Module 1:
Different DBs for Different Requirements

Text chapters 1 and 2

University of Illinois
at Springfield

Module 1 - Overview

This chapter places NoSQL databases in a historical context. Many of the techniques used in NoSQL databases have been used in the past.

- **Notice that different data management requirements may be best served by different database management systems.**
- **The limitations of earlier database management systems motivated the development of relational database.**
- **The limitations of relational databases provided the motivation for creating NoSQL databases.**

Early Database Management Systems

- All database models have limitations. Through the history of data management, new database models have been created to address the limitation of earlier models.
- Early DBMSs include:
  - Flat File DMS
  - Hierarchical DMS
  - Network DMS

Flat File Data Management Systems

- Flat files constitute the first data management systems. They suffered substantial limitations, including the following:
  - It is inefficient to access data in any way other than by the way data is organized in the file; for example, by customer ID
  - Changes to file structures require changes to programs
  - Different kinds of data have different security requirements
  - Data can be stored in multiple files, making it difficult to maintain consistent sets of data

Hierarchical Data Management Systems

- **Hierarchical databases improved on flat files by organizing related data in hierarchical structures.**

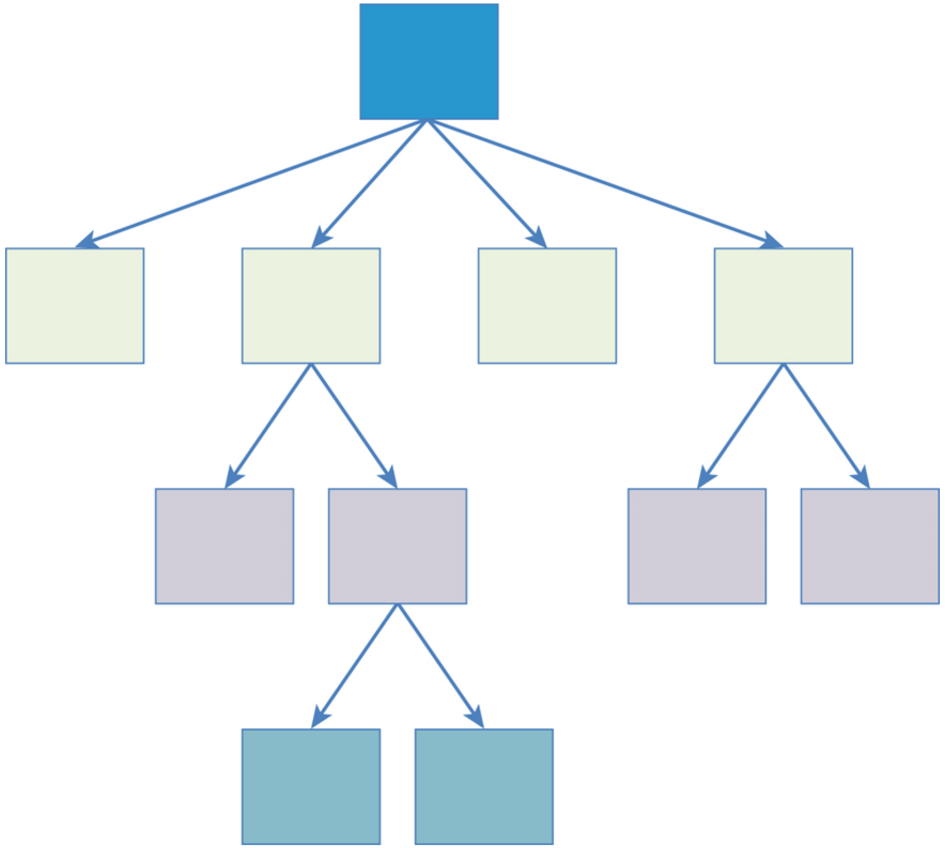


Figure 1.4 The hierarchical model is organized into a set of parent-child relations.

