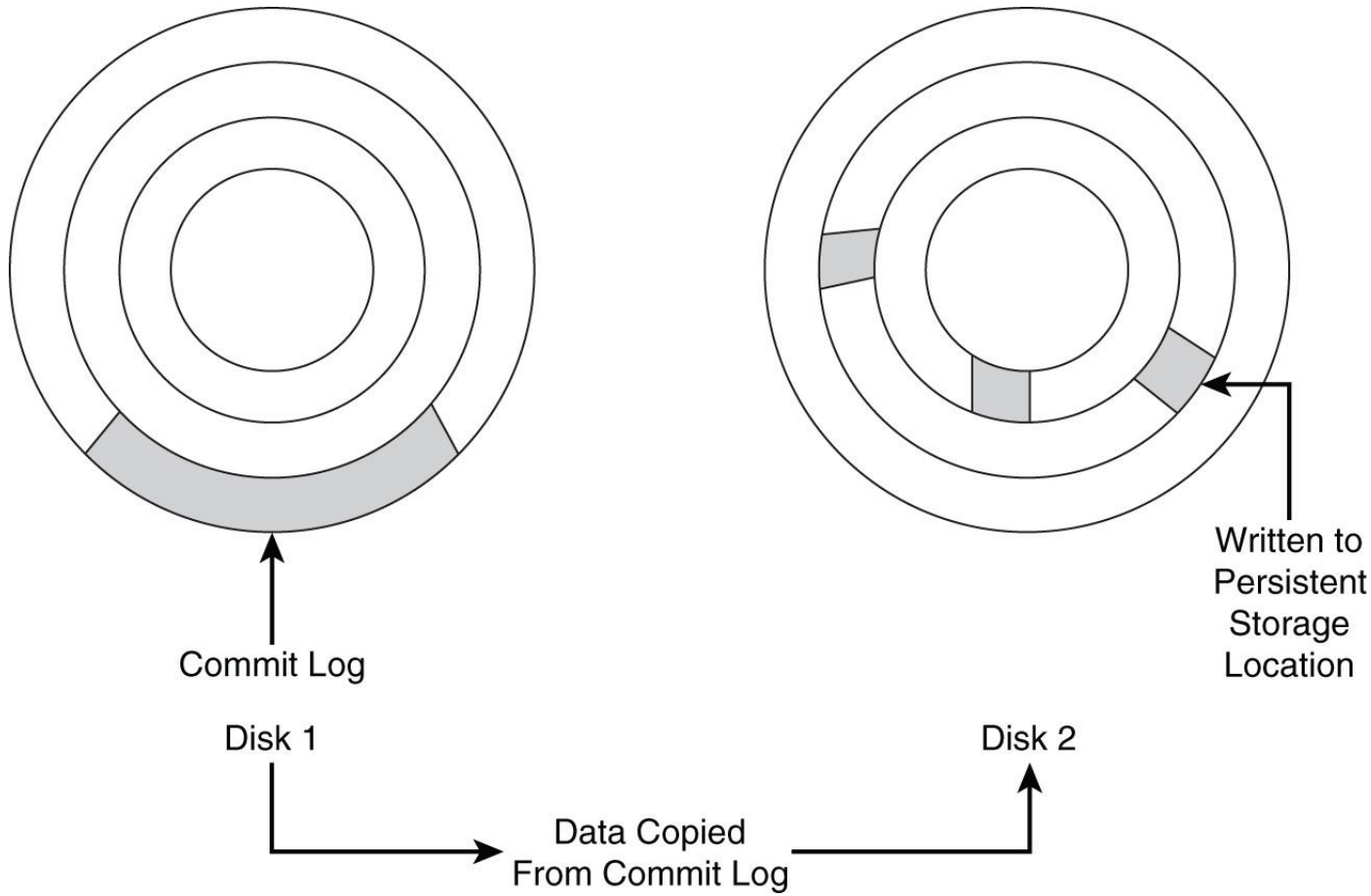

Figure 10.8 After a database failure, the recovery process reads the commit log and writes entries to partitions. The database remains unavailable to users until all commit log entries are written to partitions.

Member Function

Input Set	Test Element	In Set
{a,b,c}	a	Yes
{a,b,c}	c	Yes
{a,b,c}	e	No

Bloom Filter

Input Set	Test Element	In Set
{a,b,c}	a	Yes
{a,b,c}	c	Yes
{a,b,c}	e	Yes
{a,b,c}	f	No
{a,b,c}	g	No

Low Probability but Possible

Figure 10.9 Member functions always return accurate results. Bloom filters usually return accurate results but sometimes make false-positive errors.

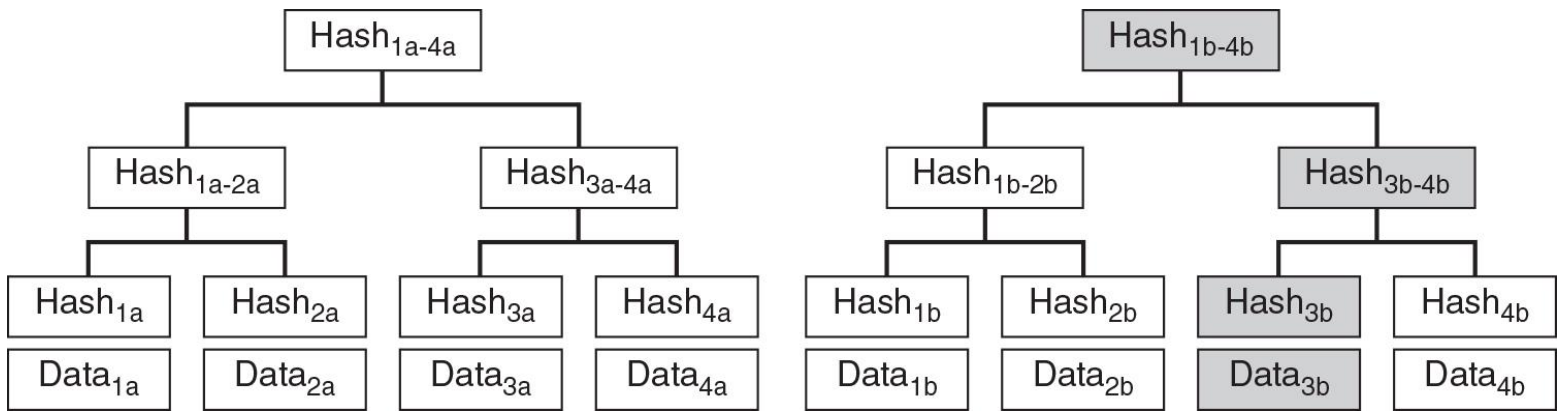

Figure 10.10 Hash trees, or Merkle trees, allow for rapid checks on consistency between two data sets. In this example, data3a and data3b are different, resulting in different hash values in each level from the data block to the root.

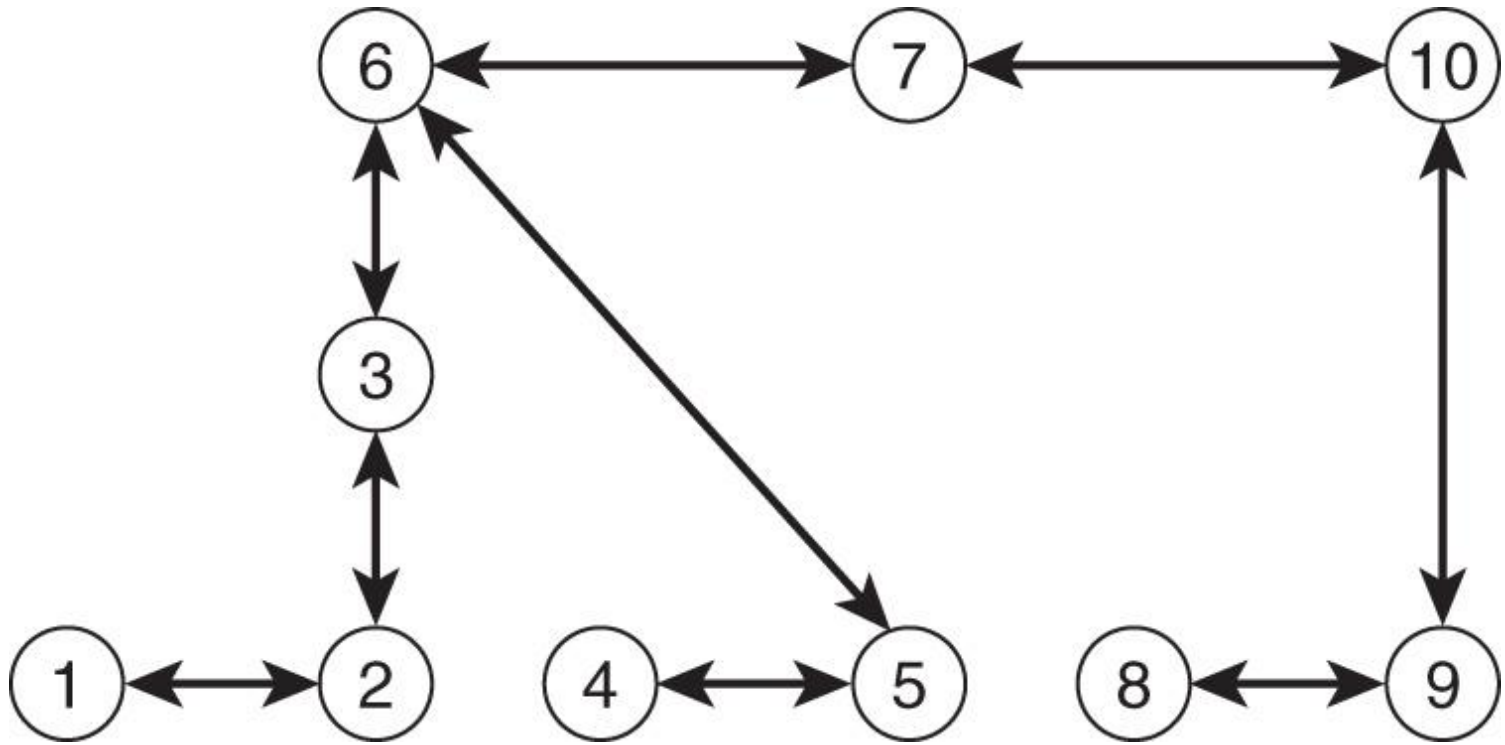

Figure 10.11 Gossip protocols spread information with fewer messages than a protocol that requires all nodes to communicate with all other nodes.

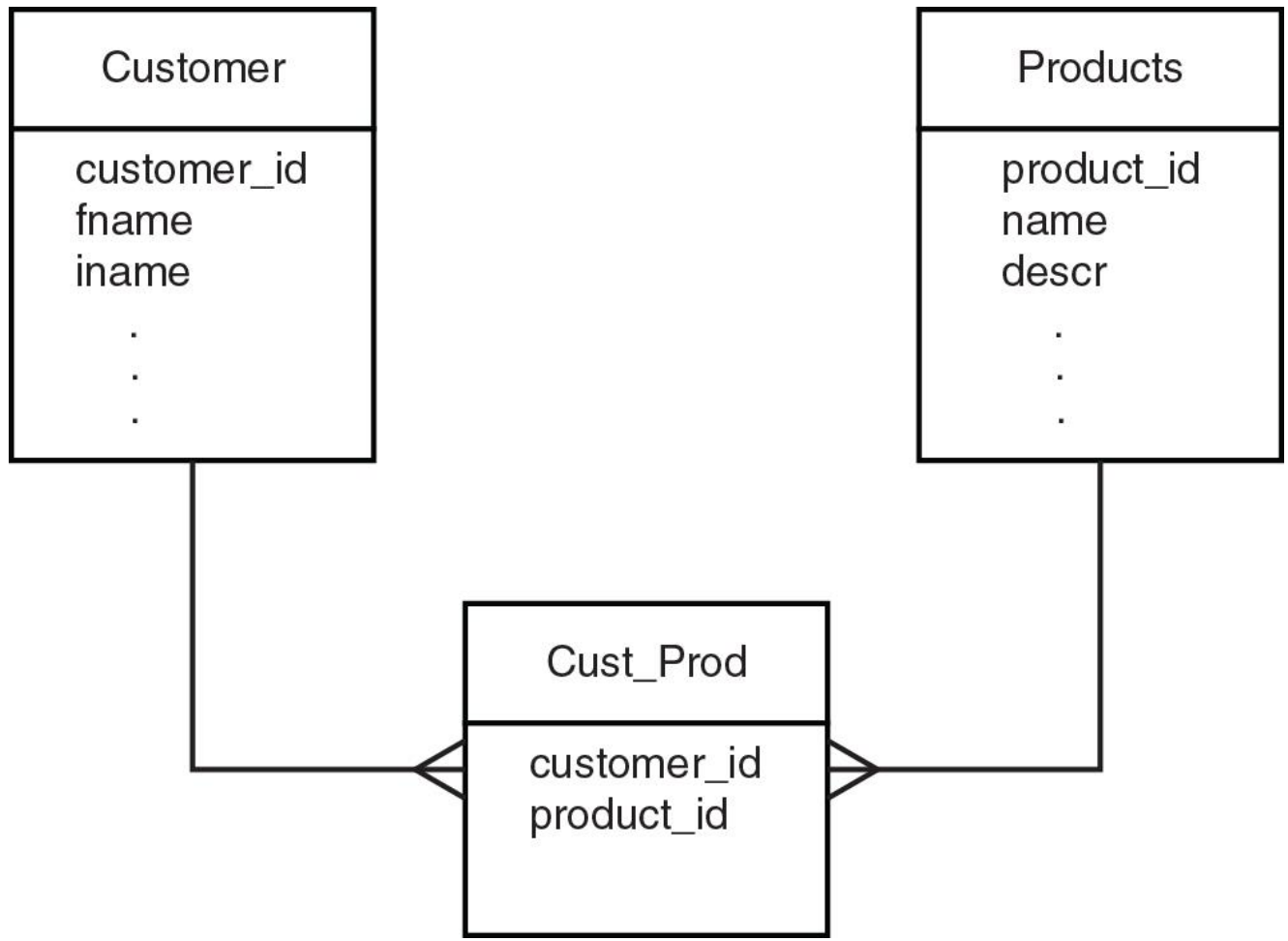

Figure 11.1 In relational databases, many-to-many relations are modeled with a table for storing primary keys of the two entities in a many-to-many relationship.

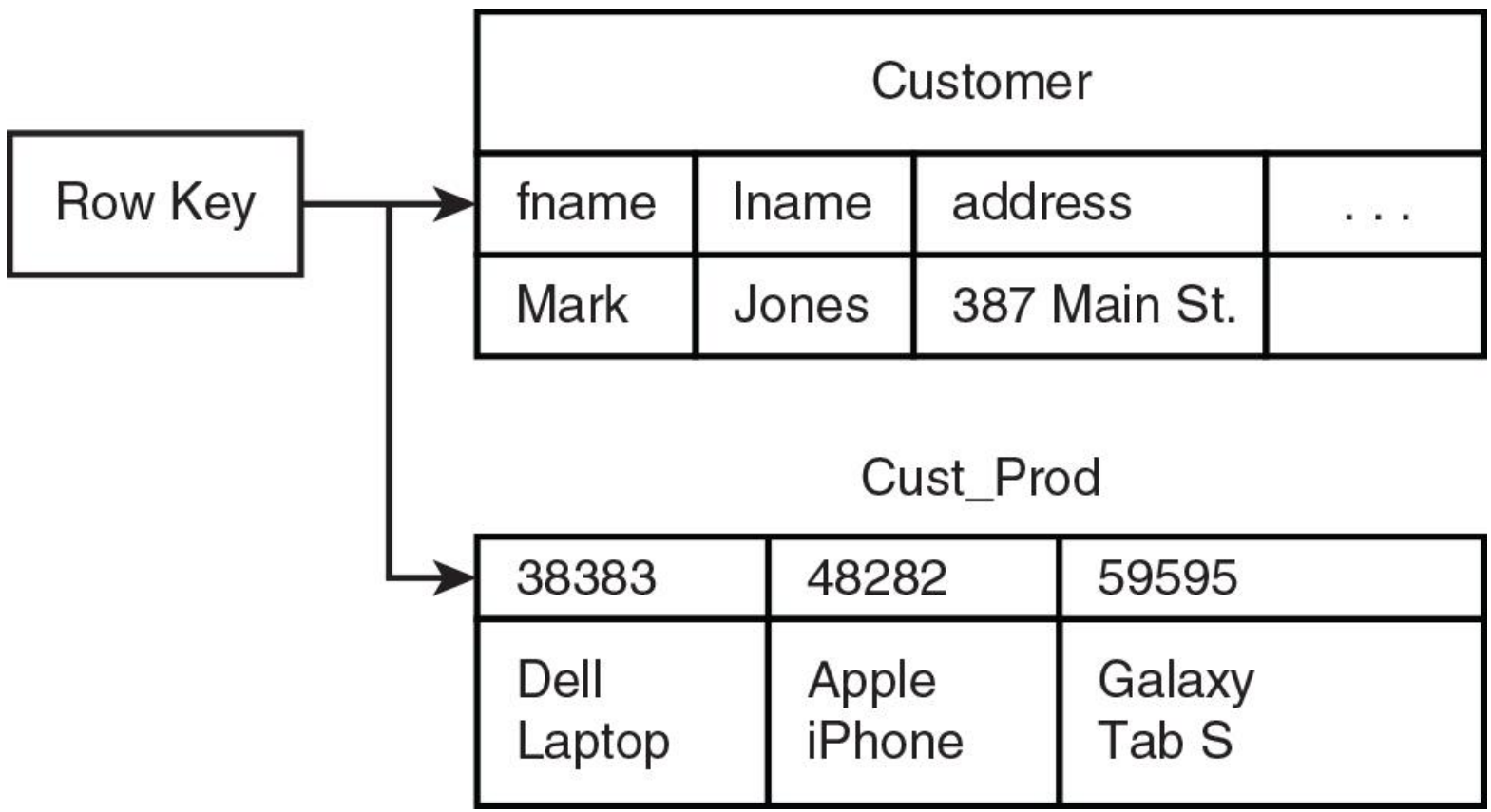

Figure 11.2 In a column family database, many-to-many relationships are captured by denormalizing data.

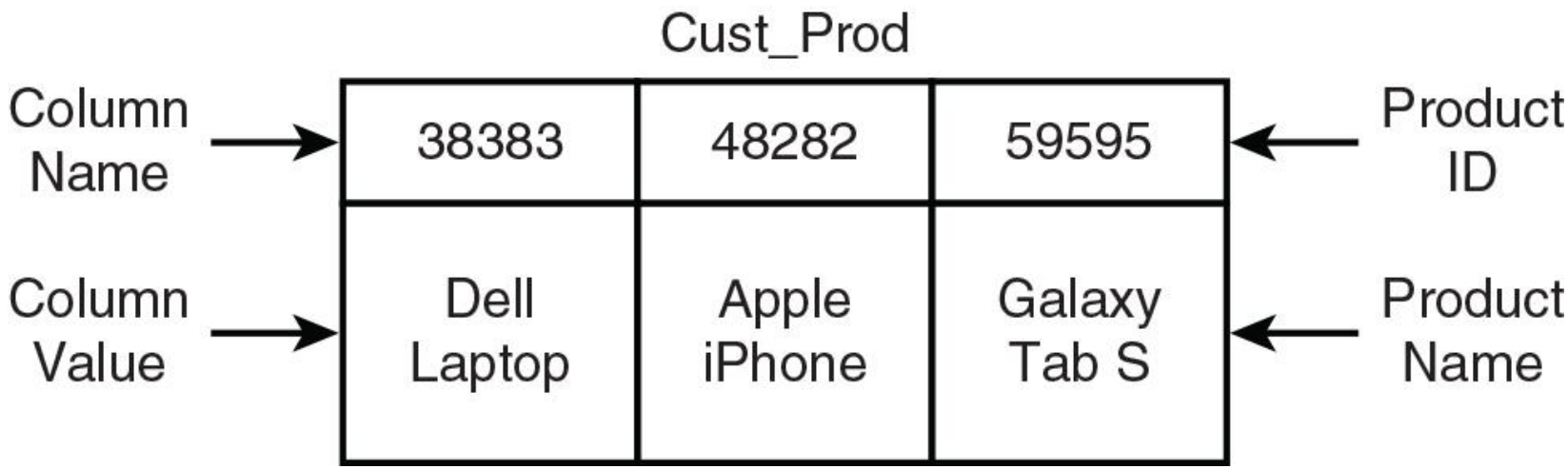

Figure 11.3 Both the column name and the column value can store data.

