

Amazon SimpleDB

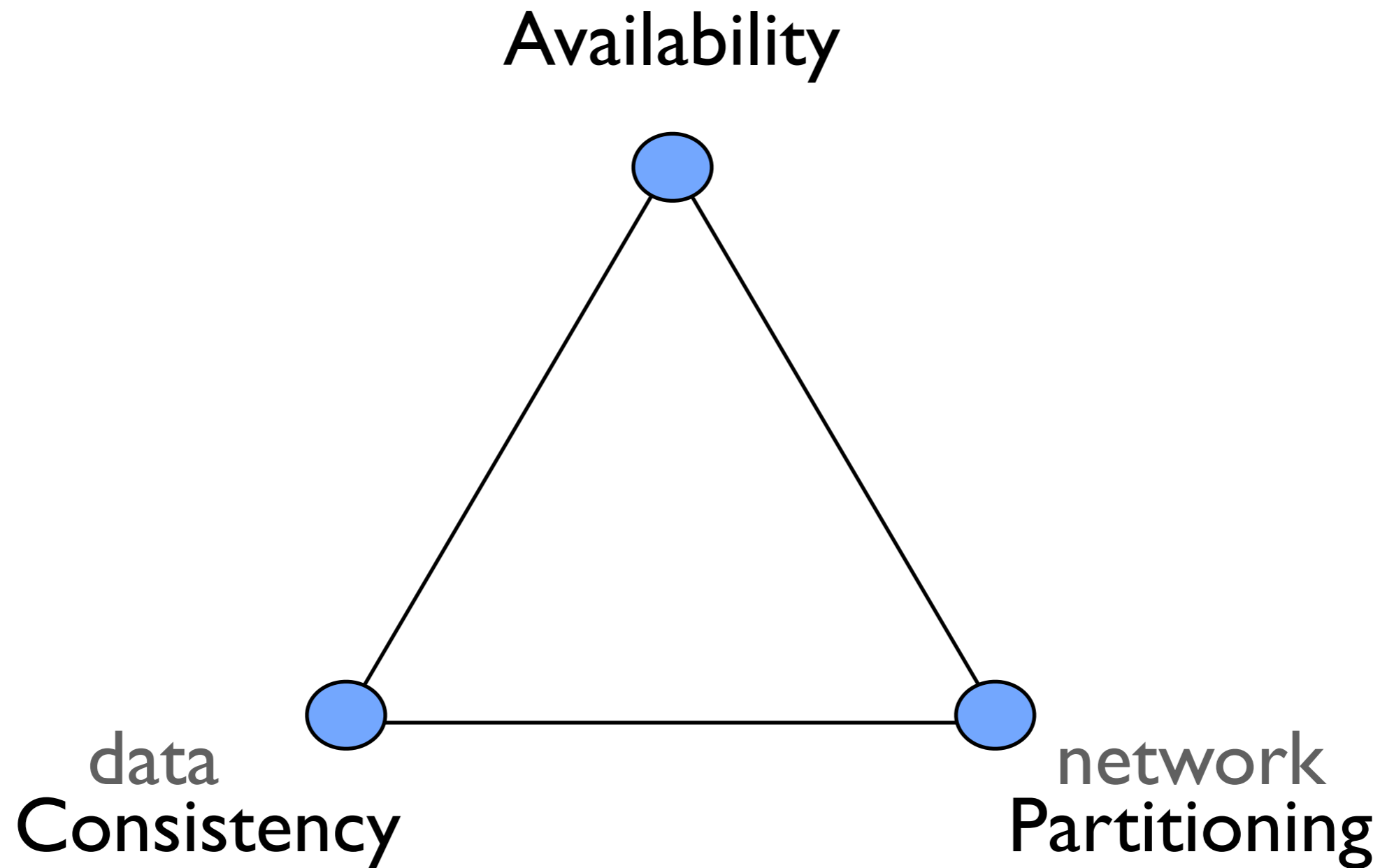
A distributed, highly-scalable, light-weight, query-able,
attribute store...