From *NoSQL for Mere Mortals®* by Dan Sullivan (ISBN: 0134023218) Copyright © 2015 Pearson Education, Inc. All rights reserved.

Hierarchical Data Management Systems

- The following are disadvantages of hierarchical databases:
  - Can duplicate data
  - Duplicate data can become inconsistent if one copy is updated but others are not
  - Potential for errors when aggregating in the presence of duplicate data

Network Data Management System

- **Network databases improved on the hierarchical database model. Network databases are not restricted to parent-child relations.**

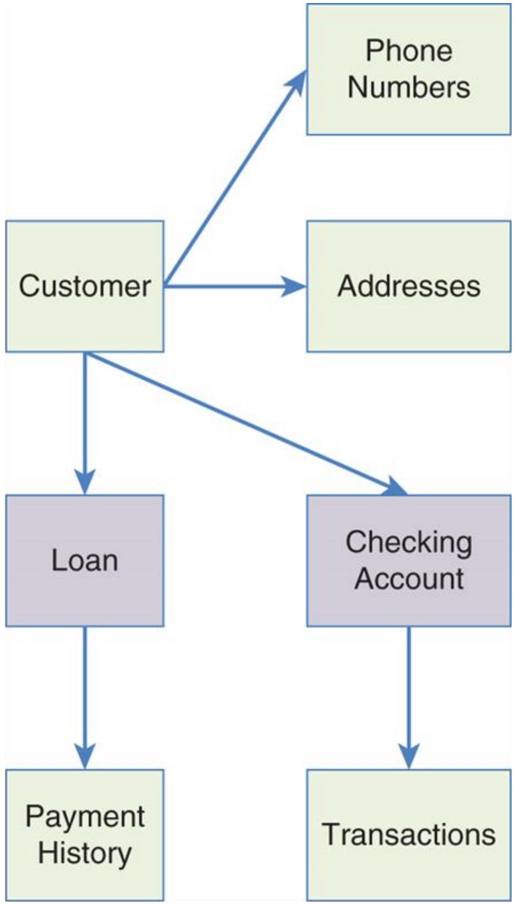


Figure 1.8 A simple network schema shows which entities can link to other entities.