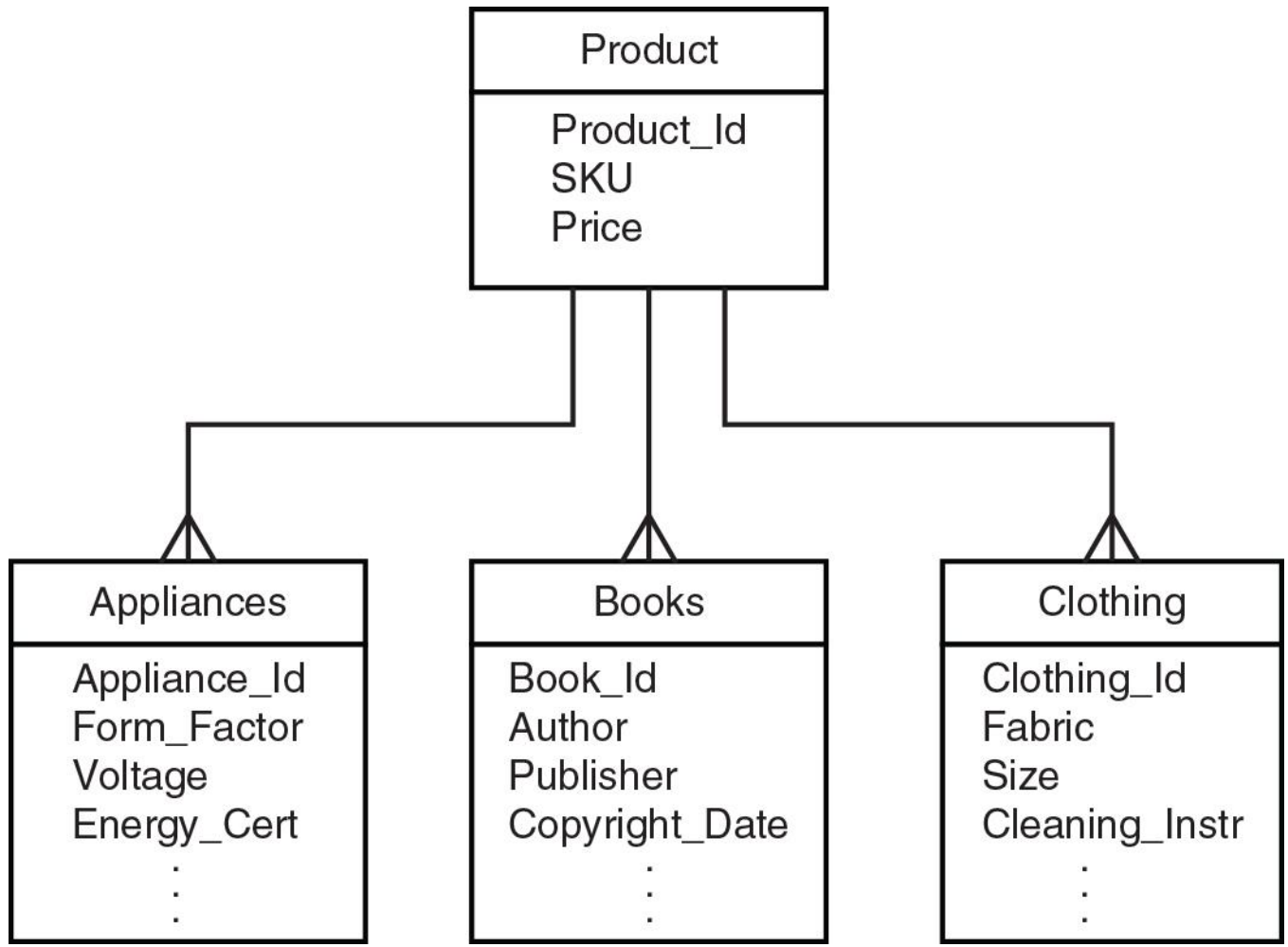

Figure 11.4 Entities can have attributes stored in multiple tables, but this is not recommended for column family databases.

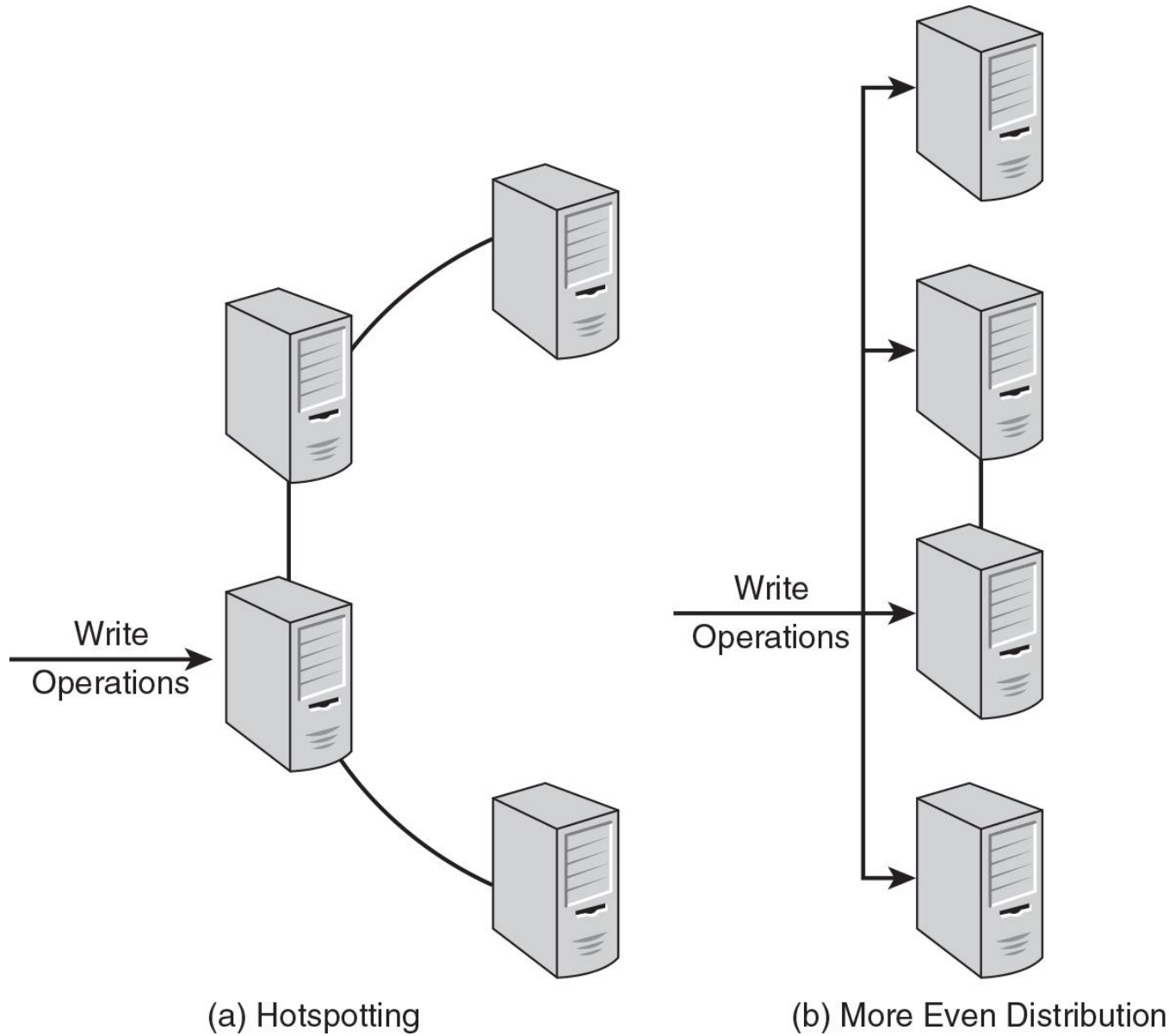

Figure 11.5 (a) Hotspotting leads to underutilization of cluster resources while (b) more even distribution of operations leads to more efficient use of resources.

Column Family		
Column $Name_1$	Column $Name_2$	Column $Name_3$
$value_{1a}$: $timestamp_{1a}$	$value_{2a}$: $timestamp_{2a}$	$value_{3a}$: $timestamp_{3a}$
$value_{1b}$: $timestamp_{1b}$	$value_{2b}$: $timestamp_{2b}$	$value_{3b}$: $timestamp_{3b}$
$value_{1c}$: $timestamp_{1c}$	$value_{2c}$: $timestamp_{2c}$	$value_{3c}$: $timestamp_{3c}$

Figure 11.6 HBase provides for column value versions. The number of versions maintained is controlled by database parameters, which can be changed according to application requirements.

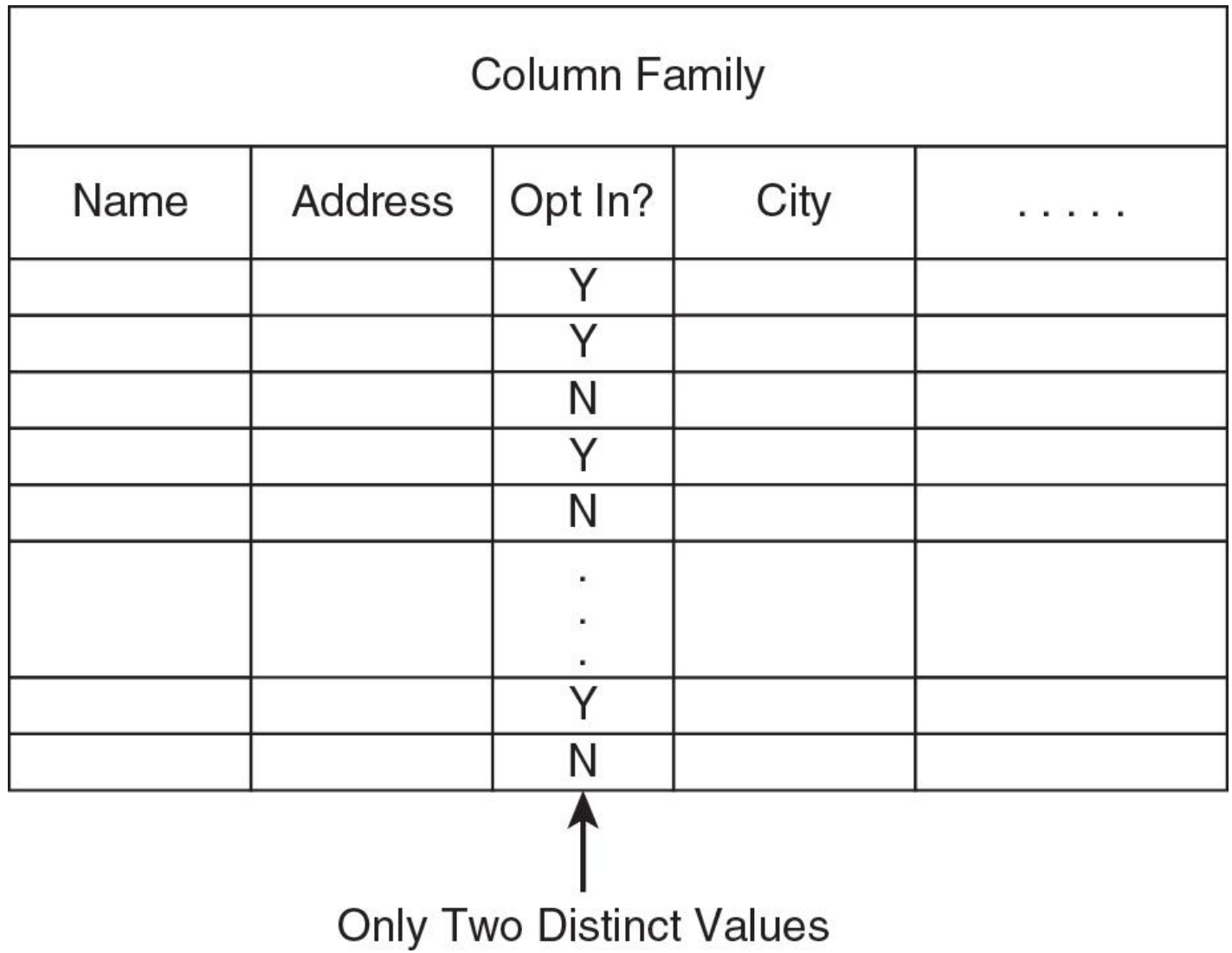

Figure 11.7 Columns with few distinct values are not good candidates for secondary indexes.

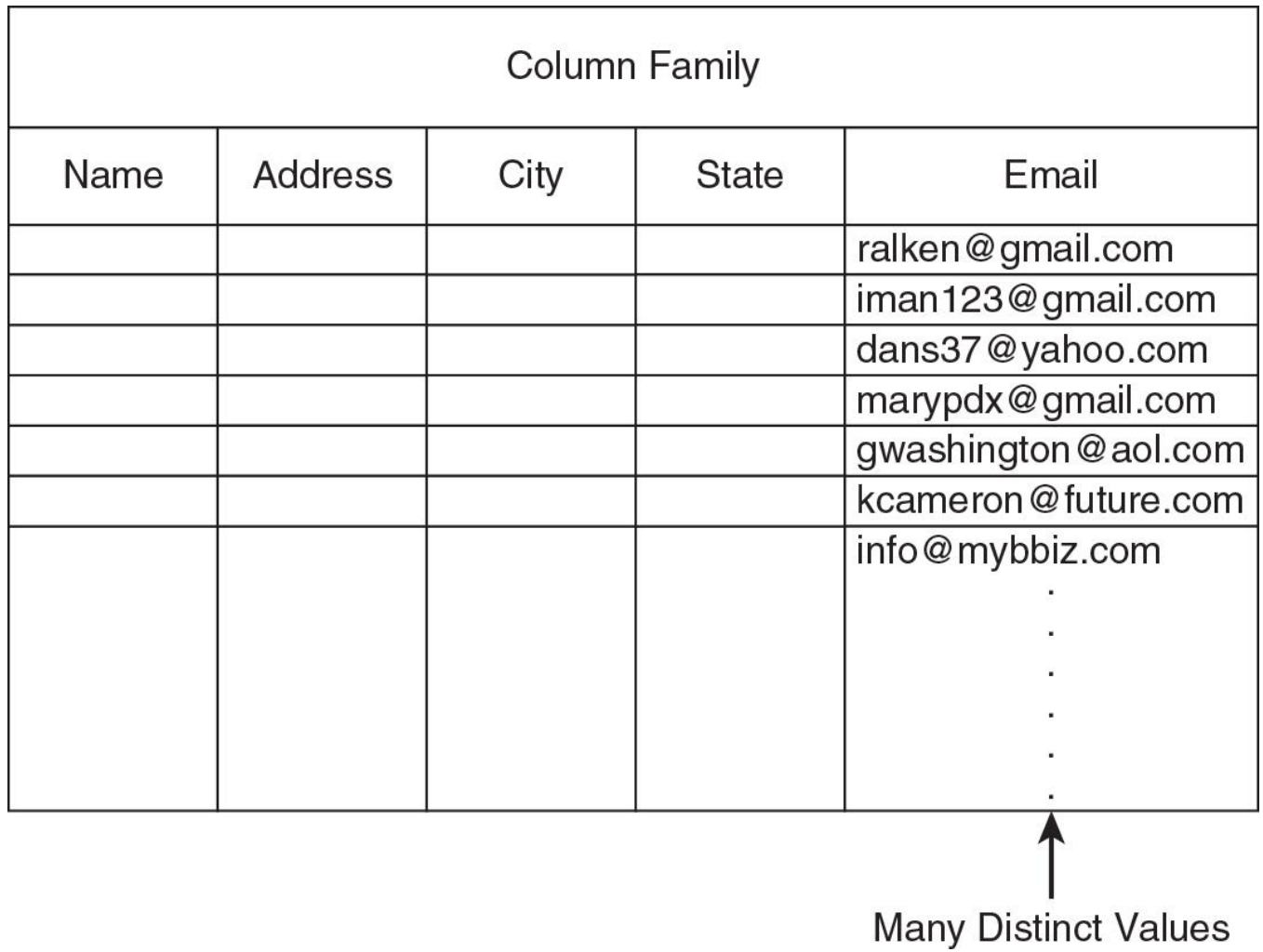

Figure 11.8 Rows with too many distinct values are also not good candidates for indexes.

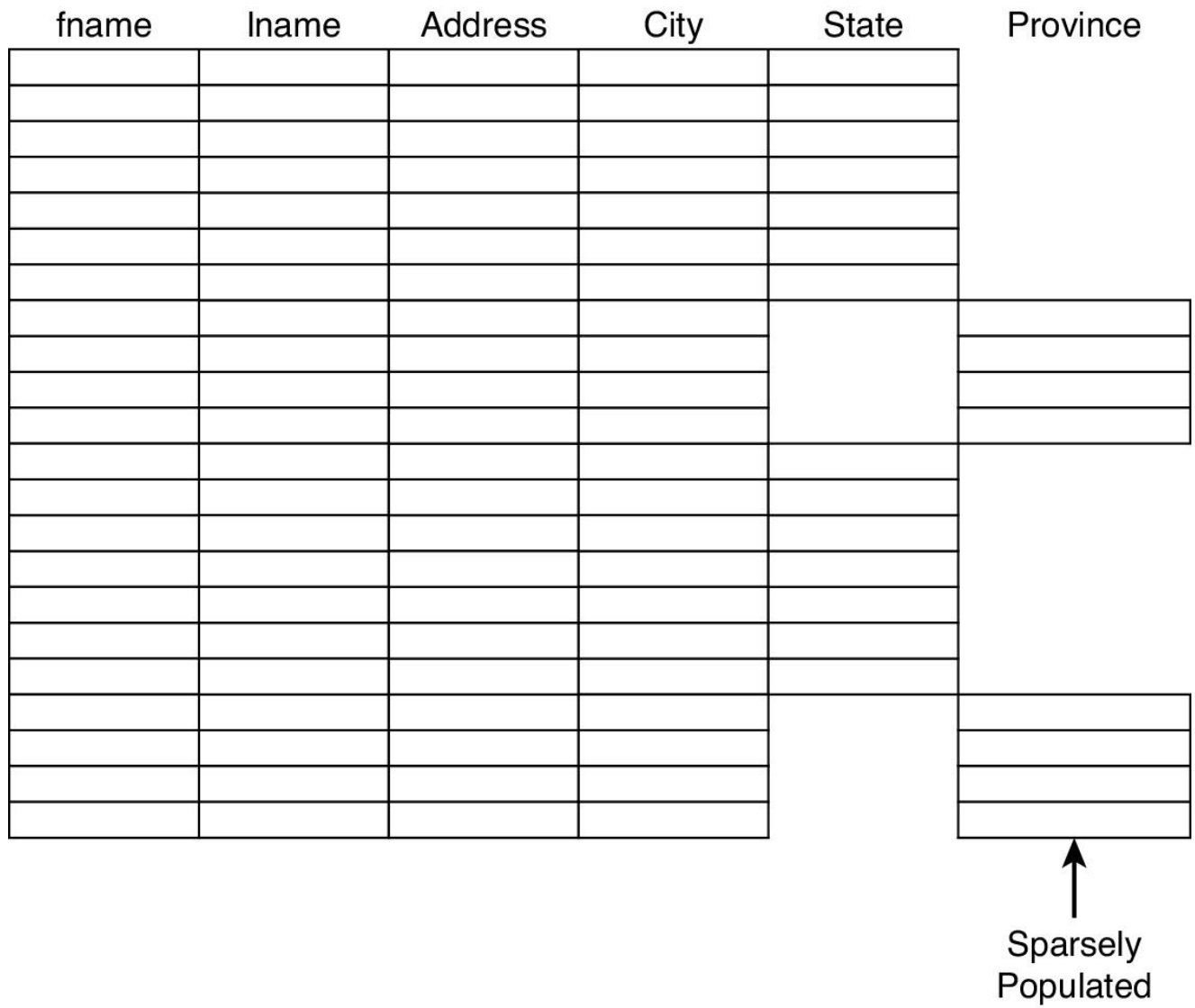

Figure 11.9 Sparsely populated columns should not be indexed.

Customer

Row key	fname	lname	street	city	state
123	Jane	Smith	387 Main St	Boise	ID
287	Mark	Jones	192 Wellfleet Dr	Austin	TX
1987	Harsha	Badal	298 Commercial St	Provincetown	MA
2405	Senica	Washington	98 Morton Ave	Windsor	CT
3902	Marg	O'Malley	981 Circle Dr	Santa Fe	NM

Product

Row key	name	descr	qty_avail	category	
38383	Dell Latitude E6410	Laptop with ...	124	Computer	
48282	Apple iPhone	iPhone 6 with ...	345	Phone	
59595	Galaxy Tab S	Samsung tablet ...	743	Tablet	

Cust_by_Prod

Row key	123	287	1987	2405	3902
38383	Smith		Badal		
48282	Smith	Jones			O'Malley
59595				Washington	

Prod_by_Cust

Row key	38383	48282	59595		
123	Dell Latitude E6410	Apple iPhone			
287		Apple iPhone			
1987	Dell Latitude E6410				
2405			Galaxy Tab S		
3902		Apple iPhone			

Figure 11.10 Example of tables as indexes method.