...Or lets just say a database in-the-cloud

Assumed items

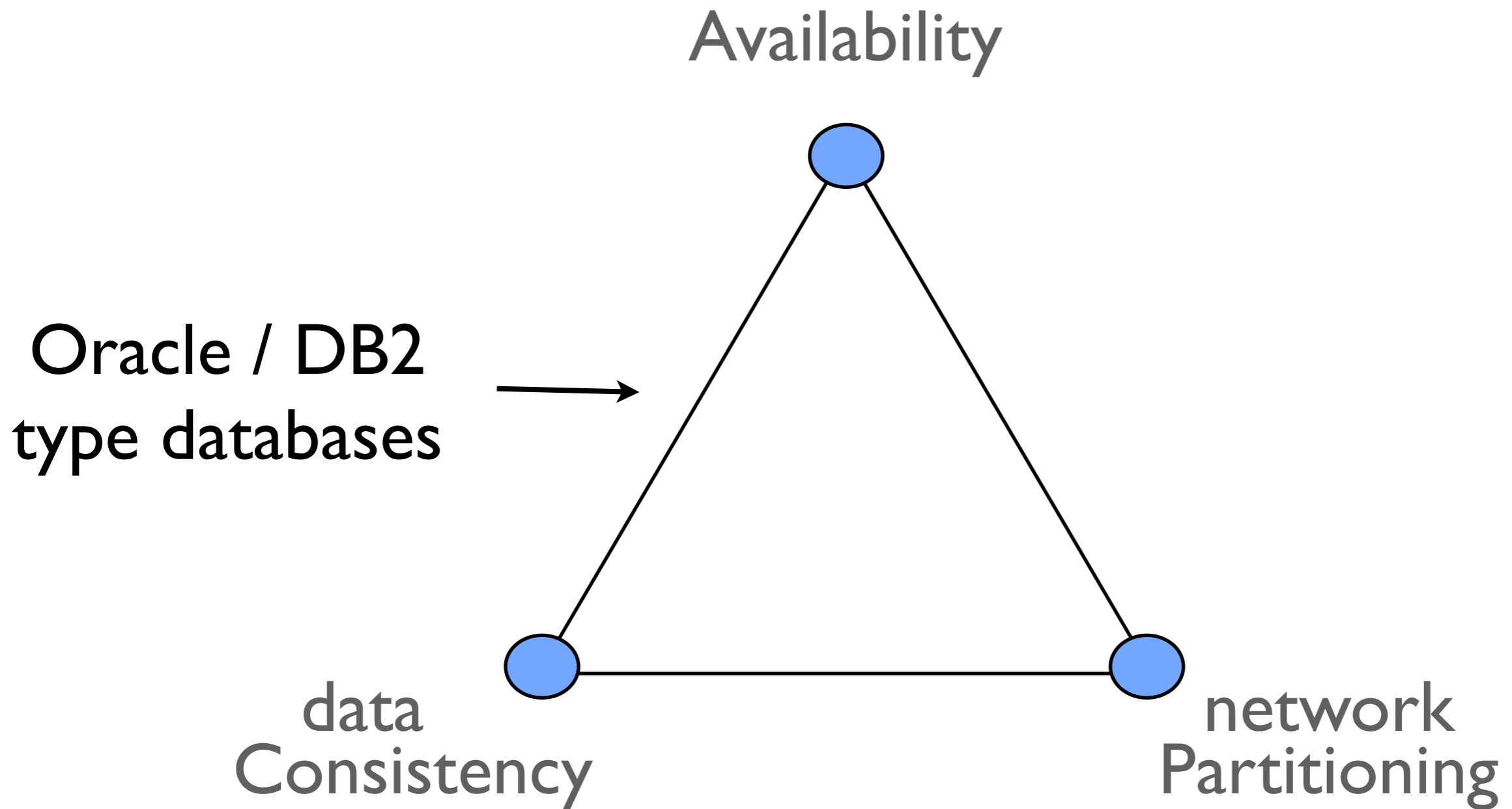
- We won't cover...
- Advantages of the on demand, service model
- No upfront costs
- No IT support staff