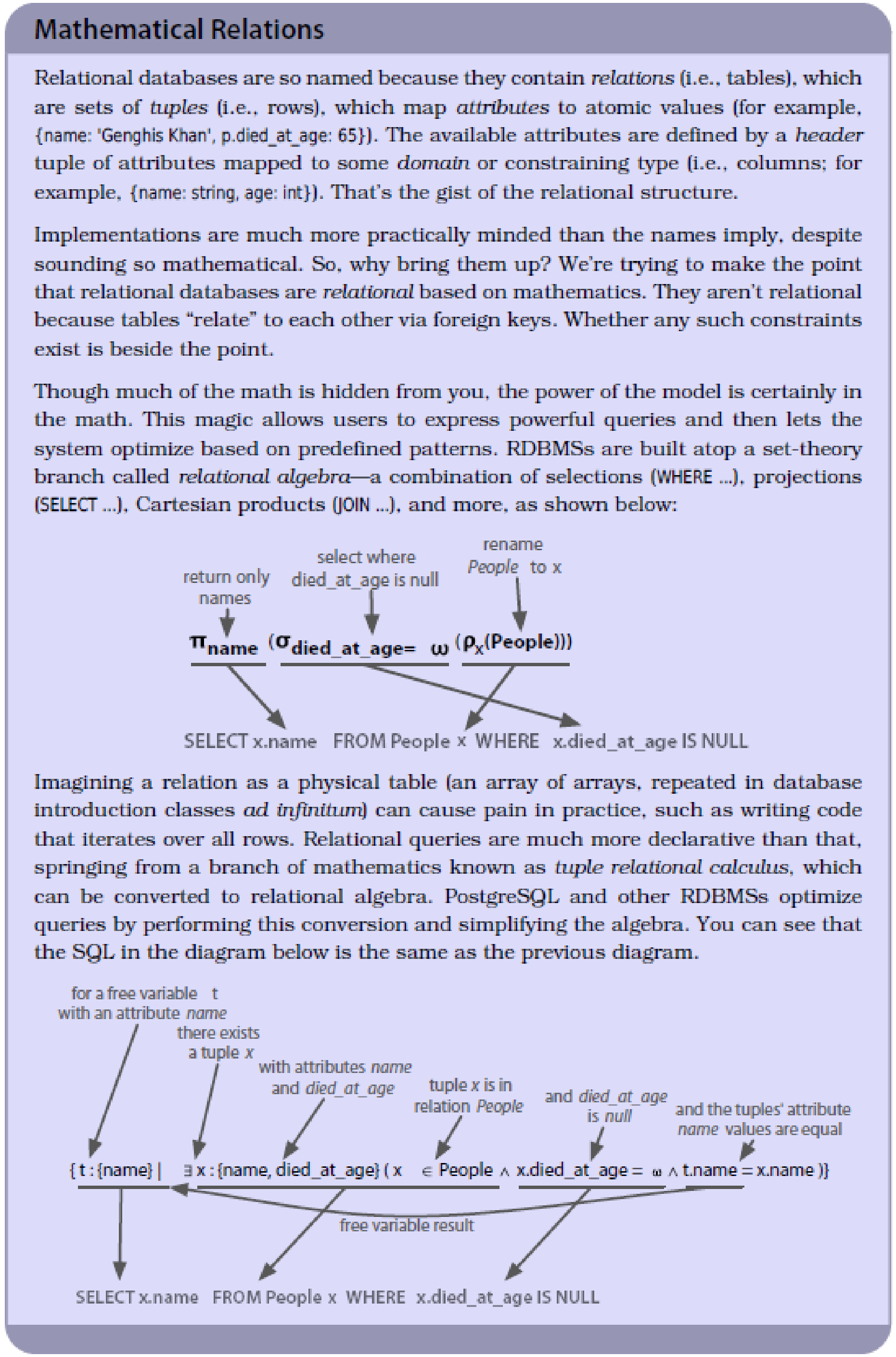
From *NoSQL for Mere Mortals®* by Dan Sullivan
(ISBN: 0134023218) Copyright © 2015 Pearson
Education, Inc. All rights reserved.

Network Data Management System

- The following are disadvantages of network databases:
  - Difficult to design and maintain
  - Changes to the database structure may require changes in the way data is retrieved or updated

Relational Database

- Relational databases were based on a formal mathematical model that used relational algebra to describe data and their relations.



Mathematical Relations

Relational databases are so named because they contain *relations* (i.e., tables), which are sets of *tuples* (i.e., rows), which map *attributes* to atomic values (for example, {name: 'Genghis Khan', p.died_at_age: 65}). The available attributes are defined by a *header* tuple of attributes mapped to some *domain* or constraining type (i.e., columns; for example, {name: string, age: int}). That's the gist of the relational structure.

Implementations are much more practically minded than the names imply, despite sounding so mathematical. So, why bring them up? We're trying to make the point that relational databases are *relational* based on mathematics. They aren't relational because tables "relate" to each other via foreign keys. Whether any such constraints exist is beside the point.

Though much of the math is hidden from you, the power of the model is certainly in the math. This magic allows users to express powerful queries and then lets the system optimize based on predefined patterns. RDBMSs are built atop a set-theory branch called *relational algebra*—a combination of selections (WHERE ...), projections (SELECT ...), Cartesian products (JOIN ...), and more, as shown below:

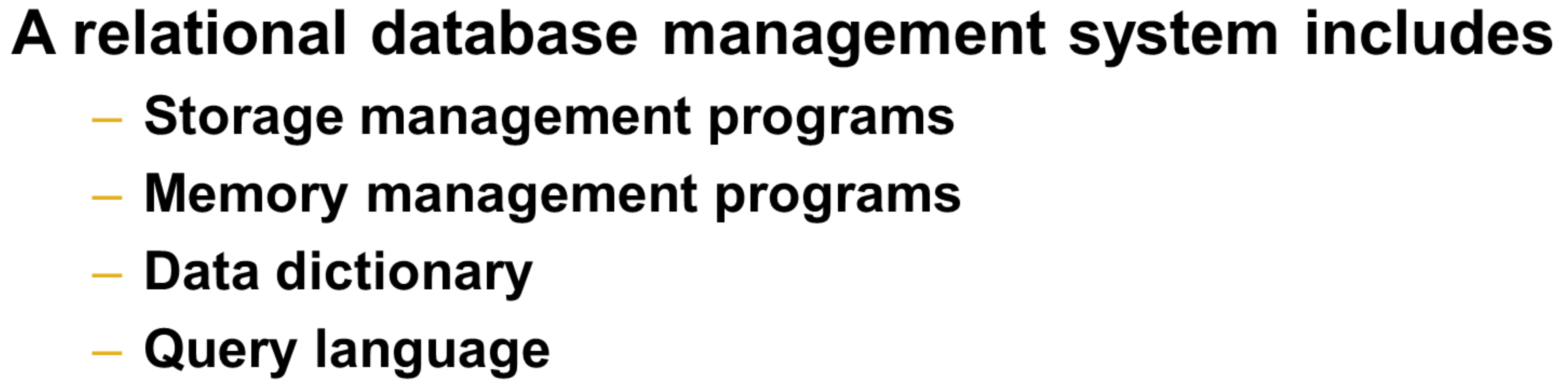
return only names
select where died_at_age is null
rename *People* to x
$\pi_{name} (\sigma_{died\_at\_age= \ \omega} (\rho_x(People)))$
SELECT x.name FROM People x WHERE x.died_at_age IS NULL

Imagining a relation as a physical table (an array of arrays, repeated in database introduction classes *ad infinitum*) can cause pain in practice, such as writing code that iterates over all rows. Relational queries are much more declarative than that, springing from a branch of mathematics known as *tuple relational calculus*, which can be converted to relational algebra. PostgreSQL and other RDBMSs optimize queries by performing this conversion and simplifying the algebra. You can see that the SQL in the diagram below is the same as the previous diagram.

for a free variable t with an attribute *name*
there exists a tuple *x*
with attributes *name* and *died_at_age*
tuple *x* is in relation *People*
and *died_at_age* is *null*
and the tuples' attribute *name* values are equal
$\{ t : \{name\} \mid \exists x : \{name, died\_at\_age\} (x \in People \wedge x.died\_at\_age = \omega \wedge t.name = x.name)\}$
free variable result
SELECT x.name FROM People x WHERE x.died_at_age IS NULL

Relational Database

- Relational databases separated the logical organization of data structures from the physical storage of those structures.