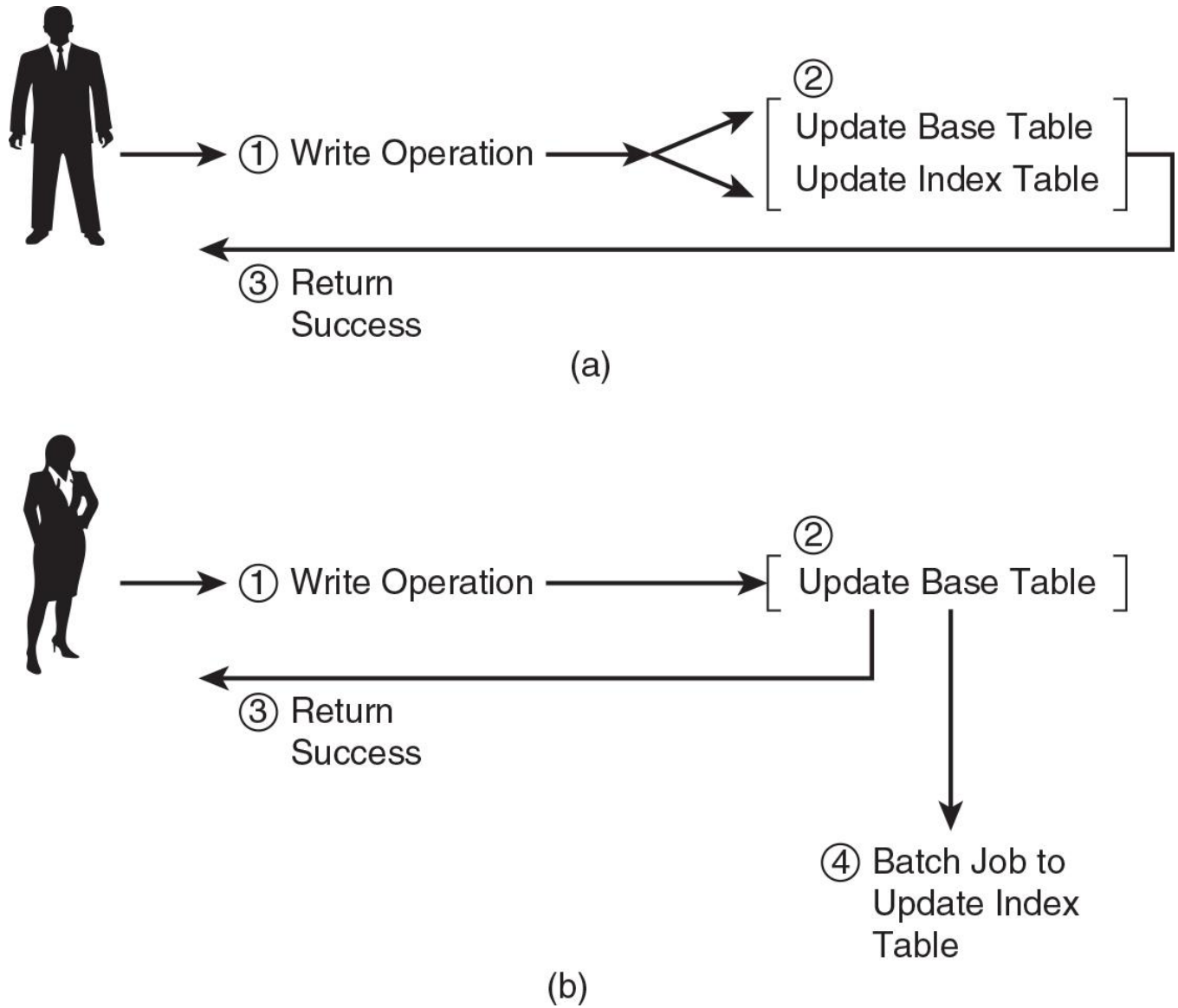

Figure 11.11 (a) Updating an index table during write operations keeps data synchronized but increases the time needed to complete a write operation. (b) Batch updates introduce periods of time when the data is not synchronized, but this may be acceptable in some cases.

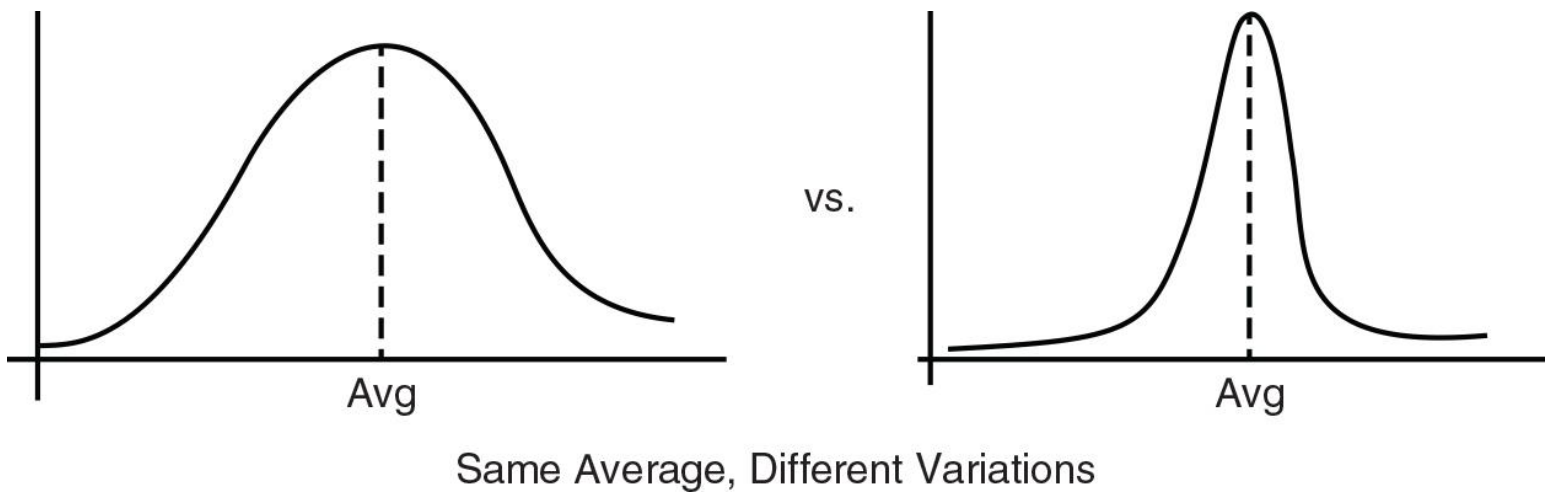

Figure 11.12 Descriptive statistics help us understand the composition of our data and how it compares with other, related data sets.

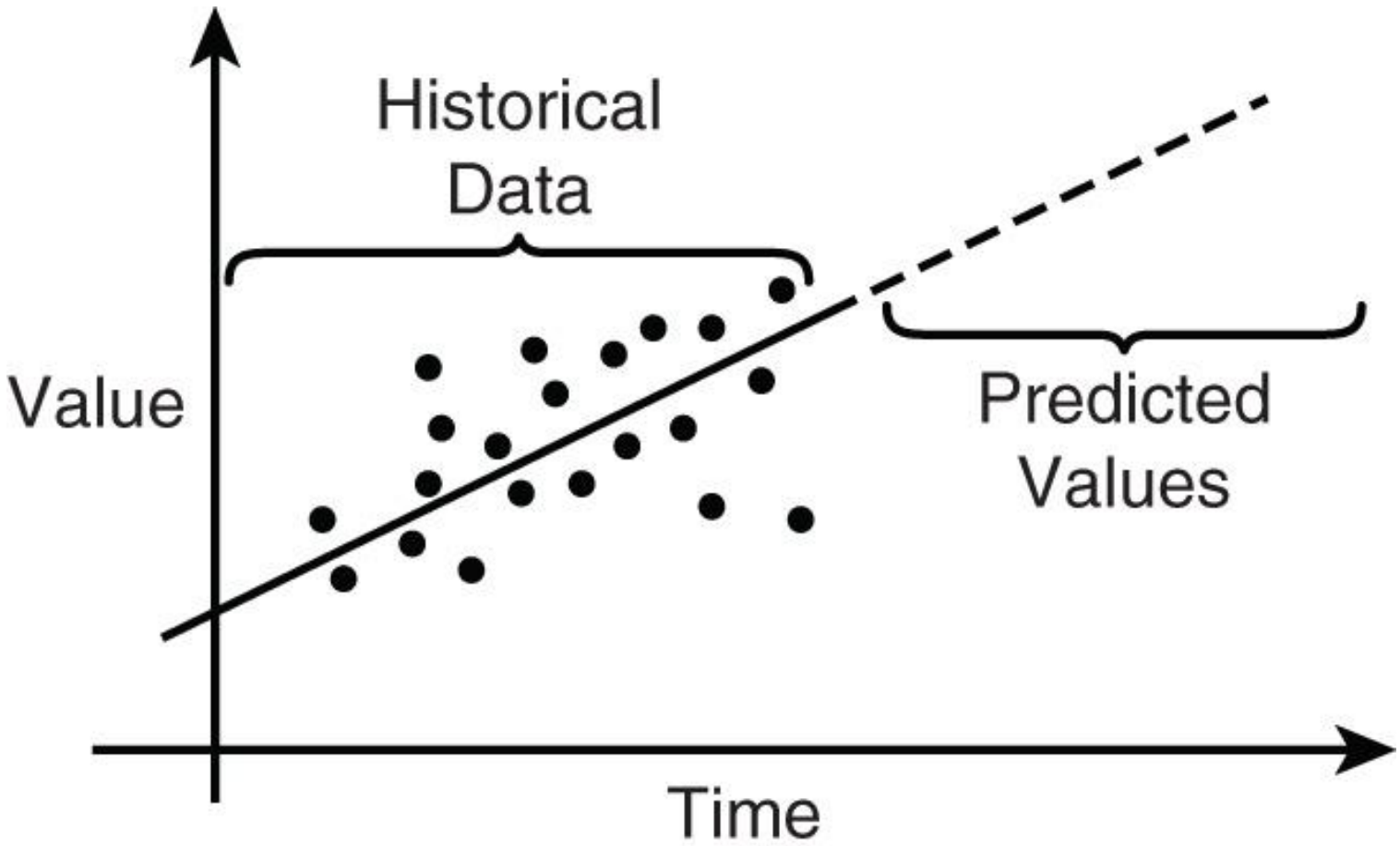

Figure 11.13 Predictive statistics help us make inferences about new situations using existing data.

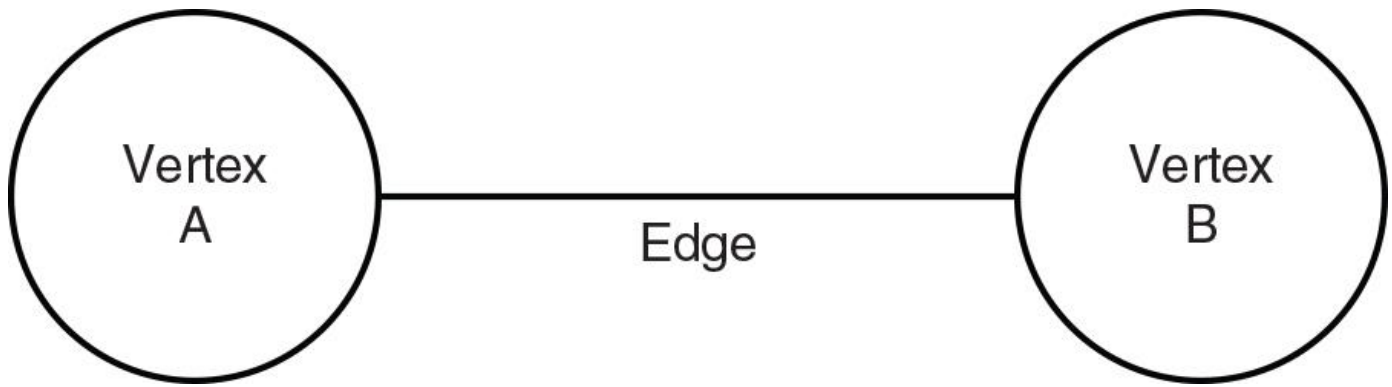

Figure 12.1 A simple graph with two vertices and one edge.

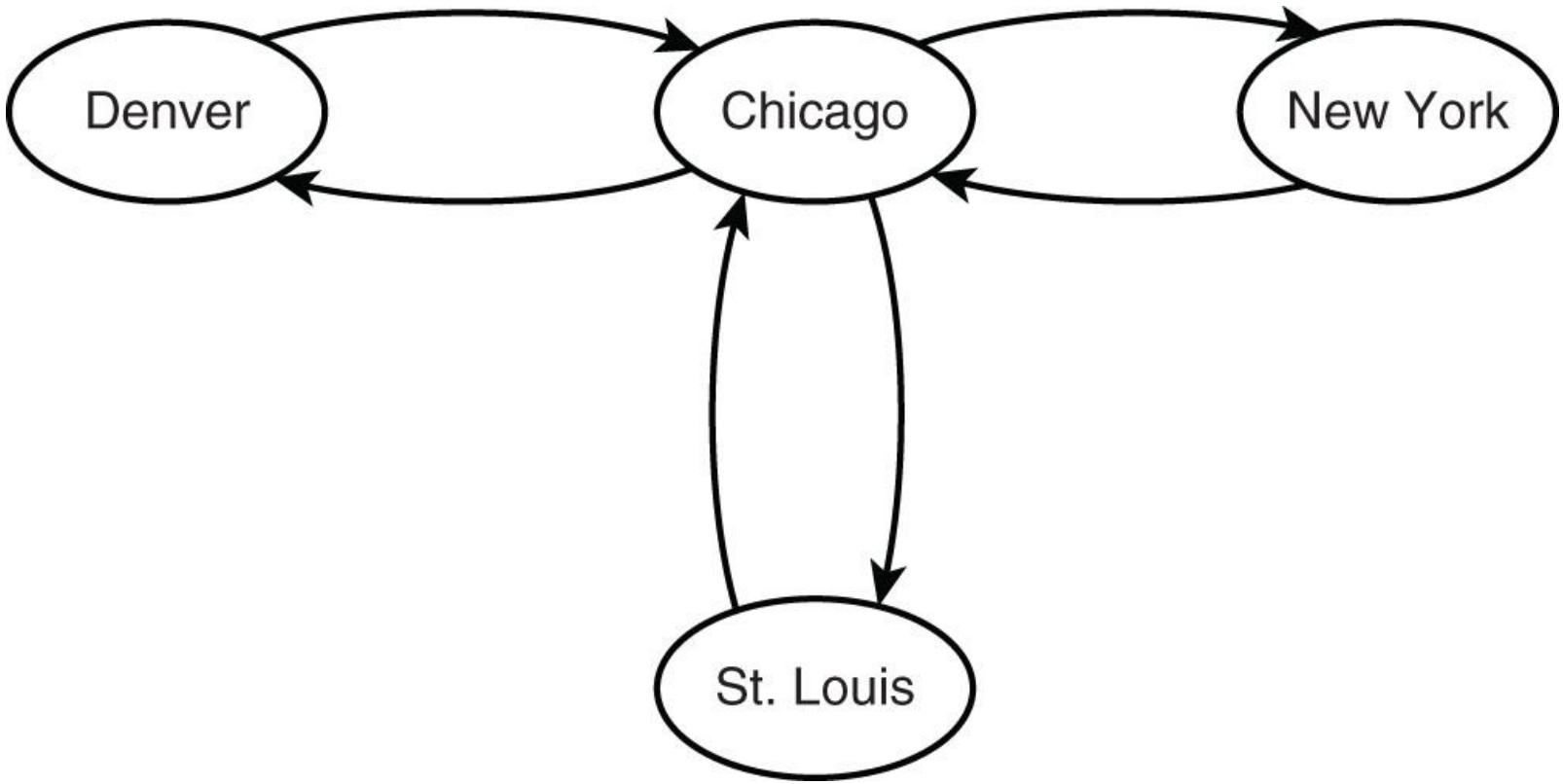

Figure 12.2 Highways between cities are modeled as vertices and edges.

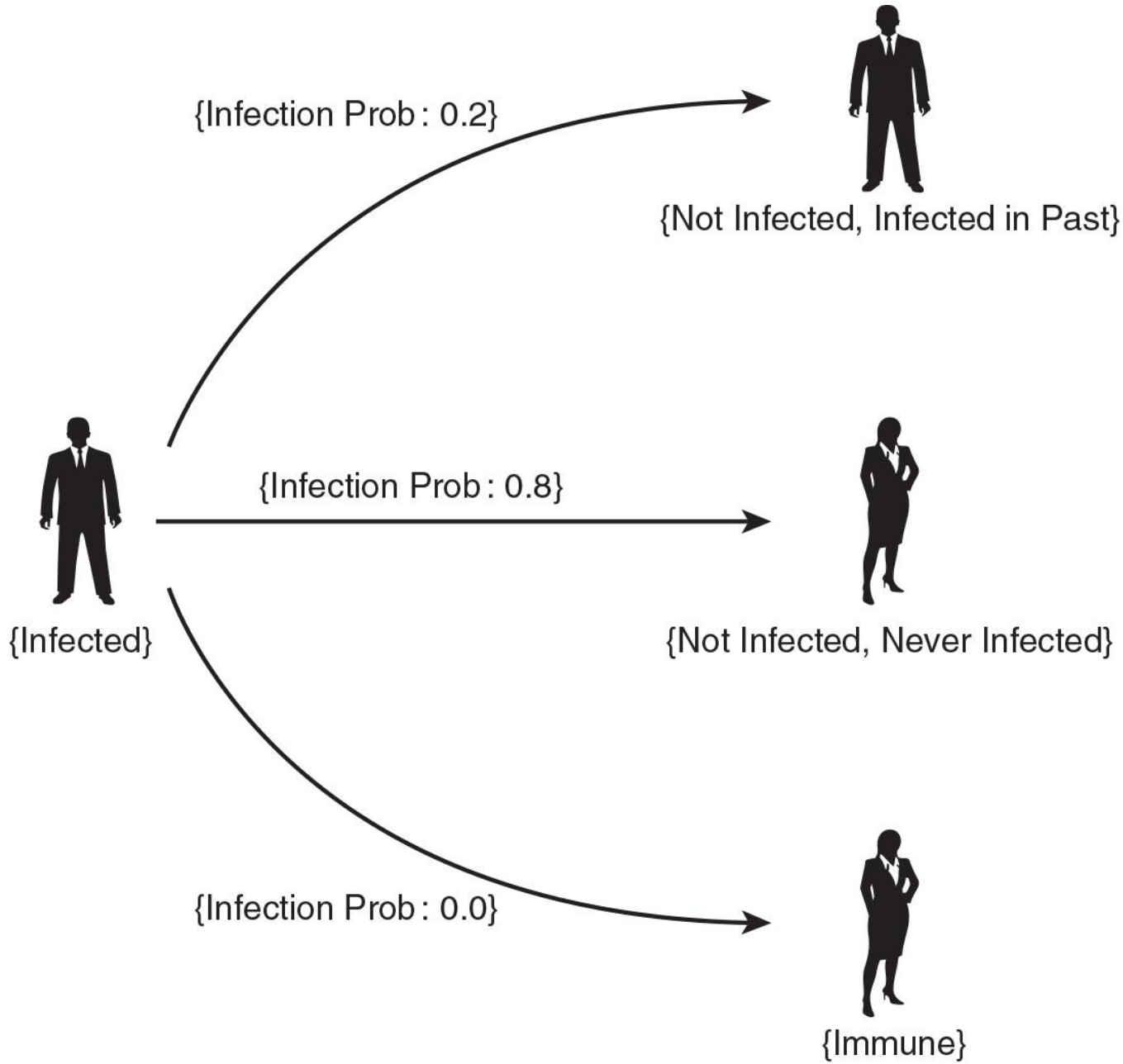

Figure 12.3 The spread of flu and other infectious diseases is modeled as graphs.