CAP Theorem*



* Eric Brewer, 2000

Traditional RDBMS



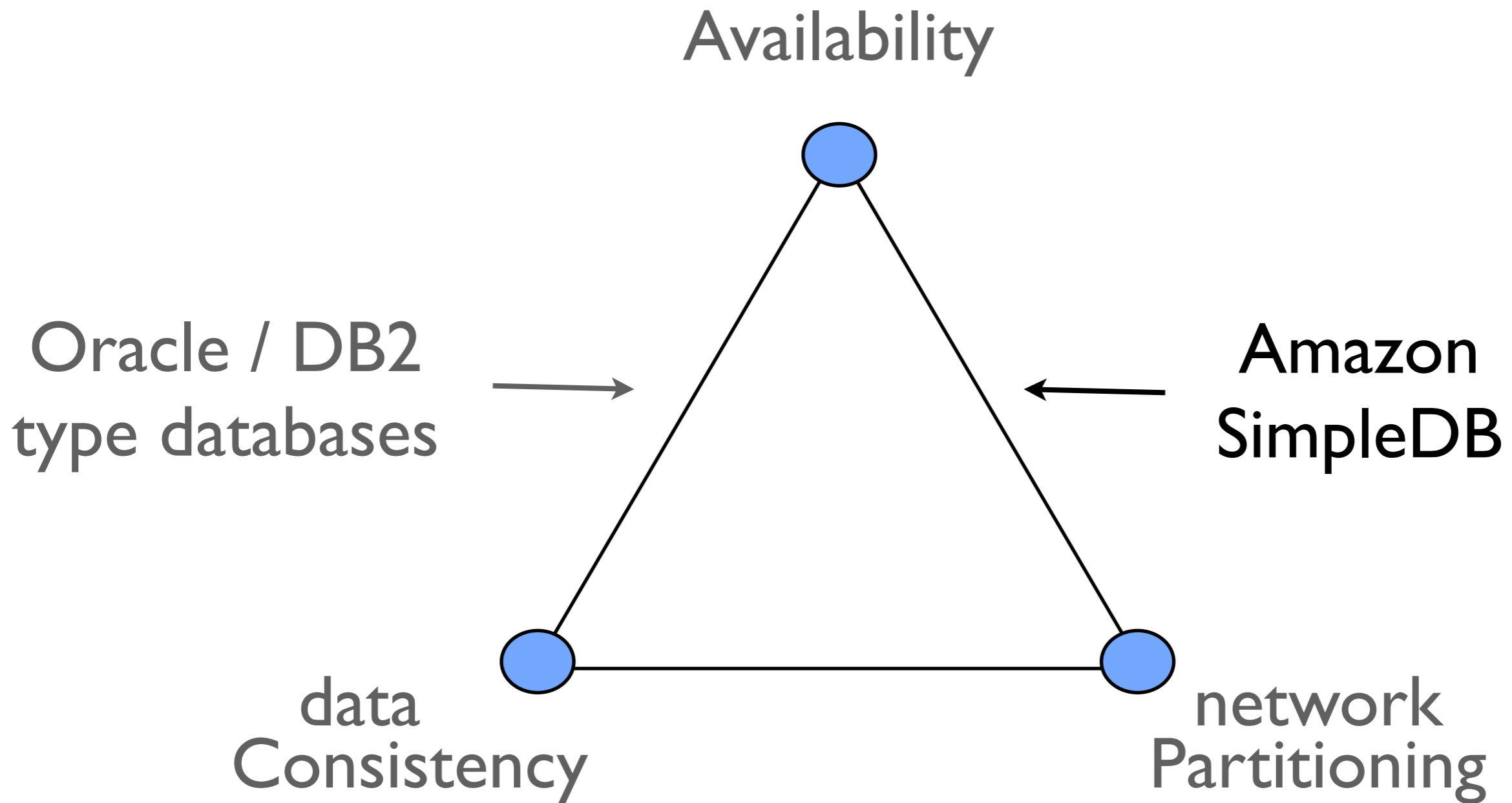
Sacrificing CAP Properties

- What does it mean?
- How can we sacrifice data consistency?
- Different types of data consistency

Sacrificing Consistency

- Client-side
- Server-side
- Eventual

The SimpleDB Model



Developers want to

- Store data
- Process data
- Query data

Probably don't want

- Schema management
- Index management
- Performance tuning
- Data access scaling

Aims

- **Cover about 80% of all database requirements**

Applications which require long-running queries and/or complex table joins, (like data warehouse applications), are probably not a good fit for SimpleDB. RDBMS provide deep functionality, but introduce more complexity (and more cost) than necessary.

- **Availability, durability, and scalability**

All data stored in SimpleDB is replicated multiple times in geographically disbursed data centers, so customer databases don't need to be backed and will automatically fail over to another replica if one is not available.

Requests can be done via https for encryption.

Architecture

Attributes name/value pair, multiple values per name

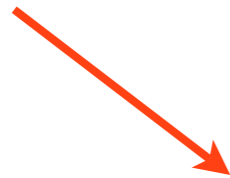
Items consists of multiple attributes, can have different set of attributes for each item in domain

Domain Elastic table structure - No scheme required

The ability to improve your data model on a dynamic, as-needed basis makes SimpleDB a perfect match for agile development.