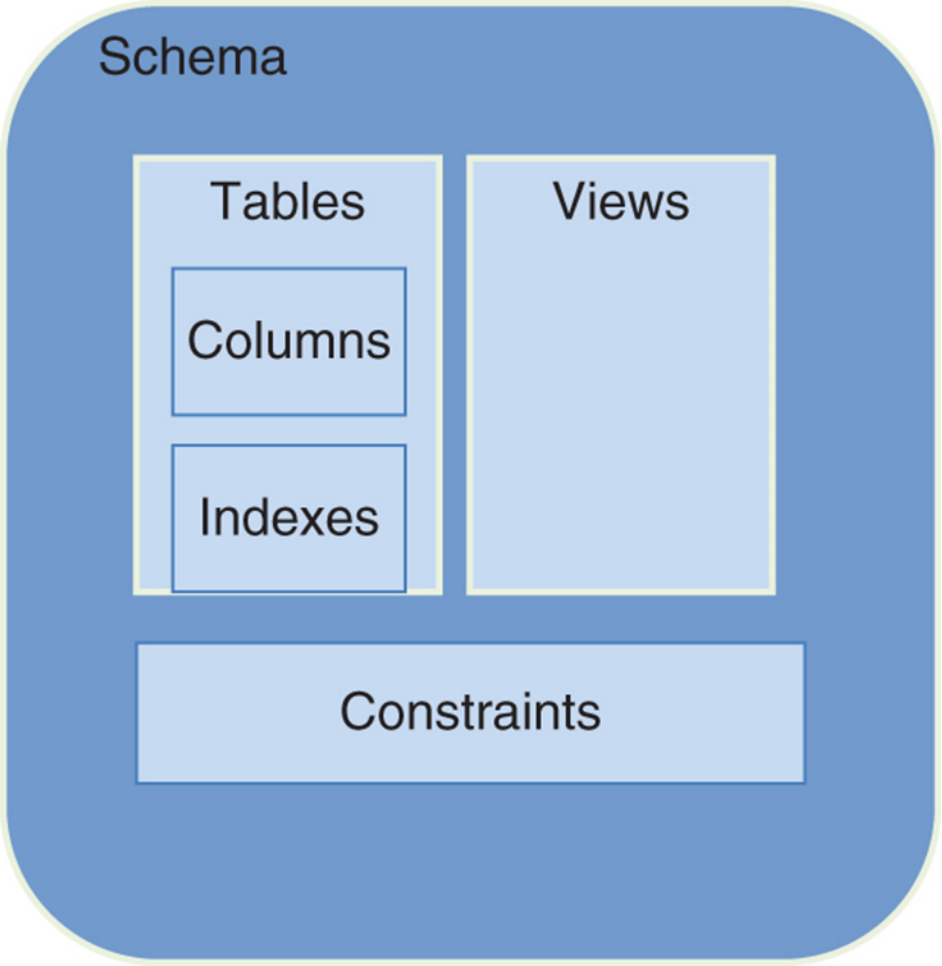
A relational database management system includes

- Storage management programs
- Memory management programs
- Data dictionary
- Query language

Relational Database

- Relational Database models are implemented in a schema

- Relational Database strength is its ACID features (atomicity, consistency, isolation, and durability) .