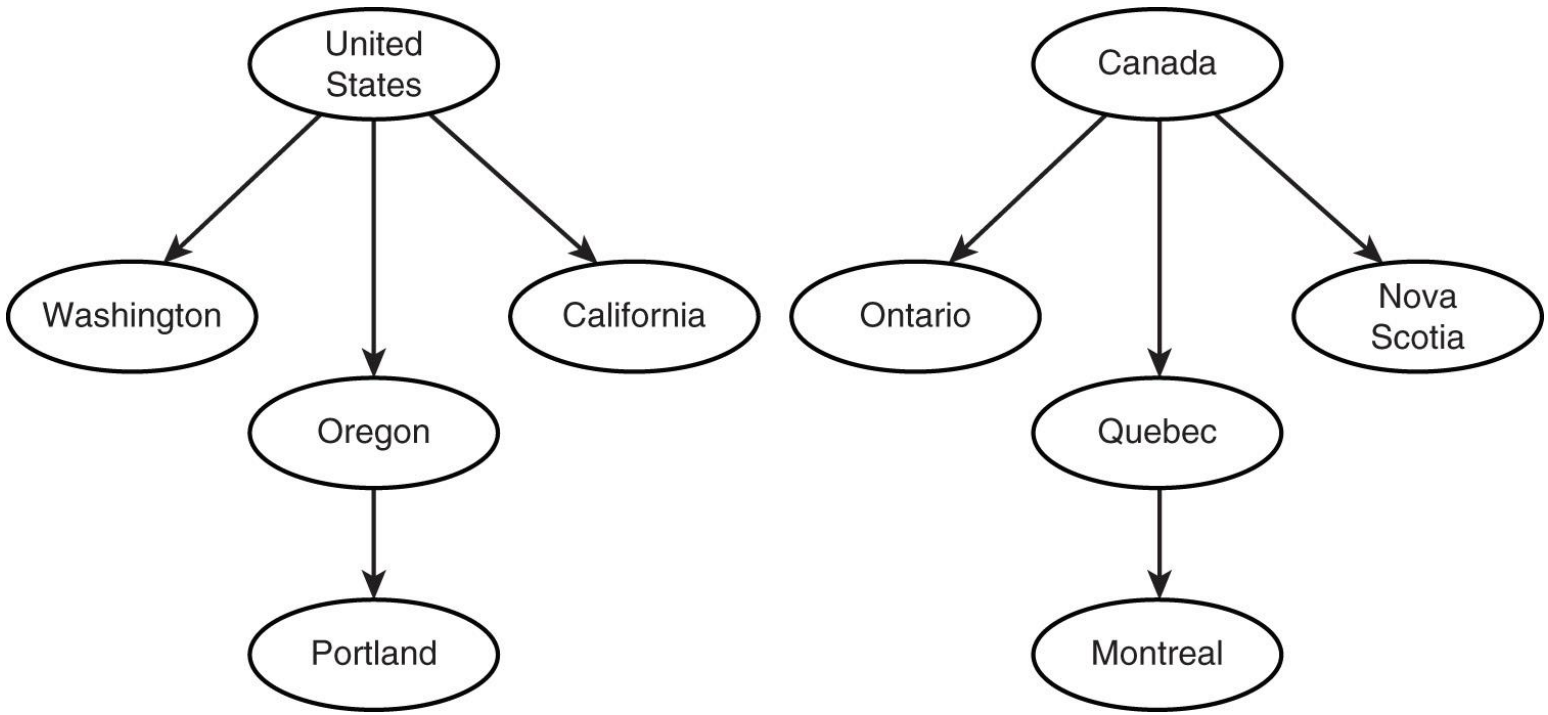

Figure 12.4 Hierarchical government structures modeled as graphs.

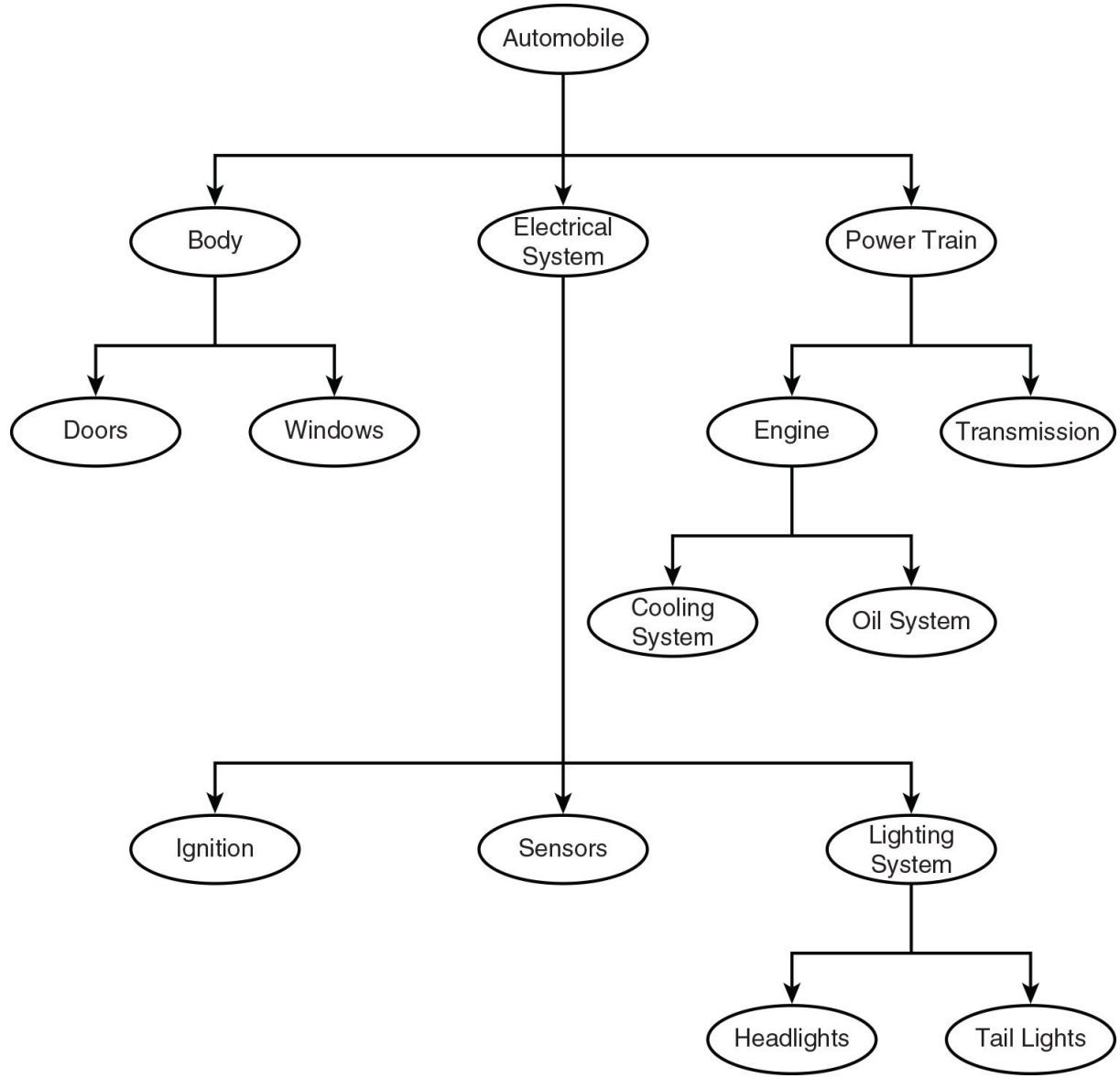

Figure 12.5 Part-of relations, such as parts of a car, are modeled using a graph.

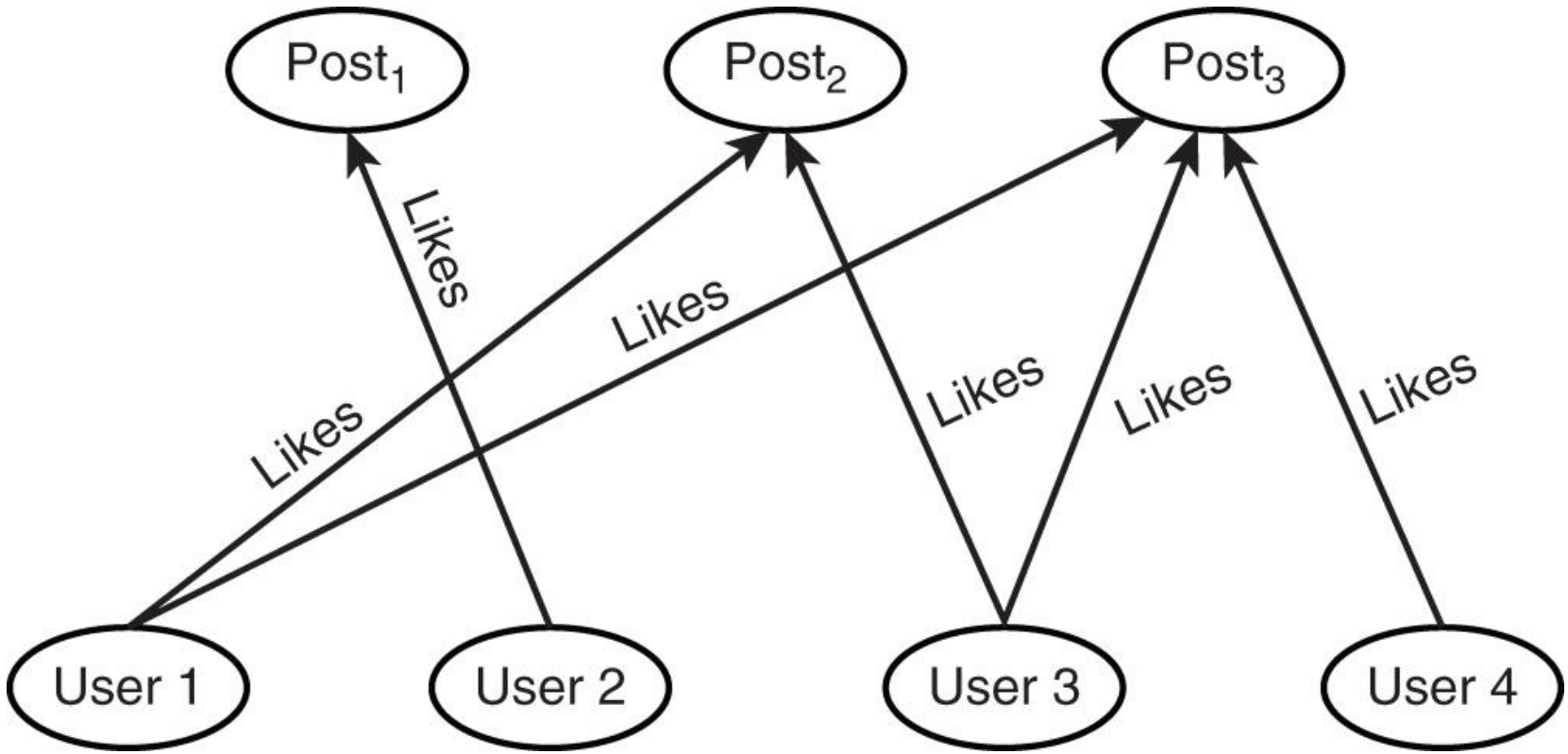

Figure 12.6 Social media posts and likes are modeled by graphs.

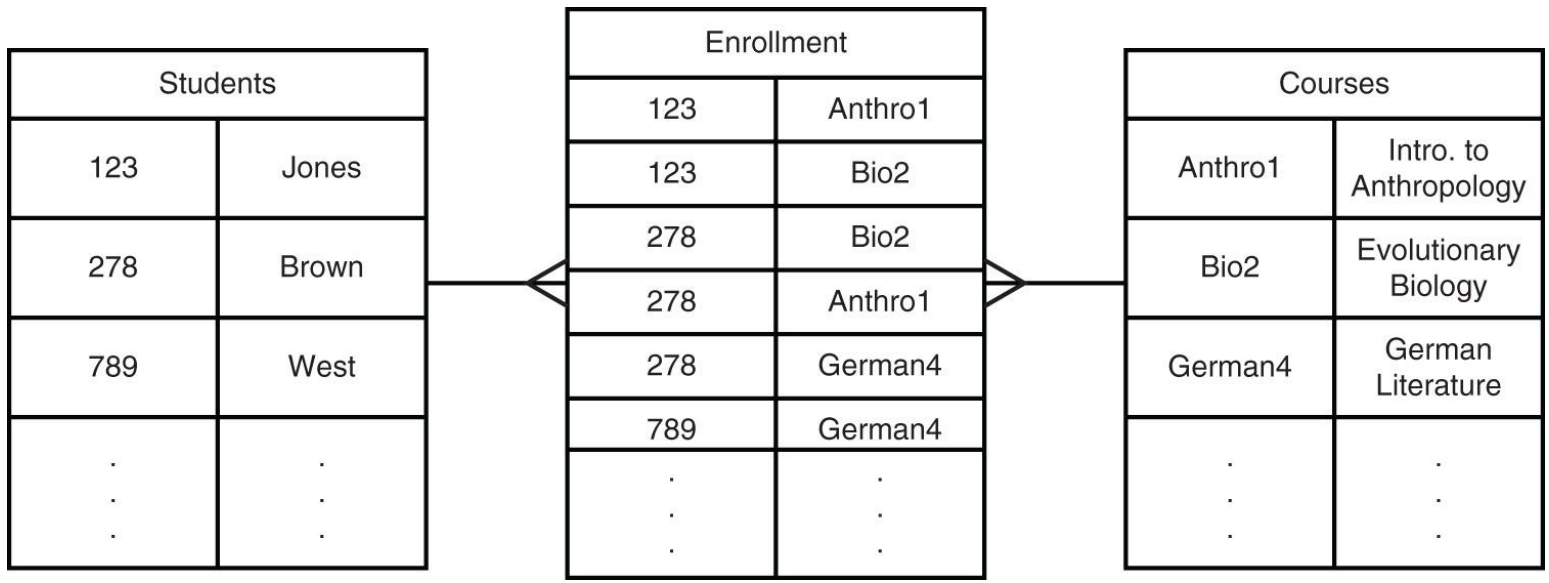

Figure 12.7 Representing a student-course relation in a relational database.

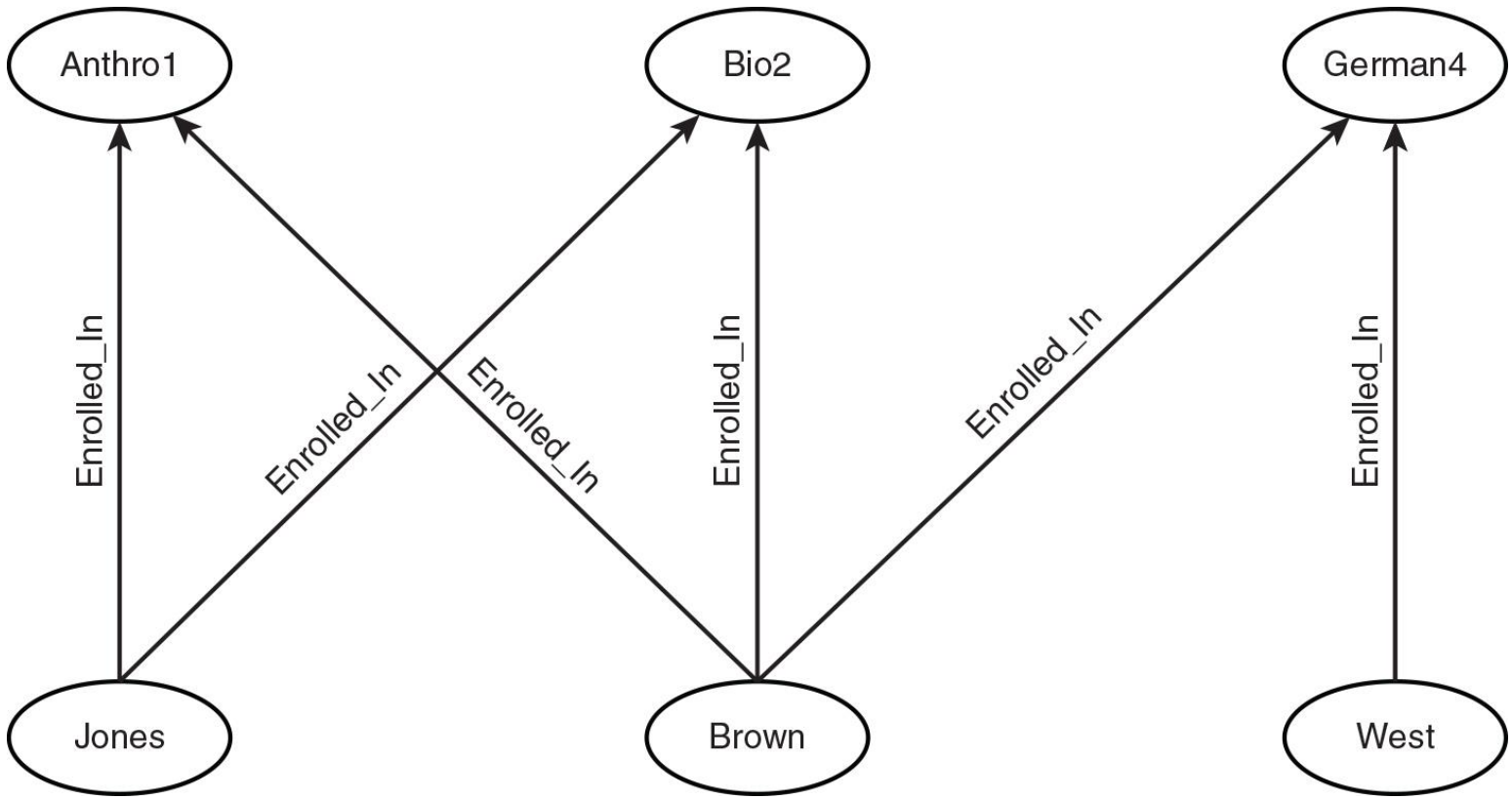

Figure 12.8 Representing a student-course relation in a graph database.

Course_of_Infection	
Patient	Infected By
Patient A	Patient B
Patient B	Patient C
Patient C	Patient D
Patient D	Patient E
Patient E	Patient F
Patient F	Patient G
Patient G	<Null>

Figure 12.9 Finding Patient Zero in an infectious disease investigation.

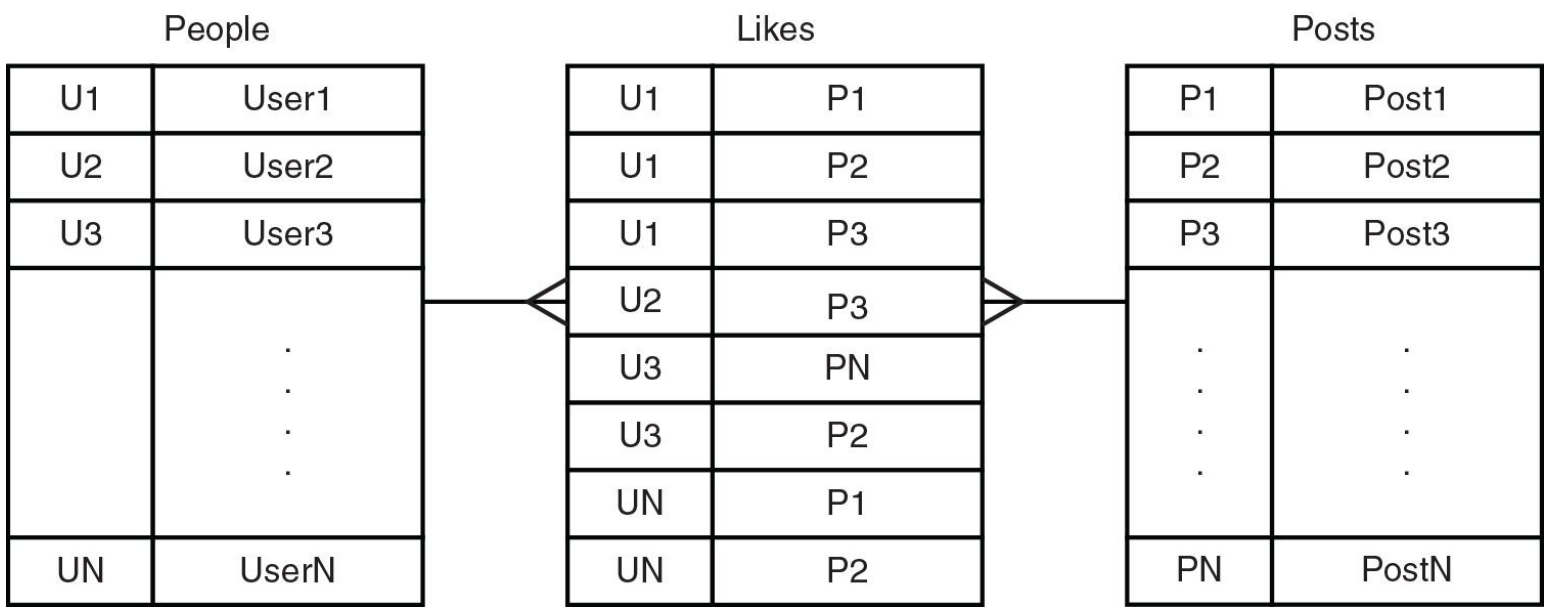

Figure 12.10 Modeling many-to-many relations in a relational database.