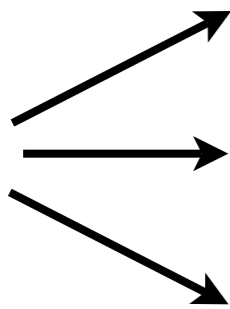
Example

Domain

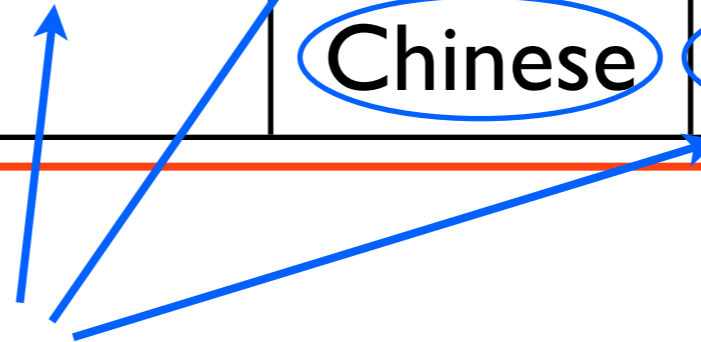


Name	Age	Fav. Food	Corp.
Jane	24		AMZN
Joe	37	Fruit, Pizza, Sushi	
John		Chinese	GOOG

Items



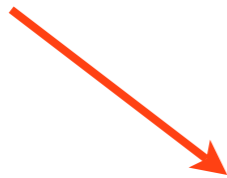
Attributes



Example

Domain

PutAttributes(Joe: (Hair: Red));



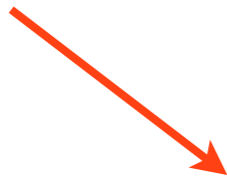
Name	Age	Fav. Food	Corp.
Jane	24		AMZN
Joe	37	Fruit, Pizza, Sushi	
John		Chinese	GOOG

Items →

Example

Domain

PutAttributes(Joe: (Hair: Red));



Items

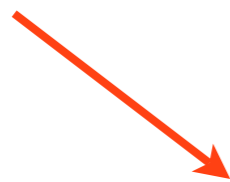


Name	Age	Fav. Food	Corp.	Hair
Jane	24		AMZN	
Joe	37	Fruit, Pizza, Sushi		Red
John		Chinese	GOOG	

Example

Domain

PutAttributes(Sarah: (Age: 13));



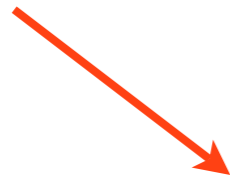
Name	Age	Fav. Food	Corp.	Hair
Jane	24		AMZN	
Joe	37	Fruit, Pizza, Sushi		Red
John		Chinese	GOOG	

Items →

Example

Domain

PutAttributes(Sarah: (Age: 13));



Items →

Name	Age	Fav. Food	Corp.	Hair
Jane	24		AMZN	
Joe	37	Fruit, Pizza, Sushi		Red
John		Chinese	GOOG	
Sarah	13			

SimpleDB API

- **CreateDomain** creates a new named domain within the scope of your AWS account.
- **DeleteDomain** deletes a domain and all of the items within it.
- **ListDomains** returns a list of all of your domains.
- **PutAttributes** creates a new item (if necessary) and adds or replaces attributes.
- **DeleteAttributes** deletes one or more attributes from an item.
- **GetAttributes** returns all or specified attributes of an item.
- **Query** retrieves a set of items which match a query expression. Large result sets can be retrieved in chunks of up to 250 items.

Pricing

- Pay per usage (like all of AWS)
- Bandwidth same for all AWS
- Machine Utilization - \$0.14 per Machine Hour

Also referred to as Query processing; expressed in CPU time (calls return amount of machine time used), not wall clock time like EC2

- Structured Data Storage - \$1.50 per GB-month

Alternatives

- CouchDB - on replicated EC2 clusters
- DIY (proven feasible, but a hassle)

Summary

- SimpleDB makes it easy and straightforward to store and retrieve structured data.
- No creating, maintaining, or migrating database schemas, No monitoring and tuning the query performance, No outgrowing the storage or processing capacity of your database server, No making backups, or replicating data.