Relational Database

- **Relational database are difficult to scale horizontally (scale out).**

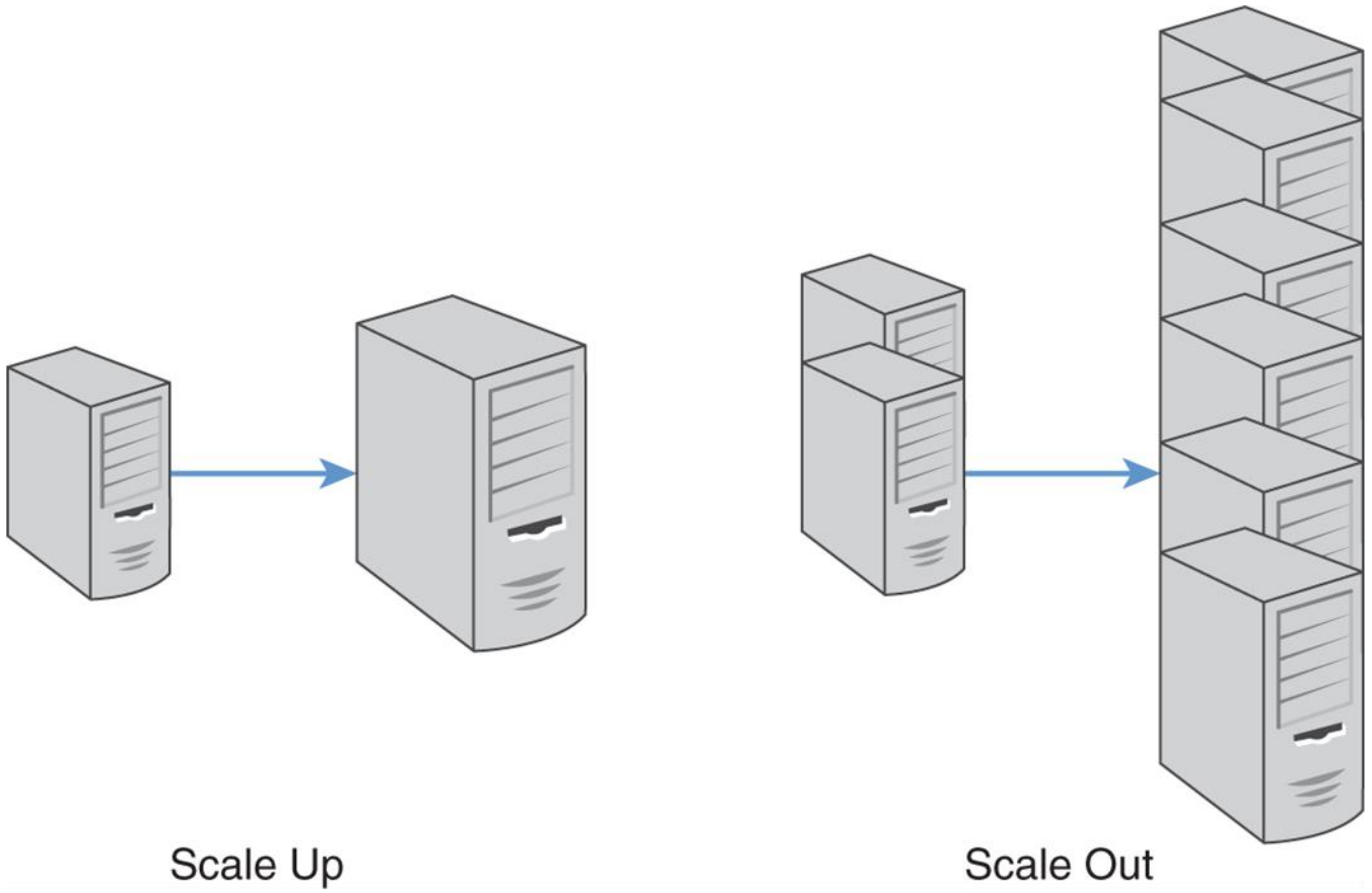


Figure 1.11 Scaling up versus scaling out.

From *NoSQL for Mere Mortals®* by Dan Sullivan (ISBN: 0134023218) Copyright © 2015 Pearson Education, Inc. All rights reserved.

Relational Database

- **NoSQL databases address four key limitations of relational databases:**
  - **Scalability**
  - **Cost**
  - **Flexibility**
  - **Availability**

Data Persistence

- **Data must be stored persistently; that is, it must be stored in a way that data is not lost when the database server is shut down.**