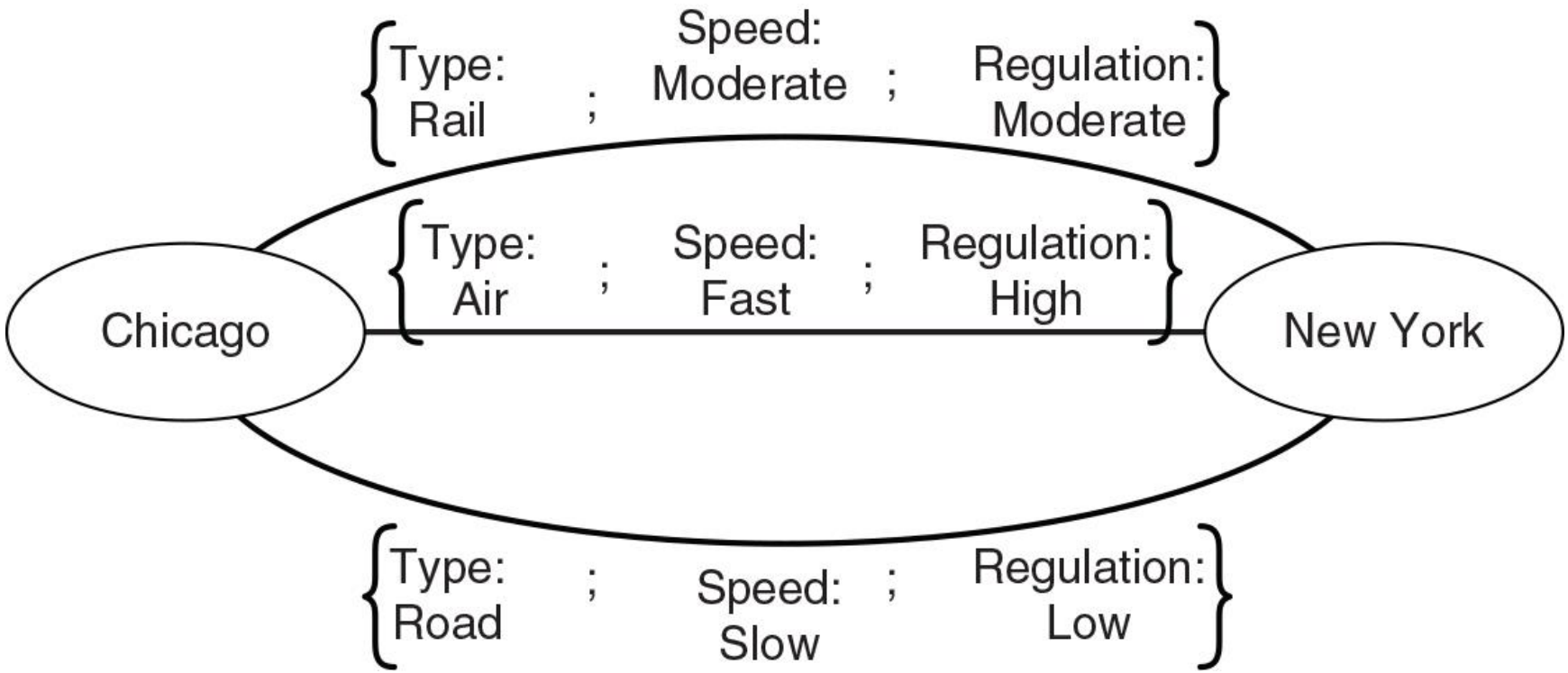

Figure 12.11 Modeling multiple types of relations in a graph database.

Figure 13.1 Vertices are used to represent objects.

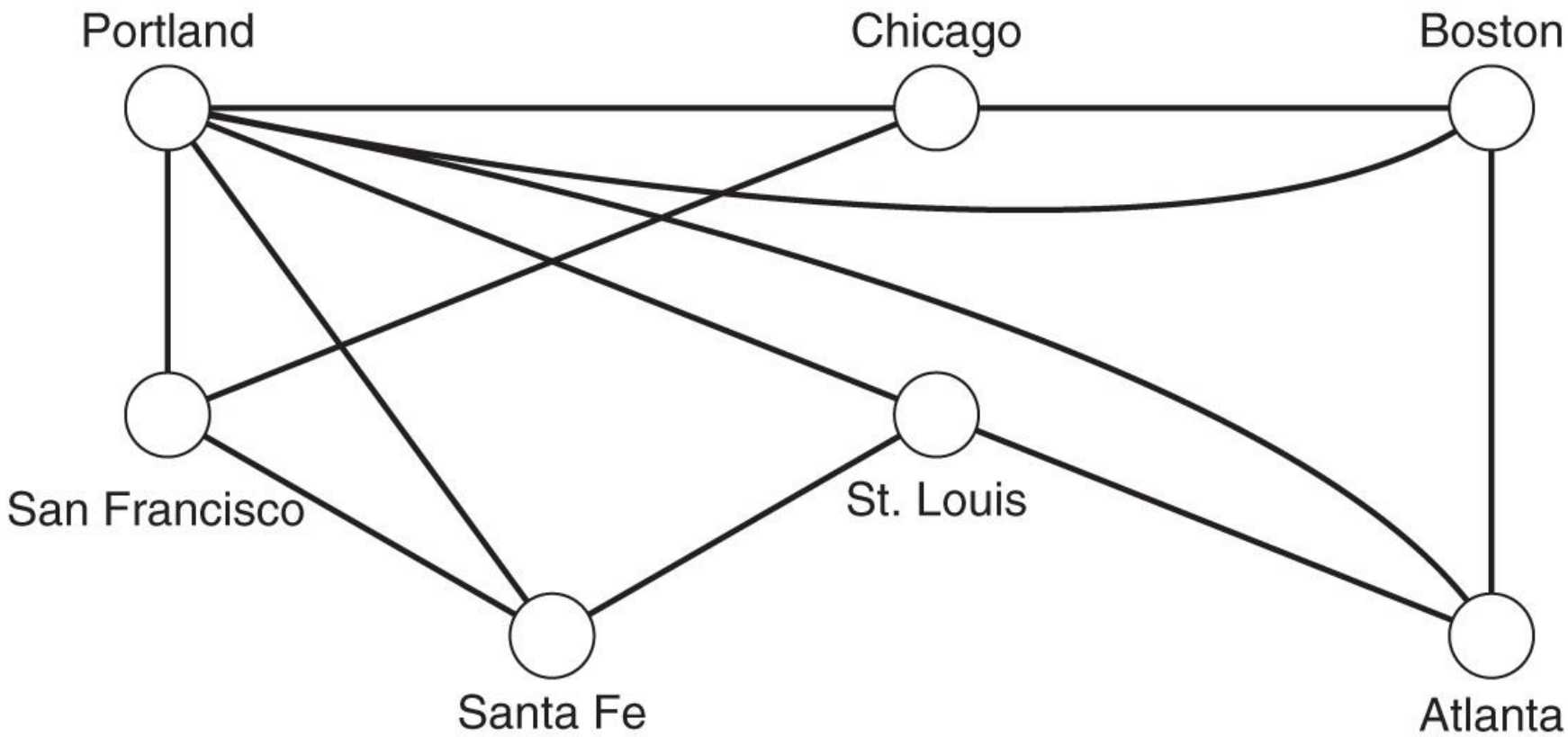

Figure 13.2 Edges represent relationships between vertices.

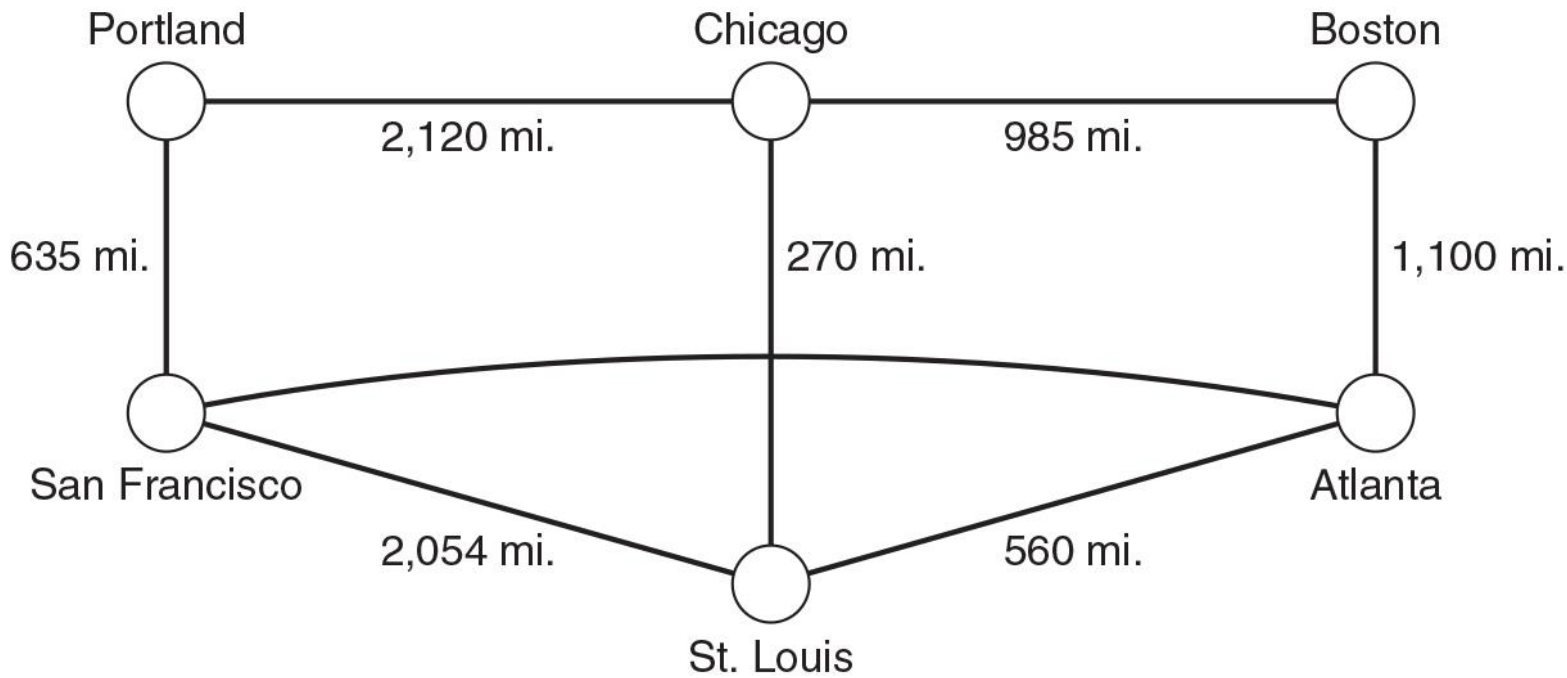

Figure 13.3 Weighted edges have a numeric property associated with them.

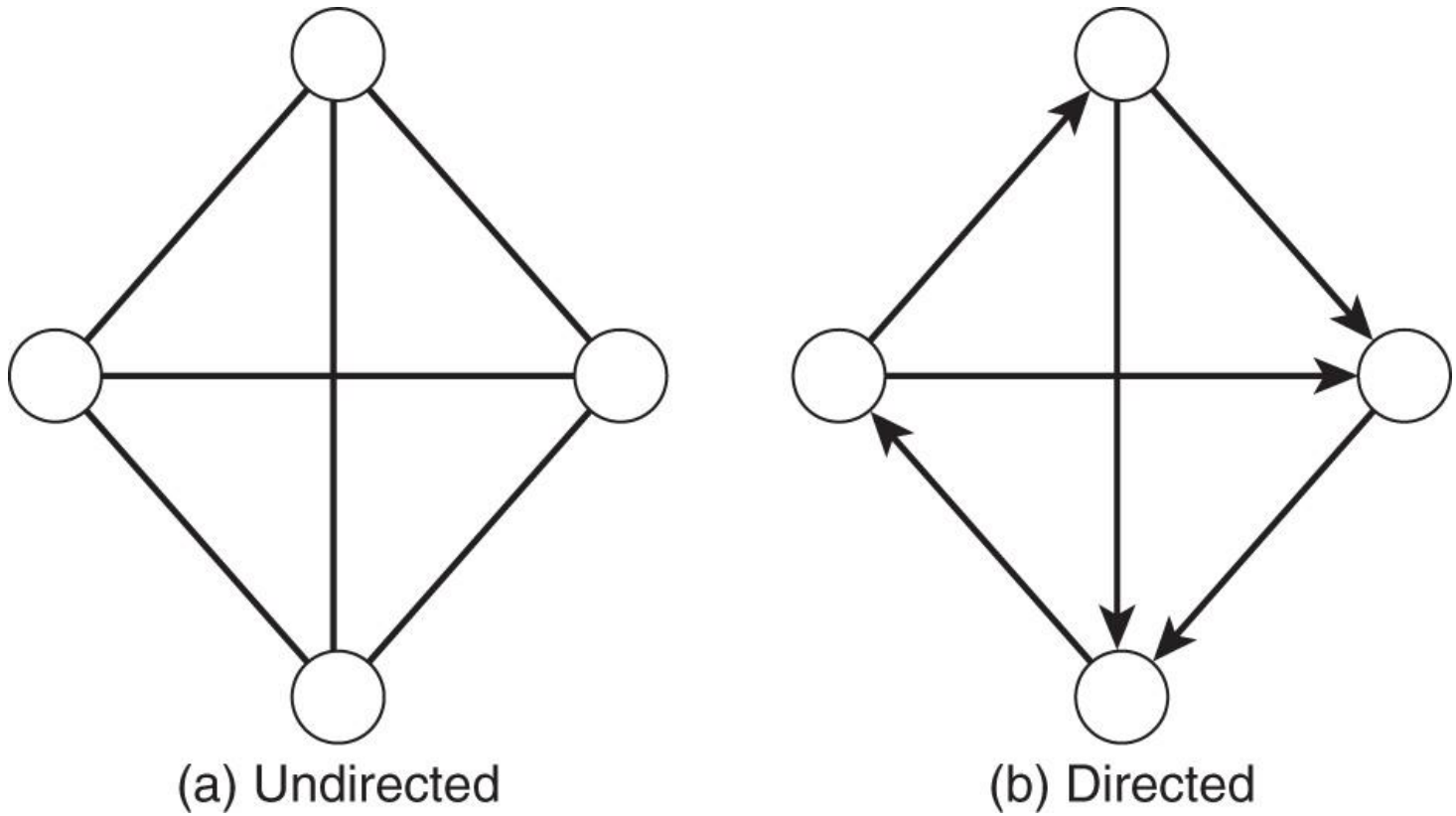

Figure 13.4 Directed and undirected edges further refine properties of relationships between vertices by capturing directionality.

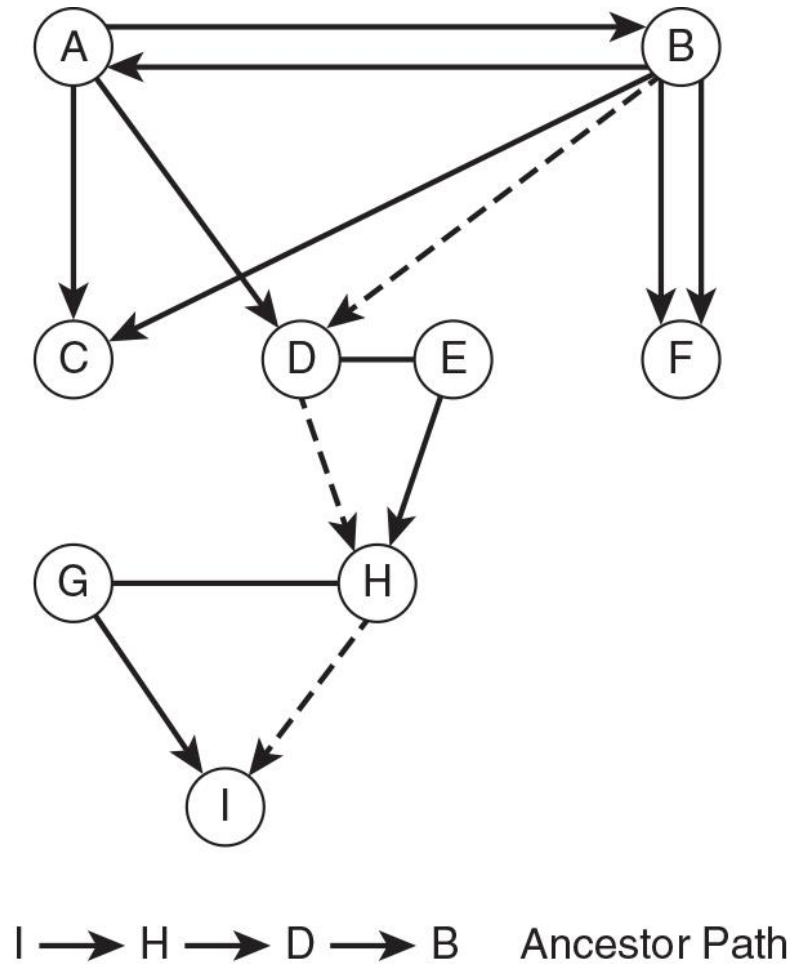

Figure 13.5 A path is a set of vertices and edges through a graph. The vertices and edges from B to D to H to I are a path from vertex B to vertex I.

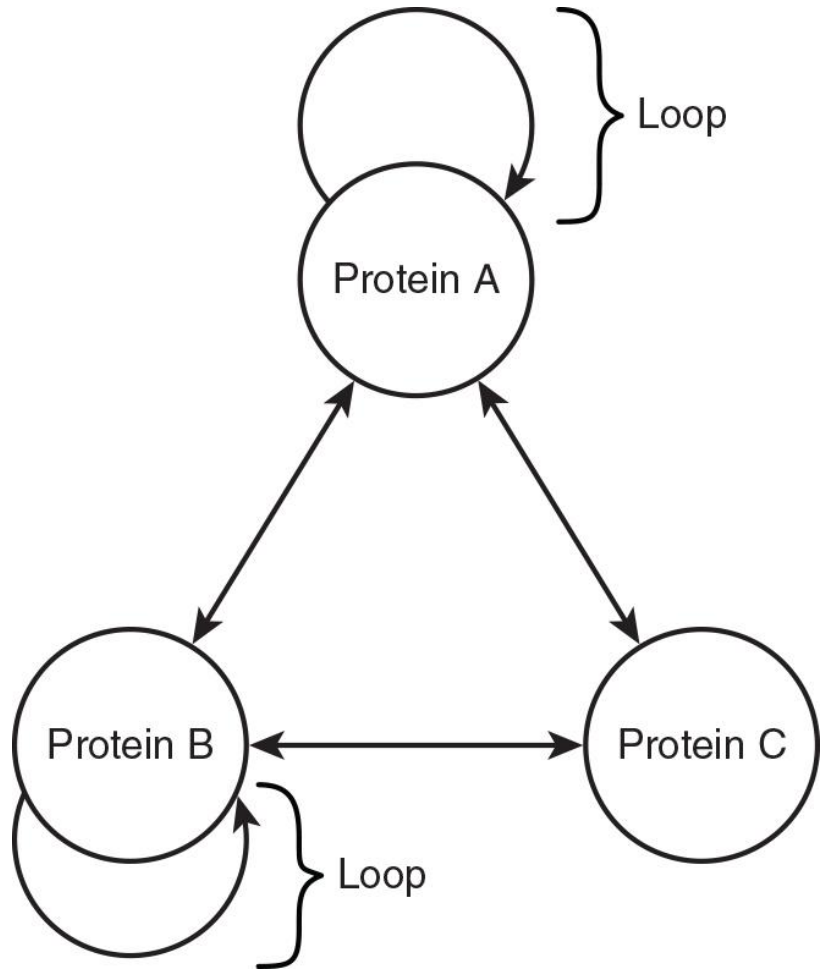

Figure 13.6 A loop is an edge that links a vertex to itself.

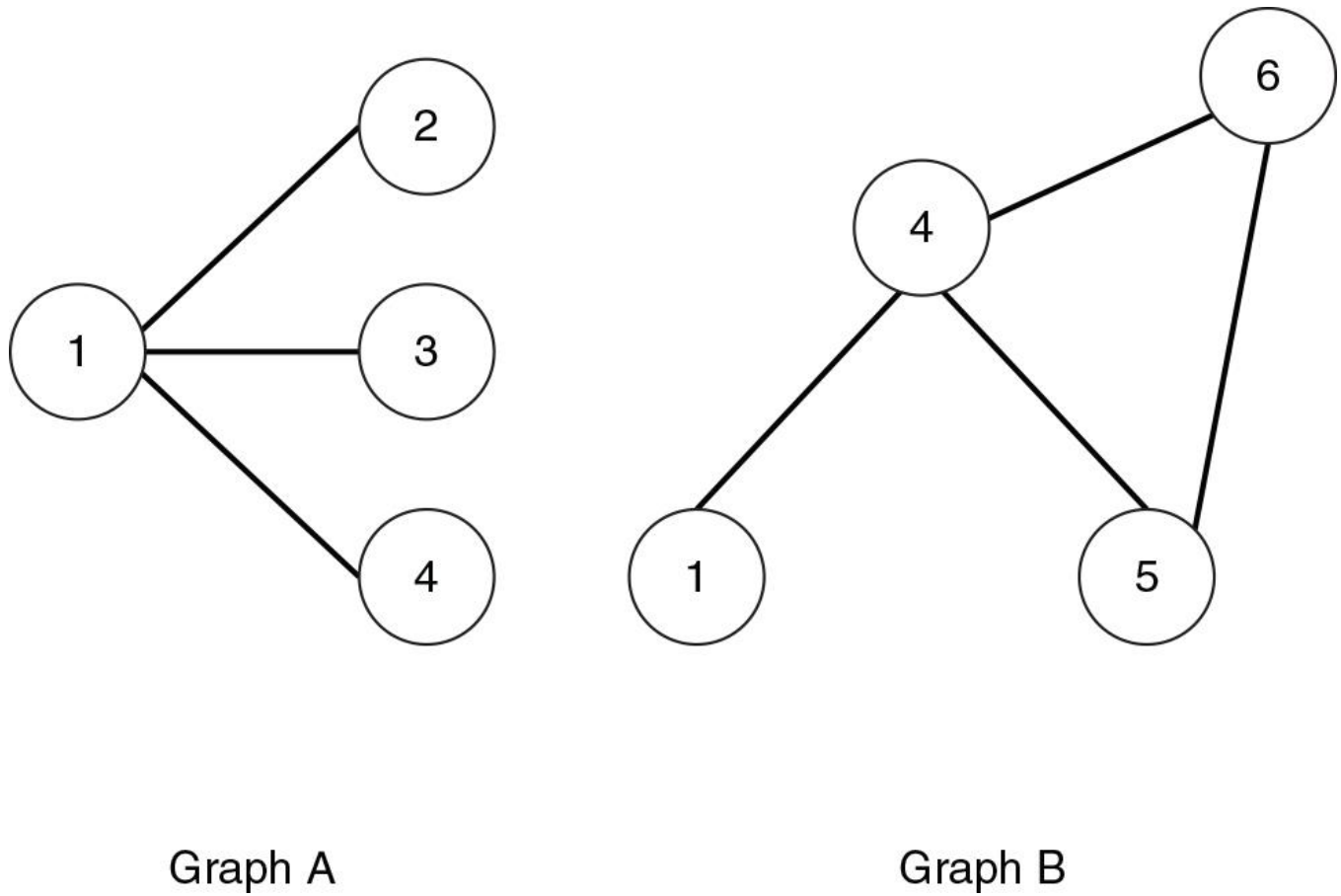

Figure 13.7 Two distinct graphs, A and B.

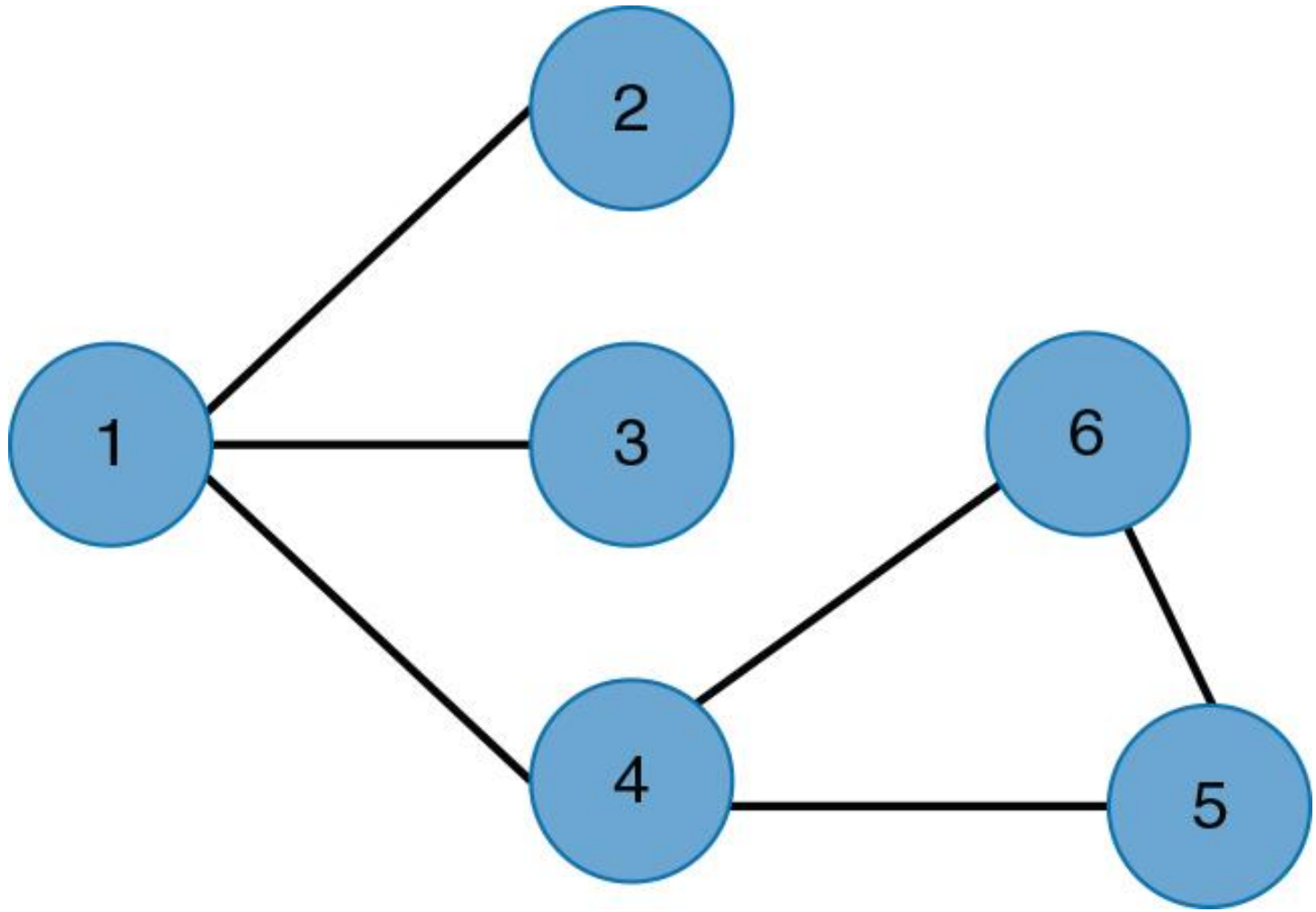

Figure 13.8 Union of graphs A and B.

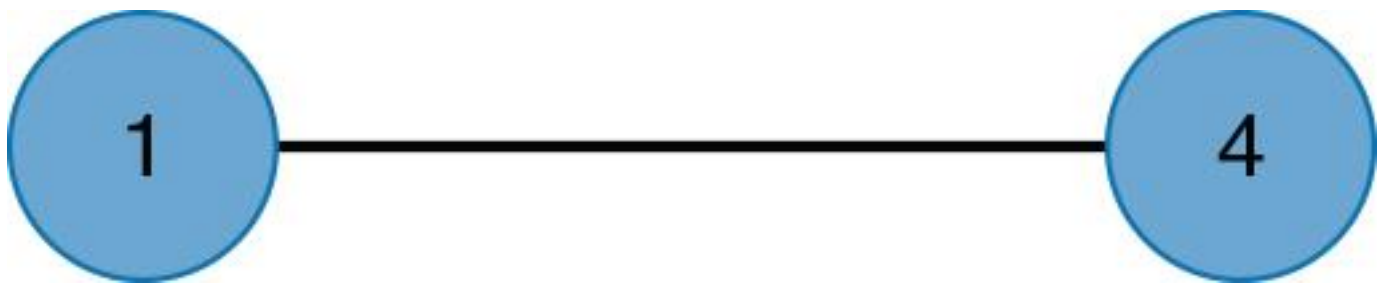

Figure 13.9 Intersection of graphs A and B.

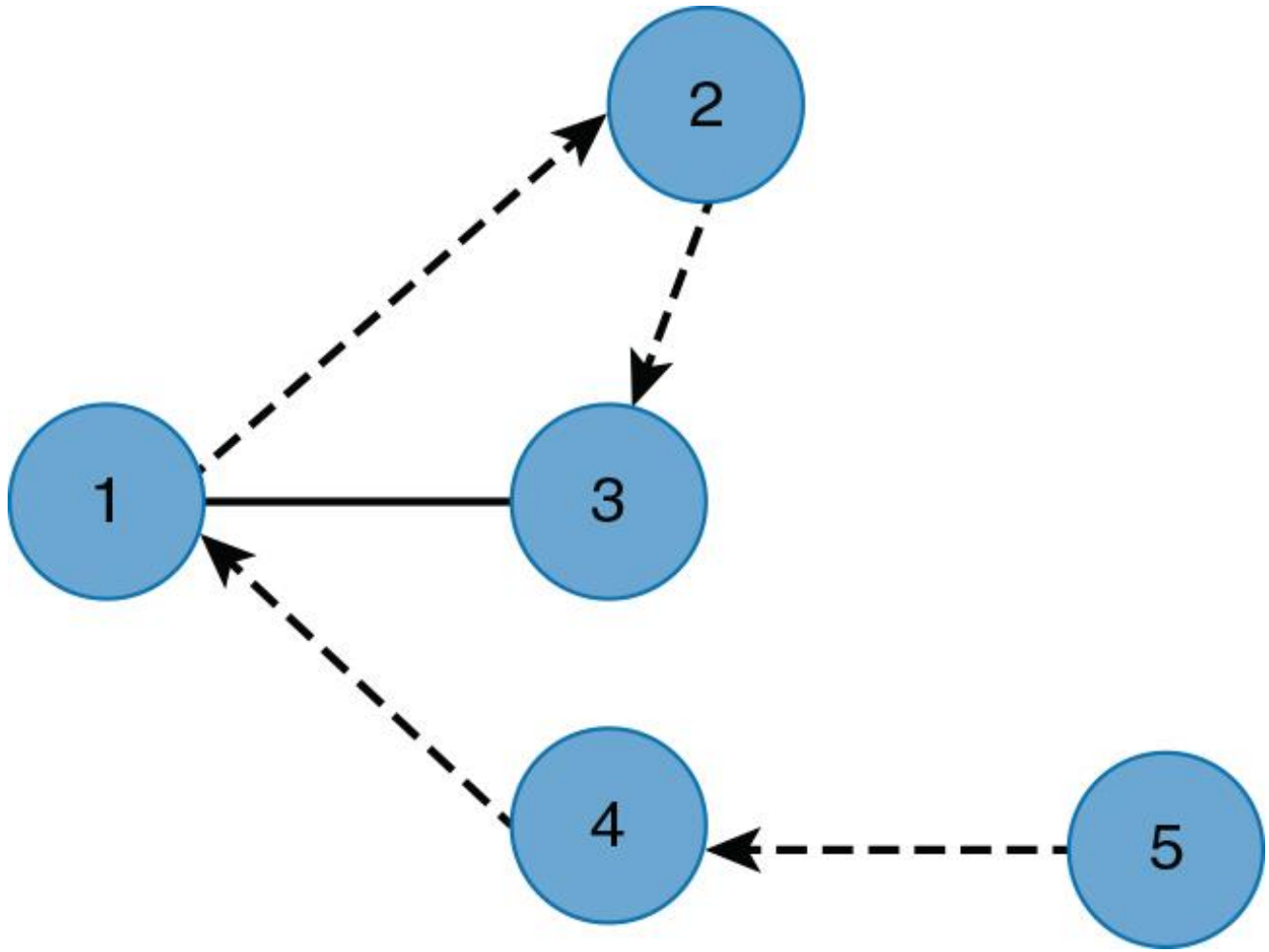

Figure 13.10 Graph traversal is the process of visiting all nodes in a graph.

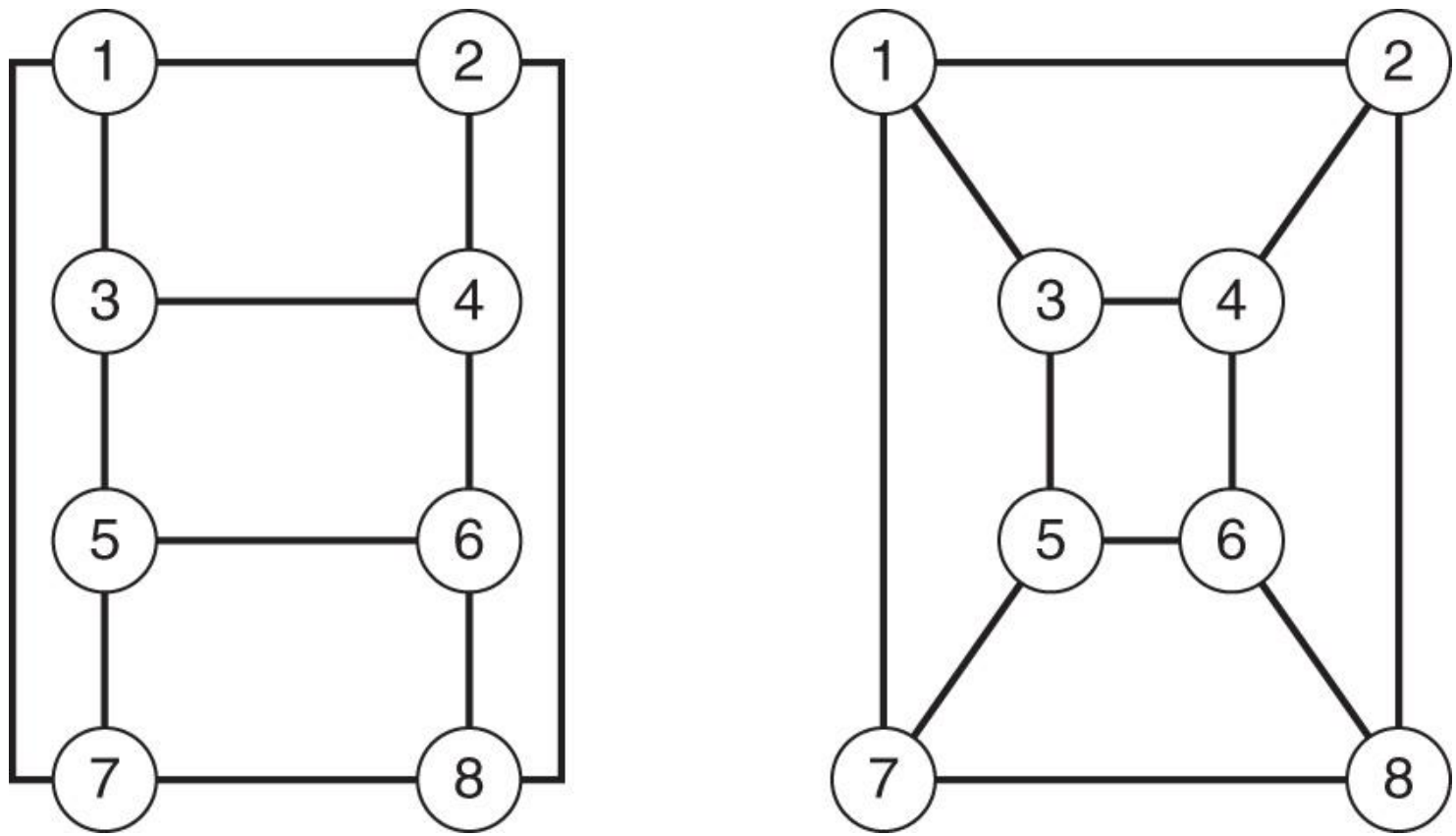

Figure 13.11 Example of two isomorphic graphs.

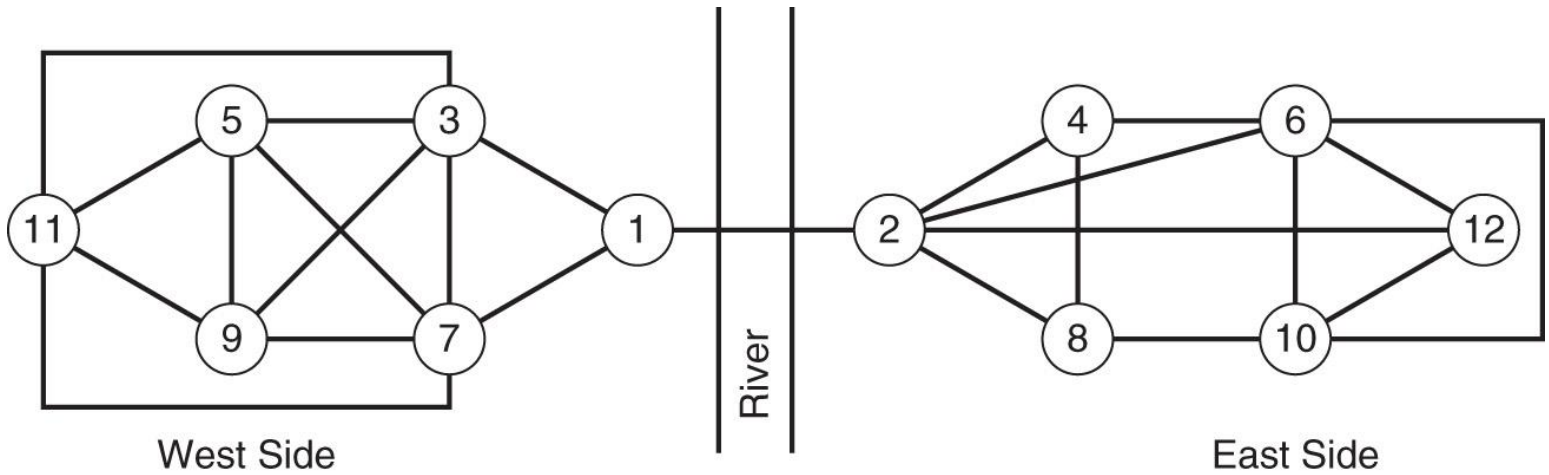

Figure 13.12 Betweenness helps identify bottlenecks in a graph.

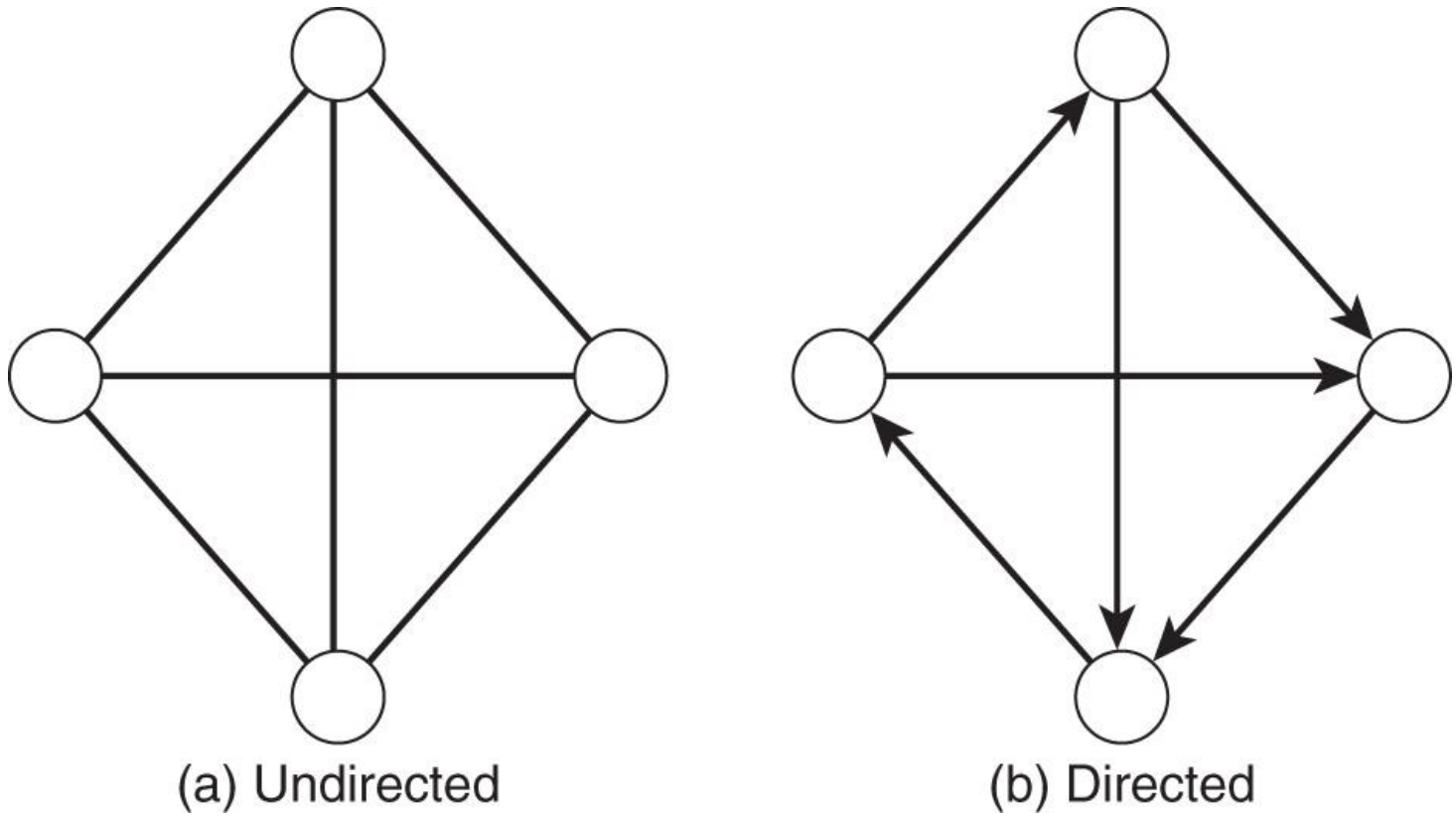

Figure 13.13 Undirected (a) and directed (b) graphs.