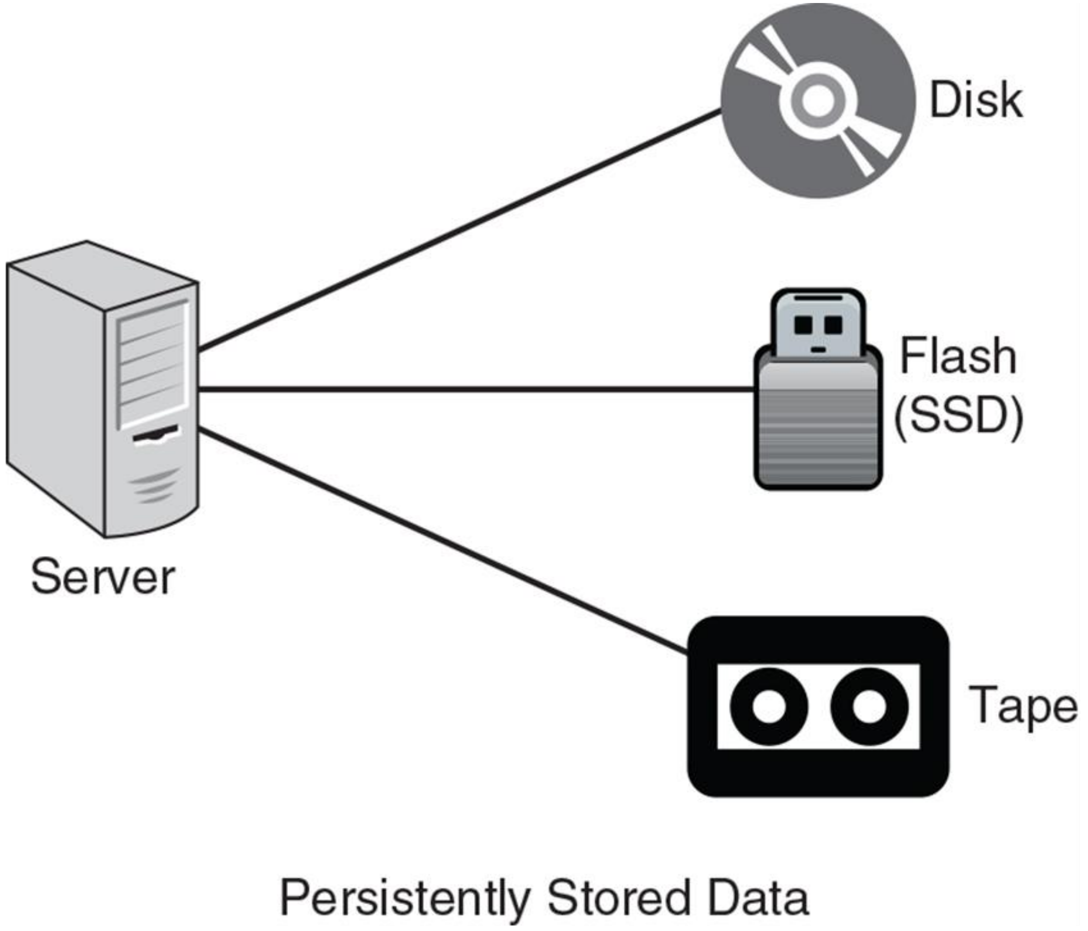


Figure 2.2 Persistently stored data is stored on disk, flash, or other long-term storage medium.

Data Consistency

- Data Consistency in database systems refers to the requirement that any transaction can only change/alter data in allowed ways.
  - Any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.
  - Even Read operations might return inconsistent data if no controls are in place to prevent inconsistent reads.

Data Consistency

- Availability is promoted by maintaining multiple copies of data; if one copy become unavailable, other copies can be used to respond to queries.
- In distributed databases, there is a trade-off between consistency and availability. Increasing the number of copies improves availability but could require longer times to update, leading to longer periods of inconsistent data

CAP Theorem

- The CAP Theorem, also known as Brewer's Theorem, states that distributed databases cannot have consistency (C), availability (A), and partition protection (P) all at the same time.
- Consistency, in this case, means consistent copies of data on different servers.
- Availability refers to providing a response to any query.
- Partition protection means if a network that connects two or more database servers fails, the servers will still be available with consistent data.