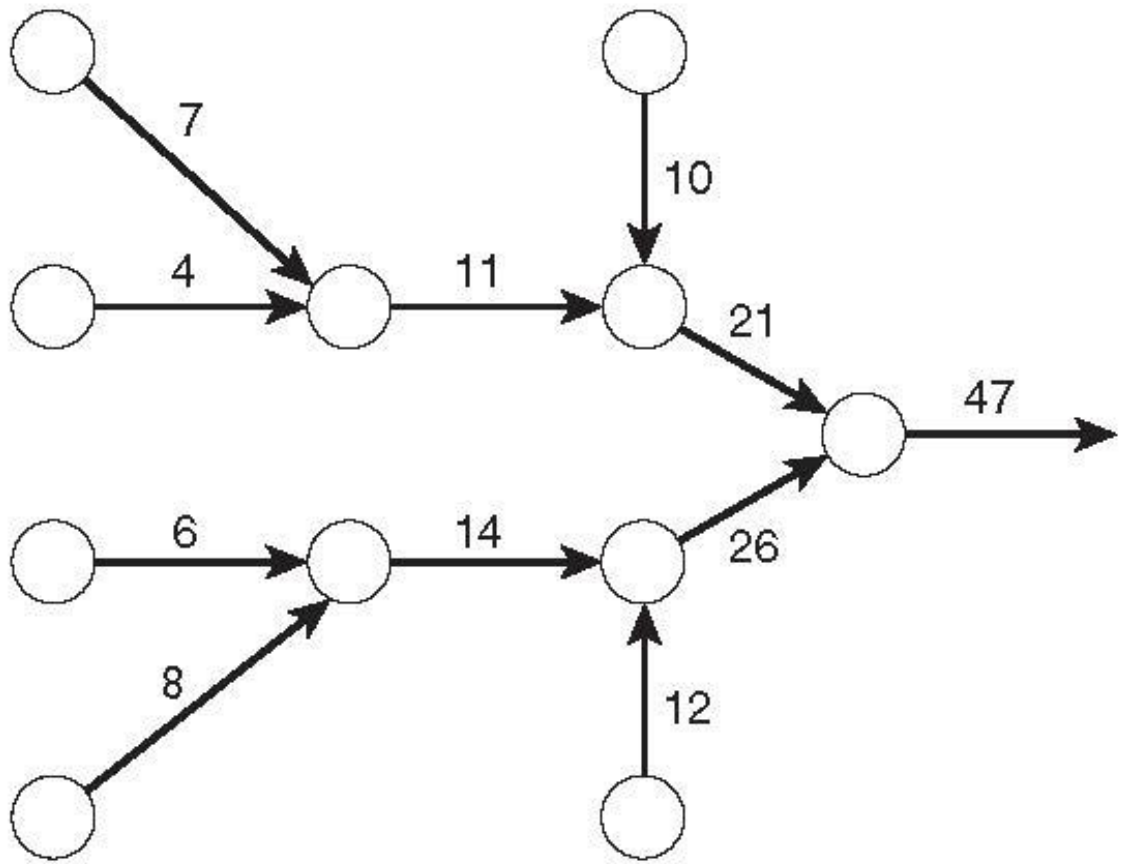

Figure 13.14 Flow networks capture information about capacities of edges and how they can be combined using vertices.

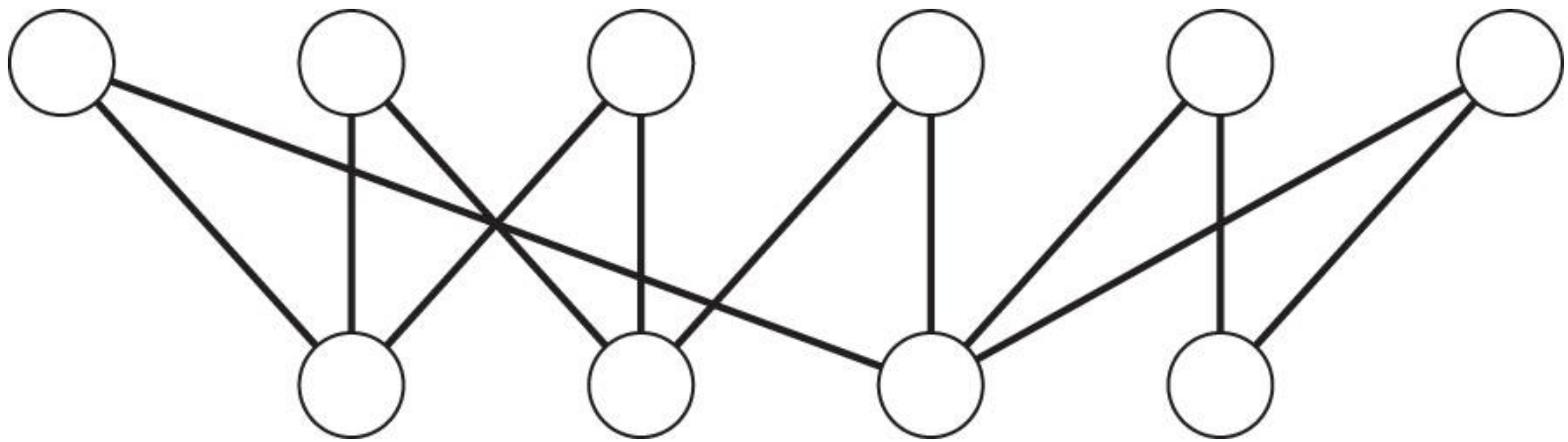

Figure 13.15 A bipartite graph consists of two subgroups of nodes.

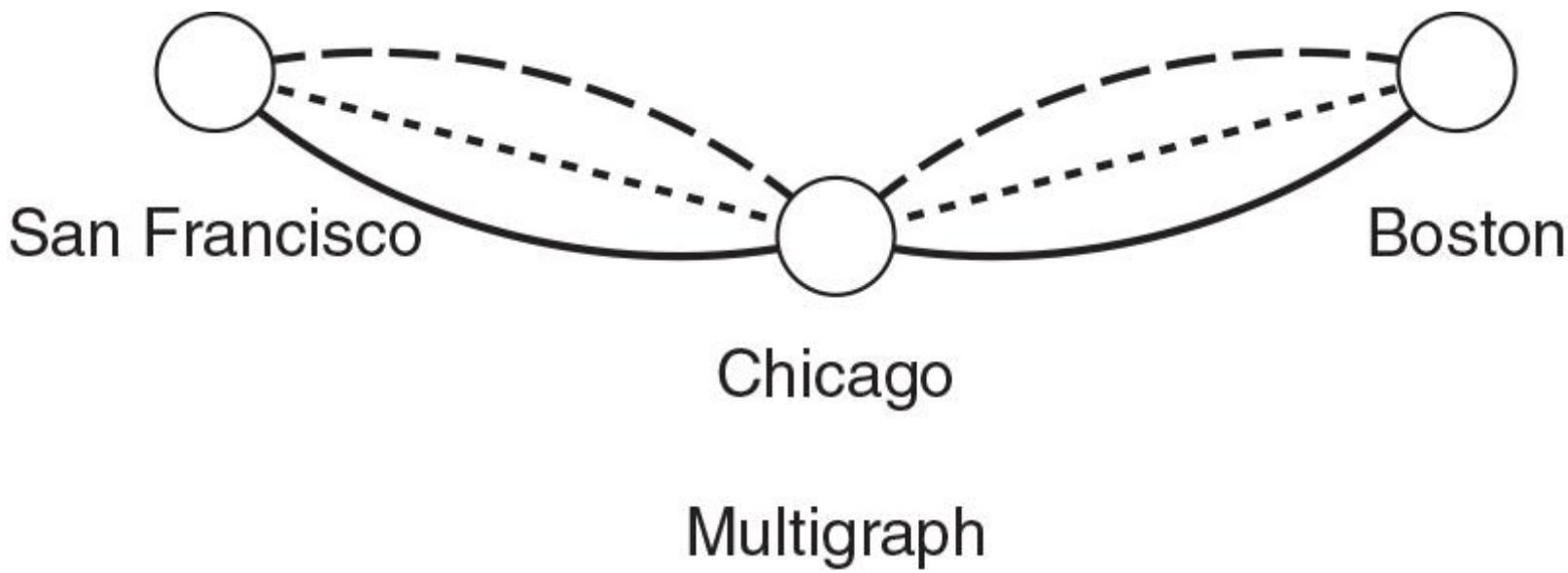

Figure 13.16 A multigraph is used to represent multiple types of relations between vertices.

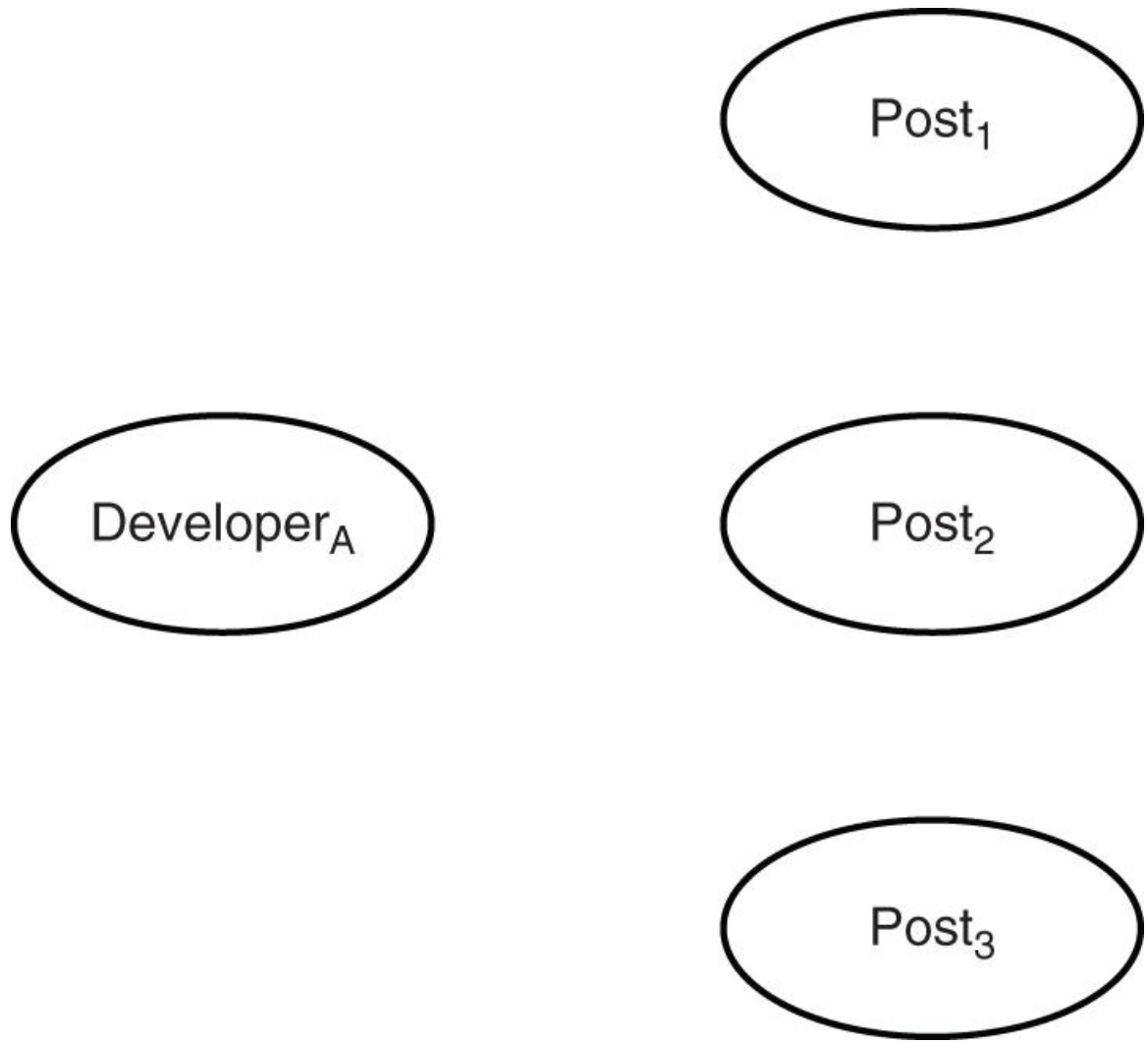

Figure 14.1 Developers and posts are two types of entities in the NoSQL social network.

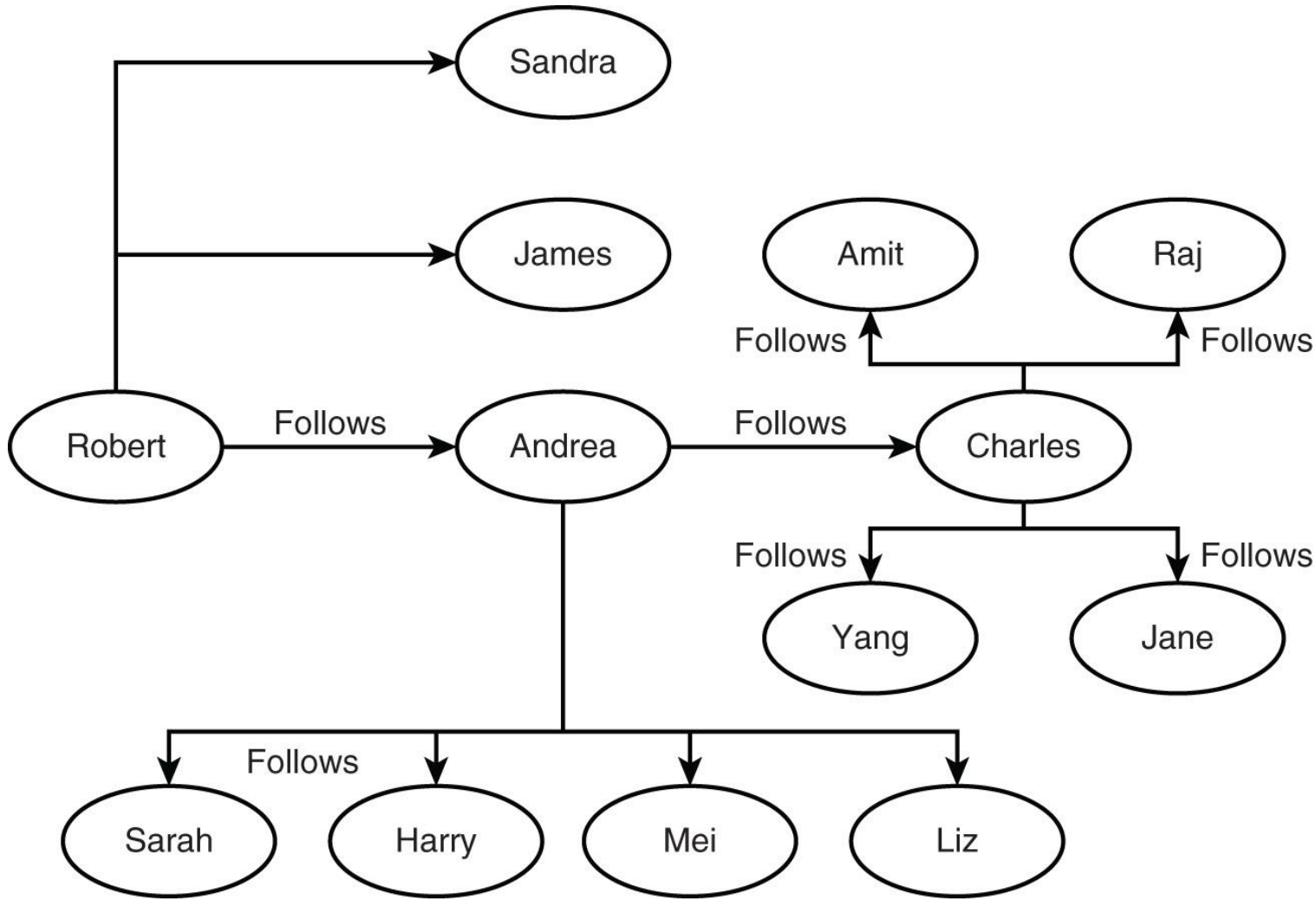

Figure 14.2 Sample set of “follows” relations in a NoSQL developer social network.

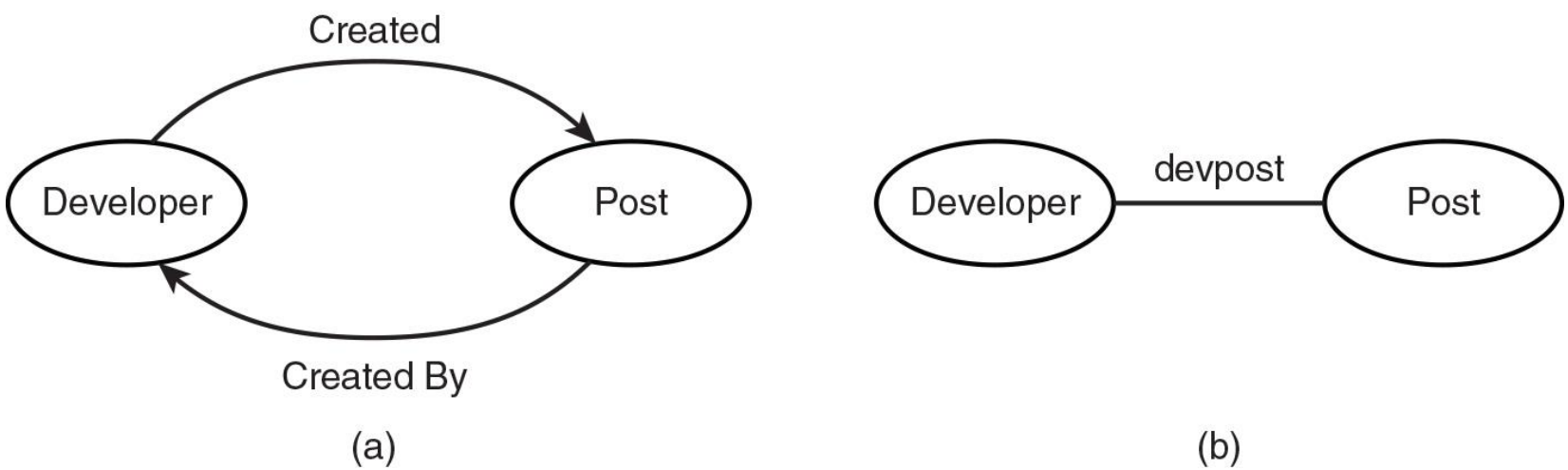

Figure 14.3 Developer and post with two directed edges (a); one edge not directed (b).

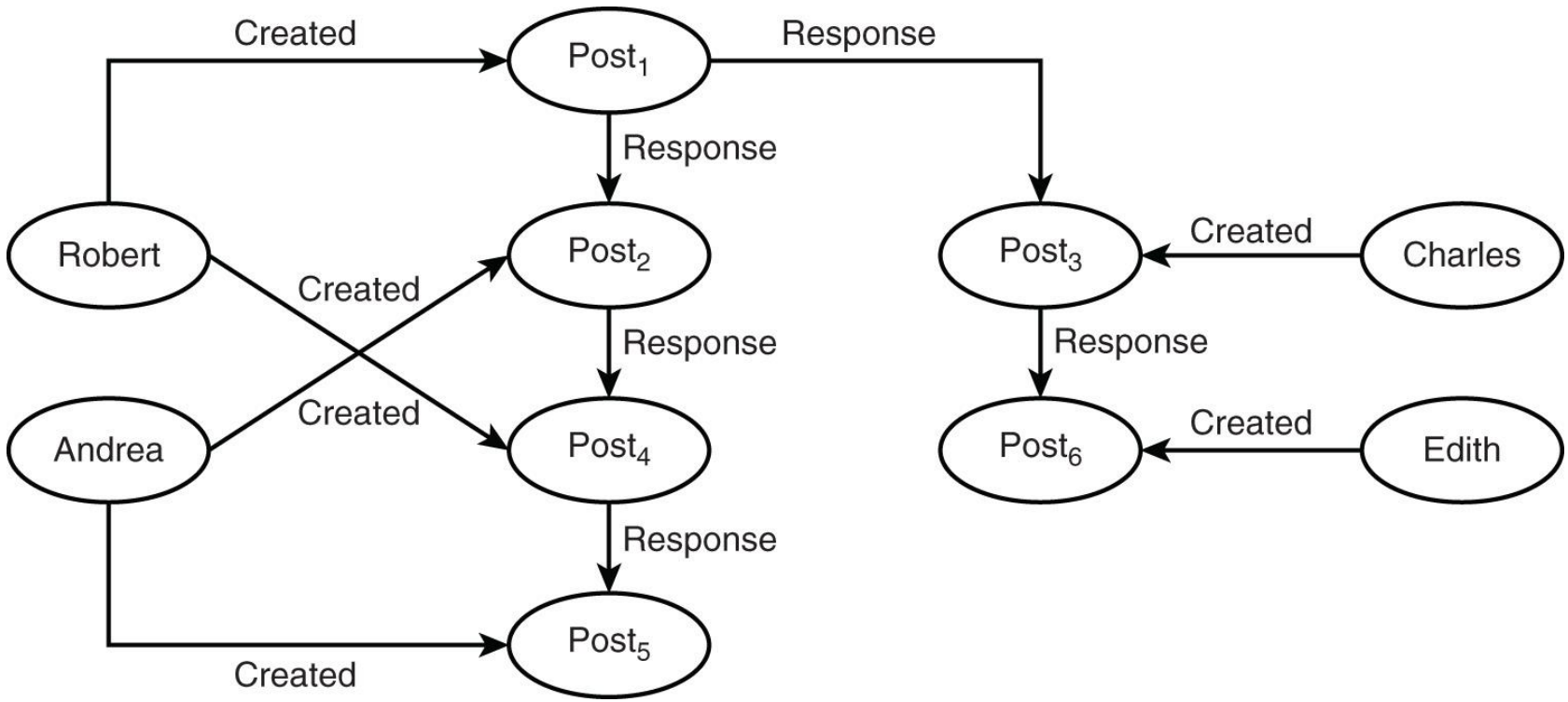

Figure 14.4 A conversation thread started with Robert posting a question.

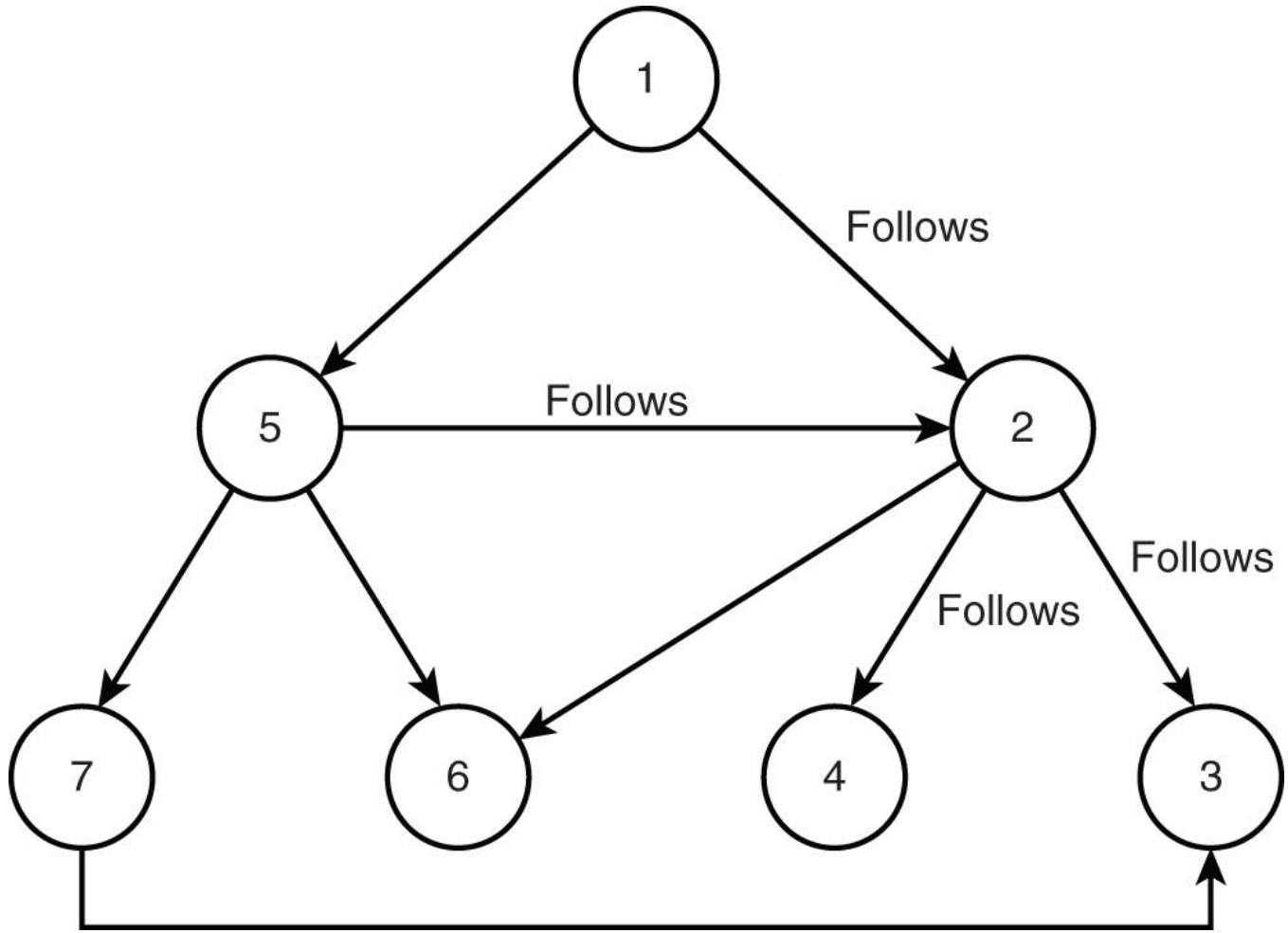

Figure 14.5 Sample directed graph with vertices with only in, out, and both in and out edges.

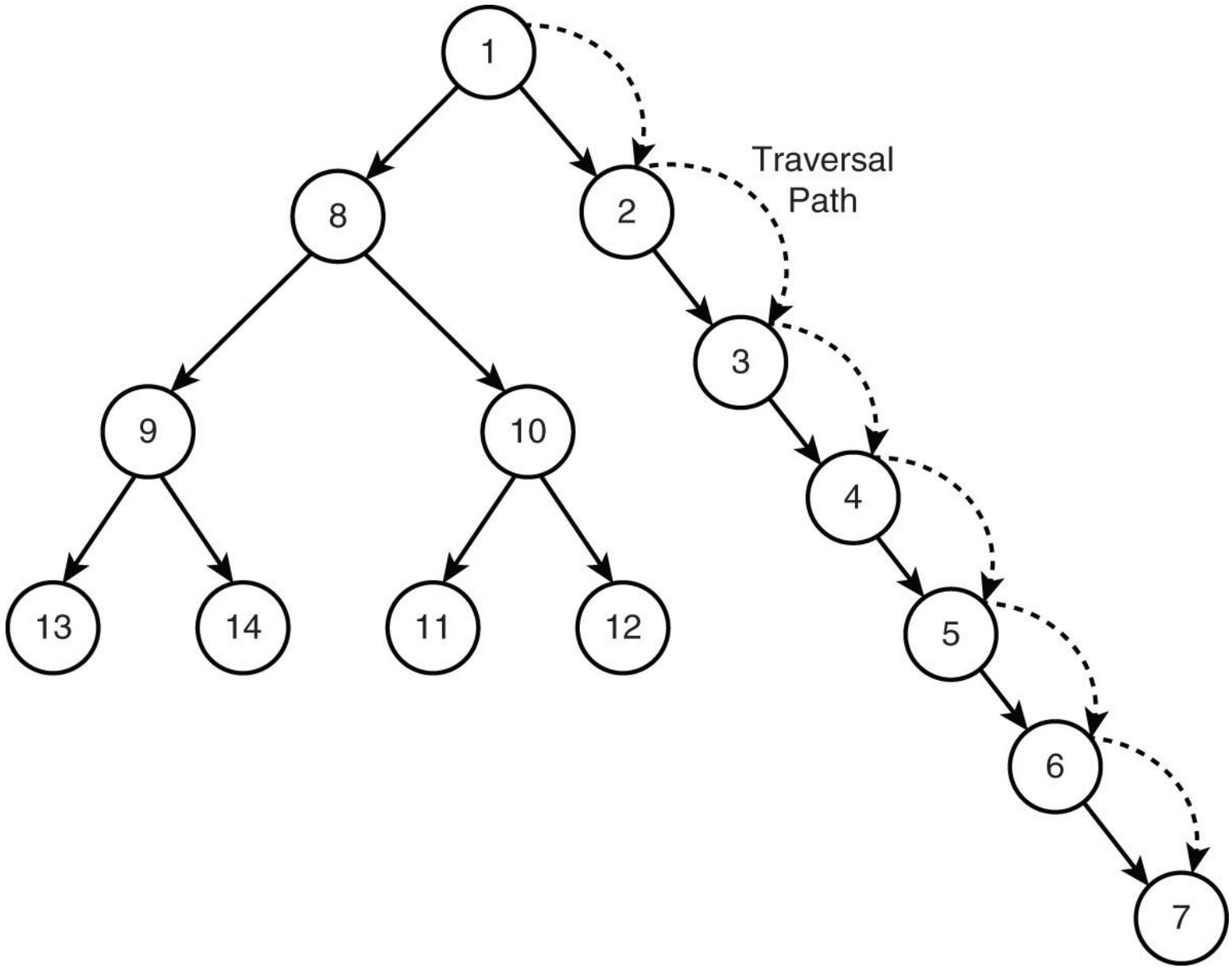

Figure 14.6 Example of a depth-first search.

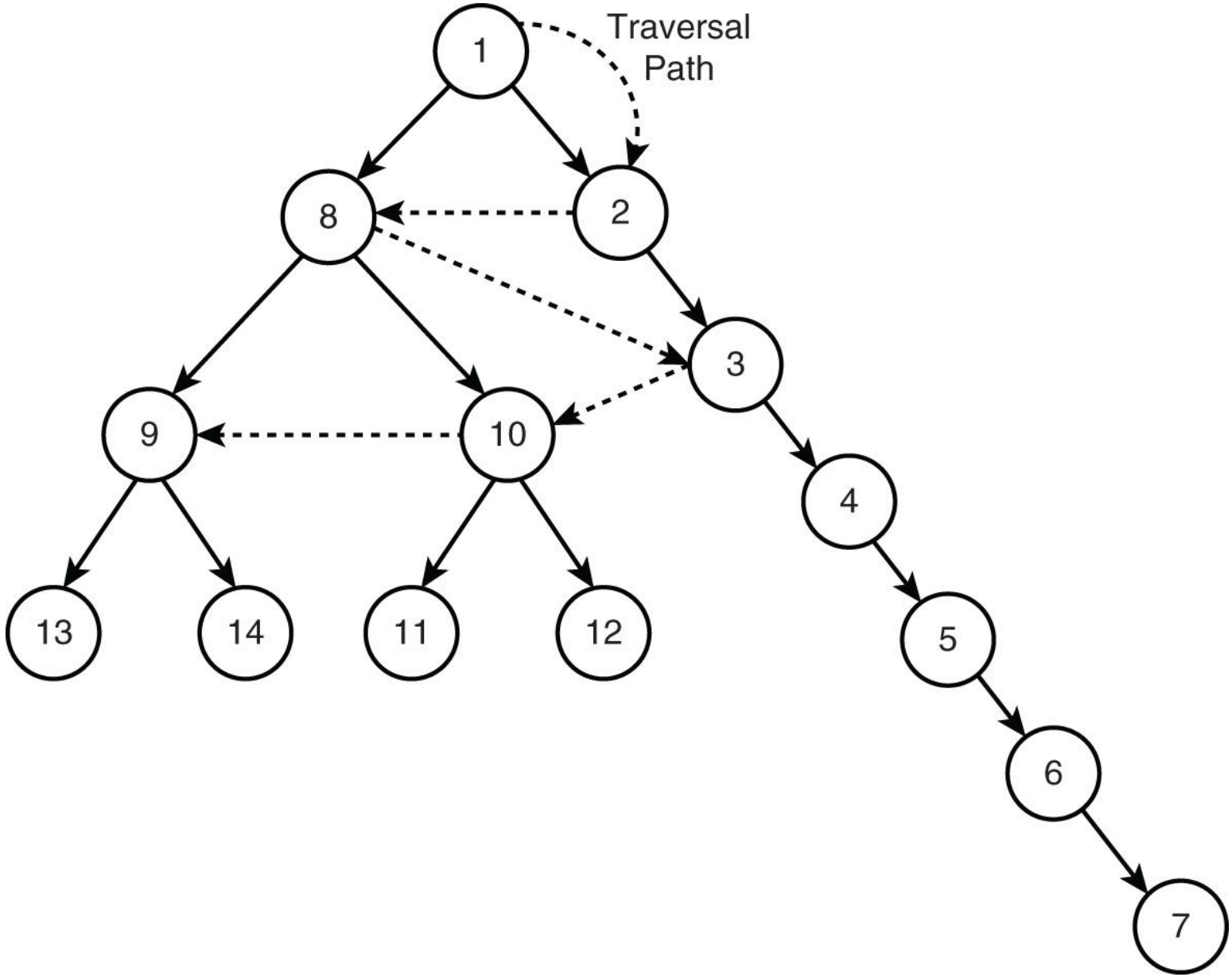

Figure 14.7 Example of a breadth-first search.

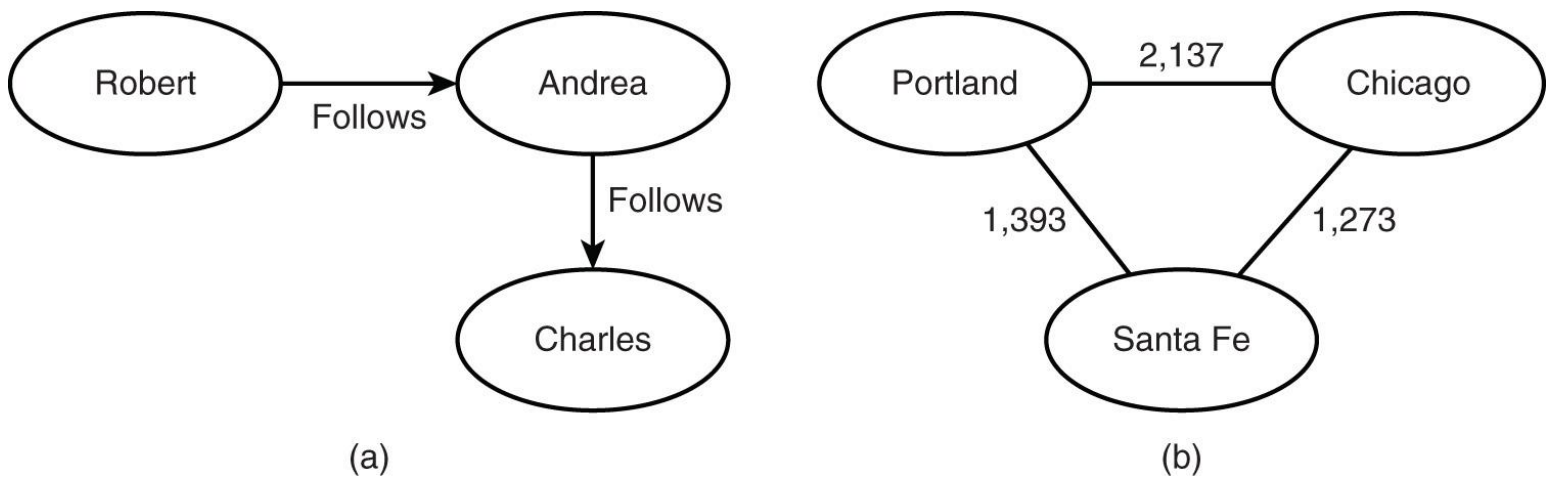

Figure 14.8 Undirected edges are used for symmetrical relations; directed edges are used for nonsymmetrical relations.

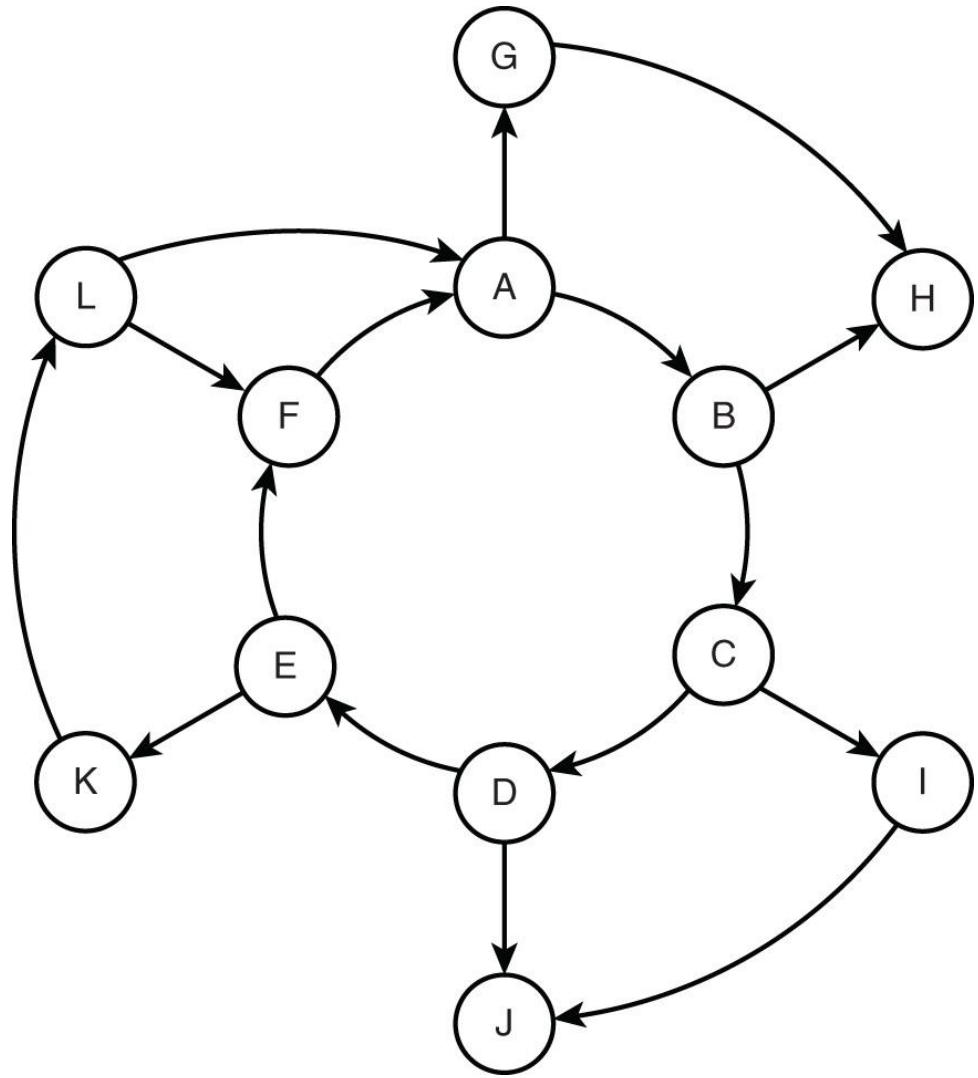

Figure 14.9 Cycles can cause problems with traversal if they are not accounted for in the traversal algorithm.