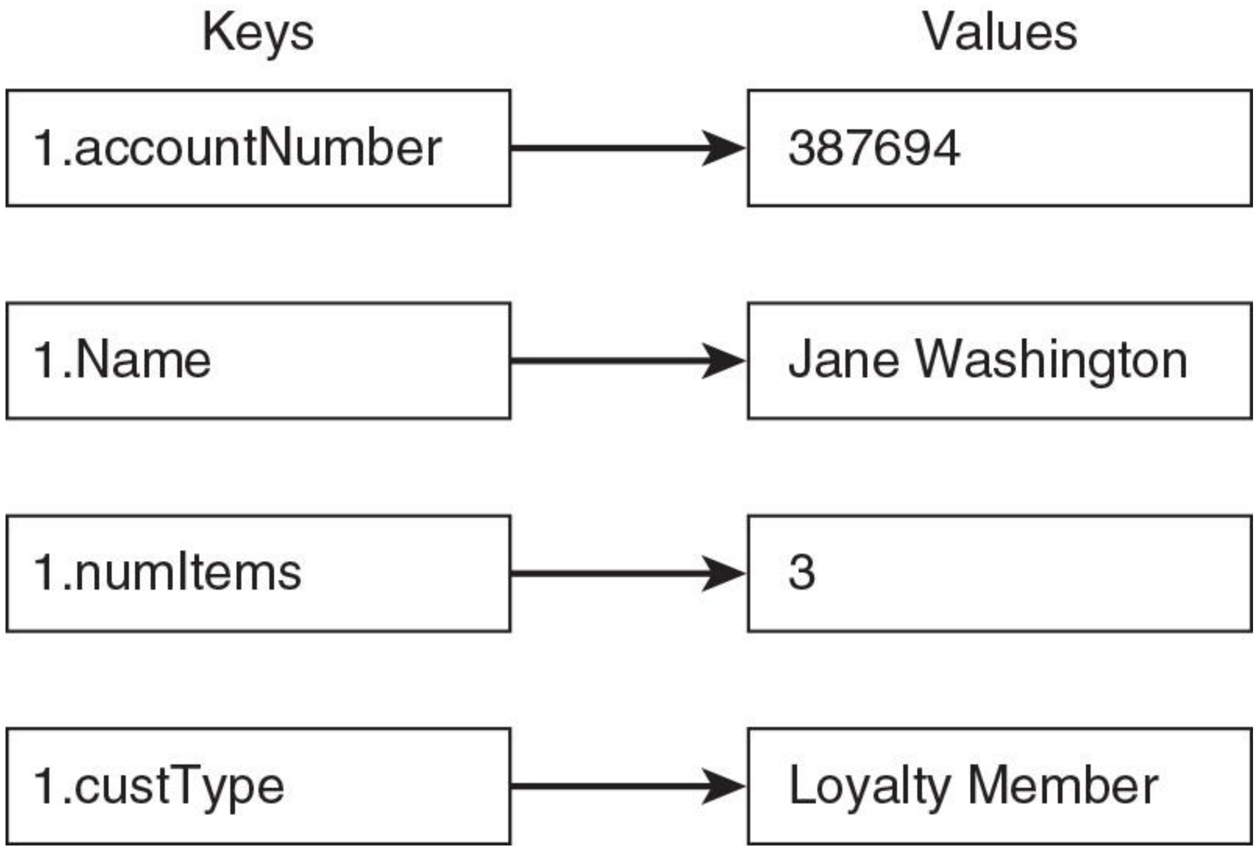
ACID vs BASE

- **ACID features (atomicity, consistency, isolation, and durability) are available in relational databases.**

- **NoSQL databases often support BASE: basically available, soft state, and eventually consistent.**

Key –Value Database

- Key-value pair databases are the simplest form of NoSQL databases;
  - they are modeled on two components: keys and values.



Keys	Values
1.accountNumber	387694
1.Name	Jane Washington
1.numItems	3
1.custType	Loyalty Member

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

Document Database

- Document databases use a key-value approach to storing multiple key value pairs in groups known as documents.
  - Documents are typically in a standard format such as JavaScript Object Notation (JSON) or extensible markup language (XML).

Column Family Database

- Column family databases share some terms with relational databases, such as rows and columns.
- They typically use a map of maps model (that is, the elements of the top-level map are other maps) to store columns and attributes in column families.

Graph Database

- Graph databases are the most specialized of the four NoSQL databases
- Instead of modeling data using columns and rows, a graph database uses structures called nodes and relations (in more formal discussions they are called vertices and edges).
  - A node is an object that has an identifier and a set of attributes.
  - A relation is a link between two nodes that contains attributes about that relation.